Fault-tolerant design of non-linear iterative learning control using neural networks

Krzysztof Patan and Maciej Patan[a]

[a]*Institute of Control and Computation Engineering*
University of Zielona Góra, ul. Szafrana 2, 65-246 Zielona Góra, Poland

Abstract

The design of neural-network-based iterative learning control for non-linear systems is addressed in the setting of a fault tolerant control regime. Taking advantage of the repetitive character of the control task, the inherent uncertainty related to a potential faulty system state can be properly accommodated in terms of a data-driven iterative learning scheme with neural networks used for forward/inverse modeling as well as for controller synthesis. The resulting control technique is supposed to be flexible enough to accurately compensate the faults occurring on both the sensors and actuators and, additionally, take into account the disturbances and noise acting on the system. A complete characterization of the novel fault-tolerant iterative learning scheme is provided including system identification, fault detection and accommodation. Also, the painstaking convergence analysis is presented and the resulting sufficient conditions can be constructively used to determine the update of control law in the consecutive process trial. The excellent performance of the developed control scheme is illustrated by a nontrivial example of tracking control for a magnetic brake system on various scenarios involving actuator and/or sensor faults.

Keywords: iterative learning control, convergence, neural networks, fault tolerance, fault accommodation.

1. Introduction

The complexity of modern industrial control systems has reached the point where requirements imposed on control quality are extremely high according to

URL: `k.patan@issi.uz.zgora.pl, m.patan@issi.uz.zgora.pl` (Krzysztof Patan and Maciej Patan)

required accuracy and reliability of the process, allowing attaining the control goals in a safe and reproducible manner. In the context of non-linear dynamic systems, high quality control is an especially challenging issue which permanently requires the development of new systematic approaches that broaden the area of potential applications. One of the major problems underlying control design is the question of how to provide the robustness of a system to potential faults and still achieve satisfactory behavior in terms of the assumed criteria defining control efficiency without significant deterioration of system performance. In light of this, in the last 20 years, fault-tolerant control (FTC) has become an attractive area of research with many successful approaches and a strong record of tangible engineering applications (Blanke et al., 2016; Noura et al., 2003; Ducard, 2009; Patan, 2019; Rouabah et al., 2022; Conchas et al., 2023).

This is of paramount importance in the case of repetitive processes encountered in industrial environment as robustness to faults can be equally important to the performance of the system, thus reducing the cost of operation and maintenance. Under the repetitive work regime, it can be observed that in each replicated trial the existing closed-loop control algorithms produce the same tracking error with the same features, such as settling time, overshoot, oscillations, etc. That property and knowledge of system behavior from previous trials could be used to improve control performance. In this context, iterative learning control (ILC) emerged in the late 1970s, providing a solution to the accurate tracking control problem and establishing a popular area of control research, cf. Arimoto et al. (1984); Bristow et al. (2006); Ahn et al. (2007); Freeman et al. (2015); Owens (2016) for seminal papers and monographs. As for the class of linear time invariant systems, the theory is well developed, with a rich spectrum of applications. However, most of the existing ILC approaches are related to the linear form of the control law (Tao et al., 2017; Chen et al., 2022). For a class of non-linear processes with a repeatable regime of operations, there is still no theory of unified control synthesis and analysis, even if the need for such solutions is well recognized (Moore, 1993). The proposed ILC methods for control-affine, linear parameter varying or general non-linear systems are mainly based on the linear iterative controller (Xu and Tan, 2003; Bu et al., 2012; Miao and Li,

2017). Non-linear learning controller design based on parameter estimation was proposed by Bu et al. (2020). However, it is needed to transform the system to iteration-dependent time-varying form. A somewhat similar approach was proposed by Yu et al. (2021), in which dynamic linearization was used to represent a learning controller first and then ILC was formulated using input-output data. In both approaches, however, a simplified form of the learning controller is finally derived.

Although the idea of using neural networks in the context of ILC is not new, dating back to the early 1990s, there is still only a handful of ILC systems realized by means of neural networks. This is mainly due to the significant difficulty of satisfying the convergence and stability of both the control update and network training (Chow et al., 2000; Chi and Hou, 2009; Xiong et al., 2016; Wei et al., 2017). Therefore, their applicability usually is limited to a specific class of non-linear systems, and there is no straightforward extension to FTC. An approach using a radial basis function neural network was presented by Liu and Hou (2022) to deal with non-linear component of the control law. An interesting application of neural networks was discussed by Zhang et al. (2021), who used a neural network to predict ILC outputs instead of storing a large amount of data. However, the approach was derived for linear systems, and a linear learning controller was used. An efficient neural-network-based approach for ILC synthesis was provided in our previous works (Patan and Patan, 2020; Patan and Patan, 2021), where a non-linear controller was developed. The proposed controller is of a general nature and can be used to design different types of controllers, e.g., P-, D- or even PD-type.

In the setting of FTC, we need the control design to be robust to disturbances from various sources acting on the plant (Chien, 1998), thus leading to the idea of a learning controller able to adapt to potential faults appearing in the system. One of the crucial obstacles of non-linear control design is an accurate non-linear model calibration of the plant directly affecting the performance of the closed-loop controller. A reasonable remedy is to consider learning controllers that also have a non-linear or time-varying structure (Moore, 1993). The communications on FTC for iterative control are very limited. The approach presented by Li et al. (2022) was dedicated to linear stochastic repetitive sys-

tems. In turn, iterative learning control able to deal with actuator faults was discussed by Liu and Hou (2022) or Huang et al. (2021), although without an explicit fault accommodation mechanism. Fault estimation and accommodation in the framework of non-linear ILC was discussed in the authors' previous works (Patan et al., 2020; Patan and Patan, 2022), although covering separately the issues of sensor and actuator faults, respectively.

To address the above limitations and needs, the main purpose of the paper is to elaborate an efficient approach to fault-tolerant control for the class of non-linear repetitive dynamic systems with special attention paid to addressing the problems of accurate modeling, reliable fault detection and a control design able to compensate potential faults supplemented with proper convergence analysis. In particular, artificial neural networks, whose universal approximation abilities for a broad class of non-linear systems are widely known, are used to design a novel ILC scheme. It consists of three neural-network-based subsystems: a mixture of neural networks for robust forward modeling of a plant, an inverse neural model and a neural controller. Such a homogeneous structure renders it possible to effectively accommodate both sensor and actuator faults in a unified manner. The whole scheme is purely data-driven, i.e., a learning controller is capable to adapt to changing working conditions of the plant by the training process using data from previous trials. The advantage of the approach reported here is that it is supplemented with careful analysis, leading to constructive convergence conditions for the control update.

Summarizing, the contributions of the paper are as follows:

1. developing an integrated actuator and sensor fault-tolerant control system based on a novel non-linear iterative learning control scheme and neural network models;
2. providing sufficient conditions for convergence of the proposed control approach;
3. experimental verification of the proposed neural-network-based fault-tolerant control system using a tracking control example of a magnetic brake system involving actuator and/or sensor faults.

The paper is organized as follows. In Section 2, the system representation along with details concerning the learning controller is provided. Development

of both forward and inverse neural-network-based models is presented in Section 3. Section 4 discusses the proposed fault tolerant control system. In the subsequent section, convergence analysis is considered. Section 6 is devoted to experimental verification of the proposed fault-tolerant iterative learning control system. Finally, the paper is summarized in Section 7.

2. System representation

Let us consider a single-input single-output discrete-time dynamical system in the state-space:

$$\begin{aligned} \boldsymbol{x}(k+1) &= g(\boldsymbol{x}(k), u(k)), \\ y(k) &= \boldsymbol{C}^{\mathsf{T}} \boldsymbol{x}(k), \end{aligned} \tag{1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ represents the state vector, k is the discrete-time instant, n stands for the system order, $u(k) \in \mathbb{R}^1$ and $y(k) \in \mathbb{R}^1$ are the input and output of the system, $\boldsymbol{C} \in \mathbb{R}^n$ is the output vector, $g(\cdot,\cdot)$ represents an unknown smooth and invertible non-linear function, and T represents the matrix transposition operator. In the absence of faults (nominal conditions), the system (1) is represented by the following healthy model:

$$\begin{aligned} \boldsymbol{x}_N(k+1) &= \hat{g}(\boldsymbol{x}_N(k), u(k)), \\ \hat{y}(k) &= \boldsymbol{C}^{\mathsf{T}} \boldsymbol{x}_N(k), \end{aligned} \tag{2}$$

where $\boldsymbol{x}_N(k) \in \mathbb{R}^n$ represents the model state vector, $\hat{y}(k) \in \mathbb{R}^1$ is the model output, and $\hat{g}(\cdot,\cdot)$ is an approximation of the function $g(\cdot,\cdot)$. In the paper, it is assumed that faults which affect the system are of abrupt nature. Thus, the system representation considering both actuator and sensor faults is described in the state-space as follows:

$$\begin{aligned} \boldsymbol{x}(k+1) &= g(\boldsymbol{x}(k), u_f(k)), \\ y_f(k) &= \boldsymbol{C}^{\mathsf{T}} \boldsymbol{x}(k) + f_s(k)\beta(k - T_1), \end{aligned} \tag{3}$$

where $y_f(k) \in \mathbb{R}^1$ represents the faulty system output and $u_f(k) \in \mathbb{R}^1$ is the faulty input defined as

$$u_f(k) = u(k) + \beta(k - T_2) f_a(k), \tag{4}$$

where $f_a(k) \in \mathbb{R}^1$ and $f_s(k) \in \mathbb{R}^1$ are functions representing the change observed respectively in the actuator and sensor due to a fault and the time profile of a fault has the following form:

$$\beta(k-T) = \begin{cases} 0 & \text{if } k < T, \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

Finally, T_1 and T_2 represent the time of occurrence of a sensor and actuator fault, respectively. It should be stressed that in this research noise and disturbances acting upon the plant are not considered. However, they can be easily incorporated into the system representation as proposed in the authors' previous work (Patan and Patan, 2021).

In this paper, the system (1) is controlled by means of a non-linear iterative learning control strategy. This class of control algorithms has proven its applicability in many industrial areas, especially in cases of repetitive processes or when plant to be controlled works in the repetitive regime of operations (Bristow et al., 2006; Freeman et al., 2015; Oomen and Rojas, 2017). The general form of open-loop non-linear learning control is

$$u_{p+1}(k) = f(u_p(k), e_p(k)), \tag{6}$$

where $f(\cdot,\cdot)$ denotes a non-linear function representing dynamics of the controller, p stands for the operation cycle number (the so-called trial), $e_p(k)$ stands for the tracking error at the p-th trial defined as $e_p(k) = y_d(k) - y_p(k)$, where $y_d(k)$ is the reference (desired) signal, and u_p is the control signal at the p-th trial. The signals $u_p(k)$ and $e_p(k)$ are recorded during the p-th trial and stored in the memory for later use to derive control for the next operation cycle (see the upper part of the scheme presented in Fig. 2). Most solutions regarding ILC design consider the function $f(\cdot,\cdot)$ of a linear form with fixed parameters (Bristow et al., 2006). However, complex industrial plants with highly non-linear characteristics may enforce the application of much more sophisticated solutions (Moore, 1993). The learning controller may have a non-linear structure and, what is even more important, it may be time-varying. Both features can be easily achieved by means of artificial neural networks. Depending on the application, the learning controller may have a static or dynamic form, as

reported by Patan (2019). In this paper, a static form of the learning controller is considered by application of the well known feed-forward neural network. The controller representation is given as follows:

$$u_{p+1}(k) = \boldsymbol{W}^c_{3,p}\sigma_o(\boldsymbol{W}^c_{2,p}\sigma_h(\boldsymbol{W}^c_{1,p}\boldsymbol{\varphi}^c_p(k))), \tag{7}$$

where $\boldsymbol{W}^c_{1,p} \in \mathbb{R}^{n_h\times 3}$, $\boldsymbol{W}^c_{2,p} \in \mathbb{R}^{1\times n_h}$ and $w^c_{3,p} \in \mathbb{R}^1$ are adaptable controller weight parameters, $\sigma_h(\cdot)$ and $\sigma_o(\cdot)$ are the activation functions of the first and the second hidden layer, respectively, and n_h is the number of hidden units of the first layer. Finally, the regression vector $\boldsymbol{\varphi}^c_p(k) = [u_p(k), e_p(k), 1]^\mathsf{T}$ determines the P-type learning controller.

In order to adapt the controller to changing operational conditions of the plant, its parameters are subject to training after each operation cycle. Let us define the training process as minimization of the cost function defined based on the difference between the controller output and the desired input:

$$\theta^*_p = \arg\min_\theta J, \tag{8}$$

where θ_p is the vector of stacked elements of controller weight matrices, θ^*_p is the optimal controller parameter vector derived at the p-th trial, and the cost function has the form

$$J = \frac{1}{2}\sum_k (u_d(k) - u_p(k))^2, \tag{9}$$

where $u_d(k)$ is the desired control signal and $u_p(k)$ is the output of the neural controller. The desired control signal is derived by means of an inverse model discussed further in this paper. To adapt the controller parameters after each trial, the update rule based on a stochastic gradient is used:

$$\theta_{p+1} = \theta_p - \eta\frac{\partial J}{\partial \theta_p}, \tag{10}$$

where η is the learning rate. Applying the well known back-propagation idea, the following set of update rules can be determined:

- for the weight $w^c_{3,p}$:

$$\frac{\partial J}{\partial w^c_{3,p}} = \delta^3(k)\sigma_o(\cdot), \quad \delta^3(k) = \varepsilon_p(k), \tag{11}$$

where $\varepsilon_p(k) = u_d(k) - u_p(k)$;

- for the weight matrix $\boldsymbol{W}_{2,p}^{c}$:

$$\frac{\partial J}{\partial \boldsymbol{W}_{2,p}^{c}} = \delta^{2}(k)\sigma_{h}^{\mathsf{T}}(\cdot), \quad \delta^{2}(k) = \sigma_{o}^{'} \circ (w_{3,p}^{c}\delta^{3}(k)), \tag{12}$$

  where $\sigma_{o}^{'}$ is the derivative of the function $\sigma_{o}(\cdot)$ and the operator $\circ$ stands for the Hadamard product;

- for the weight matrix $\boldsymbol{W}_{1,p}^{c}$:

$$\frac{\partial J}{\partial \boldsymbol{W}_{1,p}^{c}} = \delta^{1}(k)\boldsymbol{\varphi}_{p}^{\mathsf{T}}(\cdot), \quad \delta^{1}(k) = \sigma_{h}^{'} \circ ((\boldsymbol{W}_{2,p}^{c})^{\mathsf{T}}\delta^{2}(k)), \tag{13}$$

  where $\sigma_{h}^{'}$ is the derivative of the function $\sigma_{h}(\cdot)$.

A crucial problem when dealing with control systems is the detection of possible faults which can occur in the plant. The significance of this issue follows from the fact that feedback or adaptive types of control systems can hide faults from being observed. Then proper compensation/accommodation of faults is not possible, especially in case of sensor faults. A common way to realize fault detection is a model-based approach, where the decision about faults is made using the forward model of the plant. Any change in plant operation caused by a fault can be detected comparing the output of the model representing the nominal working conditions of the plant with the output of the faulty plant. Thus, the model of the plant which has to be designed first plays a relevant role. Moreover, the model of the plant can be used to develop a learning controller, as reported in our previous works (Patan, 2019). Unfortunately, the forward model itself is insufficient to carry out fault isolation. In particular, there is a need to distinguish sensor faults from other types (actuator faults as well as faults in plant dynamics) due to the fact that they do not change the dynamic behavior of the plant at all. That is the reason for using another, inverse model of the plant. Comparing the output of the inverse model with the actual control signal, we are able to distinguish sensor faults from actuator and process ones. Moreover, in this work the inverse model is used also to train the controller with the adaptive rules (10)–(13). Both forward and inverse models can be constructed by means of an artificial neural network of the dynamic kind. Structural details are provided in the forthcoming sections.

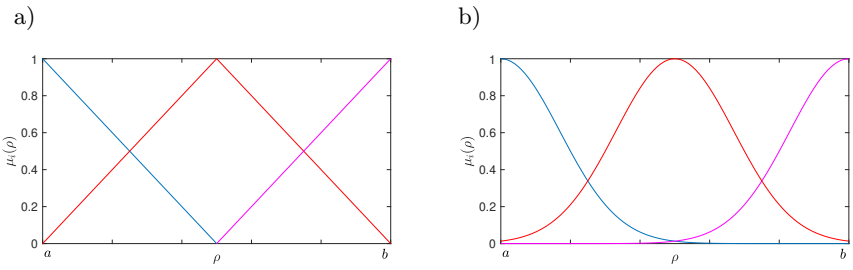

Figure 1: Arrangement of membership functions: triangular (a), Gaussian (b).

3. Models for fault diagnosis

3.1. Forward model

In order to design a fault diagnosis system which is robust to disturbances and a model mismatch, the idea of a multi-model is applied here. In particular, the problem of proper system dynamics reproduction for different operating points is investigated. Each model approximates system behavior at a particular operating point. The output of the multi-model is achieved by activate sub-models for which the nominal working conditions are close to the actual operating point ρ. To determine which model should be activated, we adopt here the idea of a membership function known from fuzzy logic theory. Furthermore, as we deal with control system design, the operating point is chosen to be the initial value of the control signal at each trial u_0. For each operating point, the membership functions should sum to unity as follows:

$$\sum_{i=1}^{M} \mu_i(\rho) = 1, \tag{14}$$

where $\mu_i(\rho)$ is the membership function of the i-th sub-model and M is the number of sub-models. Commonly used functions which satisfy the condition (14) are triangular or Gaussian ones, as shown in Fig. 1. The difference is the number of activated sub-models. In the case of a triangular membership function, only two models can be activated at the same time (see Fig. 1(a)), contrary to Gaussian models, where more sub-models take part in the estimation of the system output (Fig. 1(b)).

Although neural networks are known to possess the universal approximation property (Haykin, 2009; Nørgaard et al., 2000; Patan, 2019), proper generalization of the system response is often achieved at the cost of a bias. The mixture

of models, if properly adopted, can reduce the distribution of the model output, still keeping the bias on a low level. As a result, we are able to determine a good trade-off between uncertainty modeling and robustness with respect to disturbances. Then, the forward model is represented as follows (Patan et al., 2020):

$$\hat{y}(k) = \sum_{i=1}^{M} \hat{y}_i(k)\mu_i(\rho), \tag{15}$$

where $\hat{y}_i(k)$ denotes estimates of the system output at the i-th operating point. The model (15) can be perceived as non-linear gain scheduling (Leith and Leithead, 2000) or an ensemble of models used for regression (Sollich and Krogh, 1996). Each model $\mathcal{M}_i$ in the ensemble (see Fig. 2) is designed by means of the state-space neural network as follows:

$$\begin{aligned} \mathcal{M}_i : \hat{\boldsymbol{x}}_i(k+1) &= g_i(\hat{\boldsymbol{x}}_i(k), u(k)), \\ \hat{y}_i(k) &= \boldsymbol{C}^T \hat{\boldsymbol{x}}_i(k), \end{aligned} \tag{16}$$

with the function $g_i(\cdot,\cdot)$ defining the i-th neural model represented by

$$\hat{\boldsymbol{x}}_i(k+1) = \boldsymbol{W}^f_{i,2}\sigma_h(\boldsymbol{W}^f_{i,1x}\hat{\boldsymbol{x}}_i(k) + \boldsymbol{W}^f_{i,1u}u(k)), \tag{17}$$

where $\sigma_h(\cdot)$ is the non-linear activation function of the hidden layer, while $\boldsymbol{W}^f_{i,1x}$, $\boldsymbol{W}^f_{i,1u}$ and $\boldsymbol{W}^f_{i,2}$ are the i-th forward model weight matrices of appropriate size. Each neural network (17) is trained using data recorded during control of a plant at the specific operating point.

The ensemble is constructed using membership functions distributed over the variable ρ. The commonly used triangular and Gaussian membership functions and their distribution over the interval $\rho \in [a, b]$ are illustrated in Fig. 1(a) and (b), respectively. In this paper, we decided to apply Gaussian functions distributed over the operating range of a plant considered. In this case, the operating point is assigned to the initial value of the control signal at each trial p, i.e., $u_0 = u_p(0)$. Thus,

$$\mu_i(u_0) = e^{-\frac{(u_0-u_i)^2}{2\sigma^2}}, \tag{18}$$

where u_i is the i-th operating point and σ is the spread of the Gaussian function. By applying the same spread to each membership function we can guarantee the condition (14) to be always satisfied.

3.2. Inverse model

Taking into account the fact that the relation between the state and input is non-linear in nature, direct derivation of an inverse of $g(\cdot,\cdot)$ from the model (3) can be hard or even impossible. Moreover, to obtain a good quality fault estimator, real signals available for measurements should be employed. In this context one can employ inverse modeling of an input-output form which, using artificial neural networks, is a quite straightforward approach. The inverse model of (3) is given as follows:

$$\hat{u}_f(k) = h(y(k), \ldots, y(k-n), \hat{u}_f(k-1), \ldots, \hat{u}_f(k-m)), \tag{19}$$

where $\hat{u}_f(k)$ and $y(k)$ are estimates of the input affected by a fault and the system output at the time instant k, respectively, n and m are numbers representing past outputs and estimated inputs used, respectively, and $h(\cdot)$ represents the unknown non-linear mapping to be found. The inverse model (19) can be obtained by means of a neural network with external dynamics (Haykin, 2009; Nørgaard et al., 2000). To overcome problems with the convergence of the training process as well as to achieve the stable model, at the training stage the model had the form of a non-linear auto-regressive systrem with exogenous input given by

$$\hat{u}_f(k) = h(y(k), \ldots, y(k-n), u(k-1), \ldots, u(k-m)), \tag{20}$$

where the training set composed of output/input pairs $\{y(i), u(i)\}_{i=1}^{P}$ was recorded in normal operating conditions of the control system and P represents the size of the data set. Assuming that the model (20) is designed accurately and mimics the system dynamics well, after training it can be converted to the non-linear output error form (19) by replacing inputs $u(k)$ with their estimated equivalents $\hat{u}_f(k)$. It is a common way of dealing with dynamic neural network models. In order to make the model more robust, it is proposed to apply a bank of inverse models (Patan and Patan, 2022). We adapted the idea of heterogeneous ensembles. In that context, each model with a different structure was built upon the same training data. The heterogeneous approach can achieve greater diversity among sub-models, which can significantly improve ensemble performance. The output of the ensemble is calculated as the mean value of the outputs generated

by N inverse sub-models $\mathcal{I}_i$ (see Fig. 2):

$$\hat{u}_f(k) = \frac{1}{N}\sum_{i=1}^{N}\hat{u}_{fi}(k), \tag{21}$$

where $\hat{u}_{fi}(k)$ is the control signal estimate given by the i-th inverse sub-model. Each inverse sub-model of the form (20) was realized using the feed-forward neural network with one hidden layer with external memory as follows:

$$\mathcal{I}_i : \hat{u}_{fi}(k+1) = \boldsymbol{W}_{i,2}^{in}\sigma_h(\boldsymbol{W}_{i,1}^{in}\boldsymbol{\varphi}_i^{in}(k)), \tag{22}$$

where $\sigma_h(\cdot)$ is the non-linear activation function of the hidden layer, $\boldsymbol{W}_{i,1}^{in}$, $\boldsymbol{W}_{i,2}^{in}$ are the i-th inverse model weight matrices of appropriate size, and $\boldsymbol{\varphi}_i^{in}(k)$ stands for the regressor vector of the i-th sub-model defined as follows:

$$\boldsymbol{\varphi}_i^{in}(k) = [y(k), \ldots, y(k-n), u(k-1), \ldots, u(k-m), 1]^{\mathsf{T}}. \tag{23}$$

To achieve a high level of model robustness, each inverse model should have a different structure. Then, the designing process is aimed at selection of the number of hidden neurons and that of delayed outputs and inputs for each model separately.

4. Fault tolerant iterative learning control

The proposed fault tolerant control system is based on the control strategy realized using iterative learning control as portrayed in the previous section. The investigated FTC strategy is able to detect both actuator and sensor faults, estimate their size and finally to accommodate them. The main idea is to apply two kinds of models in order to distinguish between actuator and sensor faults (Fig. 2). With the help of the forward models $\mathcal{M}_i, i = 1, \ldots, M$, a fault can be detected. In turn, using the inverse models $\mathcal{I}_i, i = 1, \ldots, N$, the fault can be classified to one of the two fault classes considered.

In the nominal case, the forward model should estimate the output of the plant closely. Due to modeling errors, we cannot expect that the forward model will be a perfect replica of plant dynamics. Then, we can assume that modeling errors should be acceptable up to predefined level T_y. Analogously, the output of an inverse model should be as close to the control signal as possible. Again,

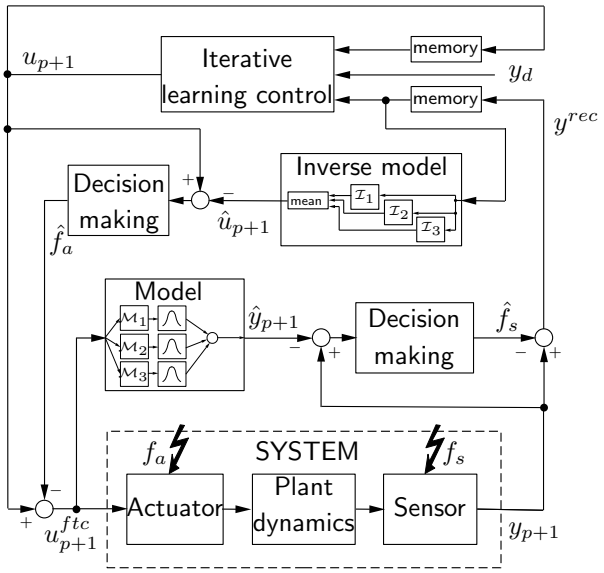

Figure 2: Proposed fault-tolerant iterative learning control.

due to modeling errors it is assumed that acceptable modeling errors are smaller that a predefined threshold T_u.

In the sequel, to make fault estimation and accommodation clear, the index p representing the trial is omitted. The routine of fault diagnosis is performed in two subsequent stages. First, each fault occurring either in an actuator or a sensor will change the value of the output signal. Then, introducing an output residual in the form

$$r_y(k) = y(k) - \hat{y}(k), \tag{24}$$

where $\hat{y}(k)$ is the output of the forward model, we can decide whether a fault occurred or not. The common approach is to apply a threshold T_y and to check if the residual $r_y(k)$ exceeds the threshold value. The second stage is to distinguish the fault class. This is carried out by means of an inverse model. Let us introduce the input residual as follows:

$$r_u(k) = u(k) - \hat{u}_f(k), \tag{25}$$

where $\hat{u}_f(k)$ is the output estimated by the inverse model. If we deal with a sensor fault, the value of the residual $r_u(k)$ is still under the threshold value T_u. Then, the sensor fault is pointed out. However, in the case of an actuator fault, the residual differs from zero significantly, and when it exceeds the threshold

value T_u the actuator fault is signalled. Details of how to perform decision making and select thresholds are provided further in this paper.

The quality of the fault detection process can be evaluated using different indexes (Bartyś et al., 2006). In this work, the following performance metrics were used:

- detection moment t_{dm}: the number of samples needed for the detection of a fault measured from the time of fault start-up to a permanent true decision about the fault;
- false detection rate r_{fd}: the ratio of the number of false alarms to the length of the trial, calculated in normal operating conditions.

4.1. Actuator fault estimation and accommodation

Substituting $u_f(k)$ with $\hat{u}_f(k)$ in (4) yields the following fault estimate:

$$\hat{f}_a(k) = \hat{u}_f(k) - u(k) = -r_u(k). \tag{26}$$

To estimate an actuator fault, one needs to develop both the inverse model of a plant as well as the forward model providing the state vector $\boldsymbol{x}(k+1)$. Moreover, a serious advantage of the procedure is that it requires modeling the system in nominal operating conditions only. This is because during the nominal work of the system $\boldsymbol{x}(k) = \boldsymbol{x}_N(k)$ and $\hat{f}_a(k) = 0$; otherwise, $\boldsymbol{x}(k) \neq \boldsymbol{x}_N(k)$ and, consequently, $\hat{f}_a(k) \neq 0$. As we obtain the fault estimate, a fault can be easily accommodated according to the formula

$$u^{ftc}(k) = u(k) - \hat{f}_a(k), \tag{27}$$

where $u^{ftc}(k)$ stands for the corrected control signal applied to the system.

4.2. Sensor fault estimation and accommodation

According to (3), the sensor fault estimate can be determined as

$$\hat{f}_s(k) = y_f(k) - \hat{y}(k) = r_y(k). \tag{28}$$

Then, using the fault estimate (28), the system output can be reconstructed via

$$y^{rec}(k) = y_f(k) - \hat{f}_s(k), \tag{29}$$

where $y^{rec}(k)$ is the reconstructed output of the system.

5. Convergence analysis

In the following, the main result of the paper is presented.

Assumption A1. Let us assume $f_a(k) = 0$, $f_s(k) = 0$ and $y_d(k)$ to be a desired reference profile defined over a discrete-time $k \in N$. It is also assumed that $y_d(k)$ is realizable, that is, there exists a unique $u_d(k)$ and an initial state $\boldsymbol{x}_d(0)$, i.e.,

$$\begin{aligned} \boldsymbol{x}_d(k+1) &= g(\boldsymbol{x}_d(k), u_d(k)), \\ y_d(k) &= \boldsymbol{C}^{\mathsf{T}} \boldsymbol{x}_d(k). \end{aligned} \tag{30}$$

Assumption A2. Let $\forall k \in N$, $\forall p$ the actuator and sensor faults satisfy the following:

$$\|f_a(k)\| \leq \epsilon_a, \quad \|f_s(k)\| \leq \epsilon_s,$$

where $\epsilon_a \geq 0$, $\epsilon_s \geq 0$ are finite bounds. Moreover, $\forall p$ the initial system state error satisfies

$$\|\Delta \boldsymbol{x}_p(0)\| \leq \epsilon_x, \tag{31}$$

where $\Delta \boldsymbol{x}_p(k) = \boldsymbol{x}_d(k) - \boldsymbol{x}_p(k)$ and $\epsilon_x \geq 0$ is a positive constant.

Assumption A3. A non-linear function g satisfies the global Lipschitz condition

$$\|g(\boldsymbol{x}_1, u_1) - g(\boldsymbol{x}_2, u_2)\| \leq L \left(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| + |u_1 - u_2| \right), \tag{32}$$

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

Definition 1. The partial derivatives of the controller activation function σ_o (6) with respect to the inputs are defined as

$$f_u(k) = w_{3,p}^c \frac{\partial \sigma_o(\cdot)}{\partial u_p(k)}, \quad f_e(k) = w_{3,p}^c \frac{\partial \sigma_o(\cdot)}{\partial e_p(k)}. \tag{33}$$

Definition 2. The λ-norm of a vector $z(k)$ is defined as follows:

$$\|z(k)\|_\lambda = \sup_{k \in N} \beta^{-\lambda k} \|z(k)\|, \tag{34}$$

where $\lambda > 0$ is a finite constant and $\beta > 1$.

Theorem 1. *Consider the non-linear system* (1) *satisfying the assumptions (A1)–(A3) and the reference trajectory* $y_d(k)$ *satisfying the assumption (A1). Applying the control law* (6)–(7) *satisfying the condition*

$$\left| \sup_k \|f_u(k)\| + \sup_k \|f_e(k) \boldsymbol{C}^{\mathsf{T}}\| \left(\frac{1 - L^{-(\lambda-1)(k+2)}}{1 - L^{-(\lambda-1)}} - 1 \right) \right| < 1 \tag{35}$$

guarantees that the tracking error remains bounded, i.e.,

$$\lim_{p\to\infty}\|y_d(k)-y_p(k)\|_\lambda \le \sigma, \tag{36}$$

where constant $\sigma > 0$ is dependent on ϵ_x, ϵ_a, ϵ_s.

PROOF. Expanding the control law (6) into the Taylor series yields

$$\Delta u_{p+1}(k) \simeq f_u(k)\Delta u_p(k) - f_e(k)\Delta y_p(k), \tag{37}$$

where $\Delta u_p(k) = u_d(k) - u_p(k)$ and $\Delta y_p(k) = e_p(k)$. According to the assumption (A1), we obtain

$$\begin{aligned}\Delta \boldsymbol{x}_p(k+1) &= g(\boldsymbol{x}_d(k), u_d(k)) - g(\boldsymbol{x}_p(k), u_p(k) + f_a(k)),\\ \Delta y_p(k) &= \boldsymbol{C}^{\mathsf{T}}\Delta \boldsymbol{x}_p(k) + f_s(k),\end{aligned} \tag{38}$$

where $\Delta \boldsymbol{x}_p(k) = \boldsymbol{x}_d(k) - \boldsymbol{x}_p(k)$. Next, taking the norm of both sides of (37), we obtain

$$\begin{aligned}\|\Delta u_{p+1}(k)\| \le& \|f_u(k)\Delta u_p(k)\| + \|f_e(k)\Delta y_p(k)\|\\ \le& \|f_u(k)\Delta u_p(k)\| + \|f_e(k)\boldsymbol{C}^{\mathsf{T}}\Delta x_p(k)\| + \|f_e(k)\boldsymbol{C}^{\mathsf{T}} f_s(k)\|\\ \le& \|f_u(k)\Delta u_p(k)\| + \|f_e(k)\boldsymbol{C}^{\mathsf{T}}\|(\|\Delta x_p(k)\| + \epsilon_s).\end{aligned} \tag{39}$$

Now, taking the norm of the both sides of the state equation in (38) and applying the assumption (A3), we have

$$\|\Delta \boldsymbol{x}_p(k+1)\| \le L\|\Delta \boldsymbol{x}_p(k)\| + L|\Delta u_p(k) - f_a(k)|. \tag{40}$$

Assuming that (A2) holds and applying the recursive form of (40), the following representation is achieved for $k = 1, \ldots, N-1$:

$$\|\Delta \boldsymbol{x}_p(k)\| \le L^{k+1}\|\Delta \boldsymbol{x}_p(0)\| + \sum_{i=0}^{k-1} L^{k-i}|\Delta u_p(i) - f_a(i)|. \tag{41}$$

Let us introduce the auxiliary sequence $\tilde{u}_p(k)$ excluding the initial zero element from the sequence $\Delta u_p(k)$:

$$\tilde{u}_p(k) = \begin{cases} 0, & k = 0,\\ \Delta u_p(k-1) - f_a(k-1), & k = 1, \ldots, N. \end{cases} \tag{42}$$

Then, using (42), the equation (41) can be rewritten as follows:

$$\|\Delta \boldsymbol{x}_p(k)\| \le L^{k+1}\|\Delta \boldsymbol{x}_p(0)\| + \sum_{i=0}^{k} L^{k-i+1}|\tilde{u}_p(i)|. \tag{43}$$

Now, substituting (43) into (39) for any $k \in \mathbb{N}_{<N}$ yields

$$\|\Delta u_{p+1}(k)\| \leq \|f_u(k)\|\|\Delta u_p(k)\| + \|f_e(k)\boldsymbol{C}^{\mathsf{T}}\| \left(\sum_{i=0}^{k} L^{k-i+1} |\tilde{u}_p(i)| + L^{k+1}\epsilon_x + \epsilon_s \right). \tag{44}$$

Applying λ-norm to both sides of (44) gives

$$\begin{aligned} \|\Delta u_{p+1}(k)\|_\lambda \leq & \sup_k \|f_u(k)\| \|\Delta u_p(k)\|_\lambda + \sup_k \|f_e(k)\boldsymbol{C}^{\mathsf{T}}\| (L^{k+1}\epsilon_x + \epsilon_s) \\ & + \sup_k \|f_e(k)\boldsymbol{C}^{\mathsf{T}}\| \sup_k \beta^{-\lambda k} \sum_{i=0}^{k} L^{k-i+1} |\tilde{u}_p(i)|. \end{aligned} \tag{45}$$

Setting $L = \beta$, one can rewrite the last supremum in (45) as follows:

$$\begin{aligned} & \sup_k \beta^{-\lambda k} \sum_{i=0}^{k} \beta^{k-i+1} |\tilde{u}_p(i)| = \sup_k \sum_{i=0}^{k} \beta^{-\lambda k + k - i + 1} |\tilde{u}_p(i)| \\ & = \sup_k \sum_{i=0}^{k} \beta^{-\lambda(i-1)} |\tilde{u}_p(i)| \beta^{-(\lambda-1)(k-i+1)} \\ & \leq \sup_k \sum_{i=0}^{k} \sup_k (\beta^{-\lambda k} |\tilde{u}_p(k+1)|) \beta^{-(\lambda-1)(k-i+1)} \\ & = \sup_k \beta^{-\lambda k} |\tilde{u}_p(k+1)| \cdot \sup_k \sum_{i=0}^{k} \beta^{-(\lambda-1)(k-i+1)} \\ & = \|\Delta u_p(k) - f_a(k)\|_\lambda \left(\frac{1 - \beta^{-(\lambda-1)(k+2)}}{1 - \beta^{-(\lambda-1)}} - 1 \right) \\ & \leq \|\Delta u_p(k)\|_\lambda \left(\frac{1 - \beta^{-(\lambda-1)(k+2)}}{1 - \beta^{-(\lambda-1)}} - 1 \right) + \epsilon_a \left(\frac{1 - \beta^{-(\lambda-1)(k+2)}}{1 - \beta^{-(\lambda-1)}} - 1 \right). \end{aligned} \tag{46}$$

Introducing

$$\gamma_1 = \sup_k \|f_u(k)\|, \tag{47}$$

$$\gamma_2 = \sup_k \|f_e(k)\boldsymbol{C}^{\mathsf{T}}\| \left(\frac{1 - \beta^{-(\lambda-1)(k+2)}}{1 - \beta^{-(\lambda-1)}} - 1 \right), \tag{48}$$

and

$$\gamma_3 = \sup_k \|f_e(k)\boldsymbol{C}^{\mathsf{T}}\| \left(\beta^{k+1}\epsilon_x + \left(\frac{1 - \beta^{-(\lambda-1)(k+2)}}{1 - \beta^{-(\lambda-1)}} - 1 \right) \epsilon_a + \epsilon_s \right), \tag{49}$$

(45) leads to

$$\|\Delta u_{p+1}(k)\|_\lambda \leq (\gamma_1 + \gamma_2)\|\Delta u_p(k)\|_\lambda + \gamma_3. \tag{50}$$

Let us rearrange (50) as

$$\|\Delta u_{p+1}(k)\|_\lambda - (\gamma_1 + \gamma_2)\|\Delta u_p(k)\|_\lambda \leq \gamma_3. \tag{51}$$

It is obvious (Chien, 1998) that (51) is satisfied if $|\gamma_1 + \gamma_2| < 1$. Then, we can conclude that

$$\lim_{p\to\infty} \|\Delta u_p(k)\|_\lambda \leq \frac{\gamma_3}{1 - |\gamma_1 + \gamma_2|}. \tag{52}$$

Finally, tracking error convergence of the proposed FTC-ILC can be proven. Again, let us apply λ-norm to (41) and carry out the same reasoning as in (45). As a result, we obtain

$$\|\Delta \boldsymbol{x}_p(k)\|_\lambda \leq \gamma_4 \|\Delta u_p(k)\|_\lambda + \gamma_5, \tag{53}$$

with $\gamma_4 = \frac{1-\beta^{-(\lambda-1)(k+2)}}{1-\beta^{-(\lambda-1)}} - 1$ and $\gamma_5 = \sup_k \left(\beta^{k+1}\epsilon_x + \left(\frac{1-\beta^{-(\lambda-1)(k+2)}}{1-\beta^{-(\lambda-1)}} - 1\right)\epsilon_a\right)$. As the trial number tends to infinity, we have

$$\lim_{p\to\infty} \|\Delta x_p(k)\|_\lambda \leq \gamma_4 \frac{\gamma_3}{1 - |\gamma_1 + \gamma_2|} + \gamma_5. \tag{54}$$

Using the output equation of the system (1),

$$\begin{aligned}
\lim_{p\to\infty} \|y_d(k) - y_p(k)\|_\lambda \leq & \|\boldsymbol{C}^\mathsf{T}\| \|\Delta x_p(k)\|_\lambda + \epsilon_s \\
\leq & \|\boldsymbol{C}^\mathsf{T}\| \left(\gamma_4 \frac{\gamma_3}{1 - |\gamma_1 + \gamma_2|} + \gamma_5\right) + \epsilon_s \\
\triangleq & \sigma(\epsilon_x, \epsilon_a, \epsilon_s).
\end{aligned} \tag{55}$$

Remark 1. From a practical perspective, a key point is the method of keeping the condition (35) satisfied. Analyzing (48), it is clear that $\lim_{\lambda\to\infty} \gamma_2 = 0$. Then, a crucial factor for satisfying convergence condition is the component γ_1. According to Definition 1, by using (7), (47) can be rewritten as

$$\gamma_1 = |w_{3,p}^c| \sup_k \left\| \frac{\partial \sigma_o(\cdot)}{\partial u_p(k)} \right\|. \tag{56}$$

Clearly, after controller weight update, one can verify the convergence criterion very easily by proper adaptation of the weight $w_{3,p}^c$.

Remark 2. Assuming that $\sigma_o(\cdot)$ is a linear function, the supremum in (56) can be further transformed as follows:

$$\sup_k \left\| \frac{\partial \sigma_o(\cdot)}{\partial u_p(k)} \right\| = \sup_k \|(\boldsymbol{W}_{1u,p}^c \circ \sigma_h^{'})^\mathsf{T} \boldsymbol{W}_{2,p}^c\|, \tag{57}$$

where $\boldsymbol{W}_{1u,p}^c$ are the input weights assigned to the controller input $u_p(k)$. In order to achieve good approximation abilities, usually $\sigma_h(\cdot)$ is selected to be a squashing function (a sigmoid or a hyperbolic tangent one). In such cases, the

maximum of the activation function derivative is equal to 1 and (57) can be rewritten as

$$\sup_k \left\| \frac{\partial \sigma_o(\cdot)}{\partial u_p(k)} \right\| = |(\boldsymbol{W}^c_{1u,p})^{\mathsf{T}} \boldsymbol{W}^c_{2,p}|. \tag{58}$$

Substituting (58) to (56), one obtains

$$\gamma_1 = |w_{3,p}(\boldsymbol{W}^c_{1u,p})^{\mathsf{T}} \boldsymbol{W}^c_{2,p}|. \tag{59}$$

6. Experimental study

6.1. System description

The proposed fault-tolerant control system was tested on the example of tracking control for the magnetic brake system (Patan and Patan, 2021). It constitutes a non-friction system dissipating high kinetic energy into thermal energy. According to the amount of heat, in practical settings the magnetic brake is usually just a component of more sophisticated systems with additional electric actuation and energy recovery, e.g., electromagnetic brakes encountered widely in high speed railways, big trucks and industrial elevators. For clarity of presentation and proper illustration of the potential of the developed approach, a relatively simplistic version of the magnetic brake is considered here. Its realization consists in rotating the conductive element immersed in a magnetic field generated by external magnets (cf. Fig. 3(a)). The movement of the conductor within the magnetic field results in inducing eddy currents, which further interact with the magnetic flux producing a spatio-temporal field of Lorentz forces. These, in turn, generate a total braking torque slowing down

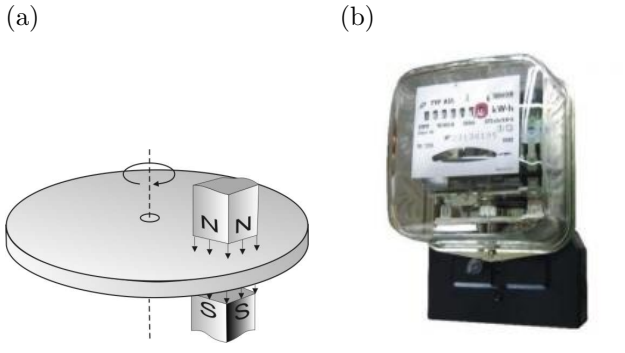

Figure 3: Schematic view of the magnetic brake (a) and the example of its practical realization: Inventor A52 energy meter (b).

the rotating element. The rotational velocity over time horizon $t \in (0, t_f]$ can be mathematically described by the initial value problem

$$J\frac{\mathrm{d}\omega(t)}{\mathrm{d}t} = M(B_0(t), \omega(t)), \quad \omega(0) = \omega_0, \tag{60}$$

where ω is rotational velocity, J is the moment of inertia of the rotating element, $B_0(t)$ is the magnetic flux generated by the external magnets, and M is the total aggregated braking torque,

$$M = \int_{\mathcal{D}} T_q(B(x,t), B_0(t), \omega(t))\,\mathrm{d}V, \tag{61}$$

where $\mathcal{D} \in \mathrm{R}^3$ is a spatial domain representing the rotating conductor, $\mathrm{d}V$ stands for the volume element of $\mathcal{D}$, T_q is the Lorentz force on the volume element related to the spatial location x, and $B(x,t)$ denotes the spatio-temporal field of the internal magnetic flux inside the conductor.

It is clear that the total braking torque (61) at any time instant depends in non-linear manner on the spatial field of eddy currents generated by the magnetic flux and shape of the domain $\mathcal{D}$ as well as the external magnetic flux. Accurate estimation of the braking torque is a key problem for both the design of the control system and fault accommodation. In practical conditions, modeling uncertainty in each repeated trial of the braking process will play a significant role and should be properly estimated based on the observed data. The torque field T_q is continuous and bounded provided by the physics of the process, so the functional (61) is smooth enough to satisfy the assumption A3. Therefore, in order to determine the system response up to satisfactory accuracy, a reasonable approach is to make use of the well-known ensemble averaging properties of neural networks. Apart from accurate modeling of the non-linear torque, this will also provide necessary robustness with respect to disturbances. Then, neural models are further incorporated into the proposed iterative learning control scheme.

For the purpose of the experiment, as a simulated reference we used an aluminum disk with a diameter of 10 cm and thickness of 1 mm rotating in the external magnetic field with a maximum flux of 0.1 T. This is close to a typical configuration encountered in analog energy meters, cf. Fig. 3(b)). The initial angular velocity ω_0 was set to a value of 10 rpm. The task was to slow down the disk to 0.01% of the initial velocity within the period of 10 seconds. The

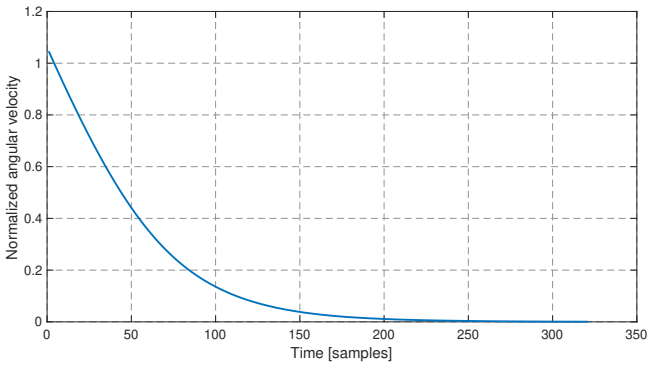

Figure 4: Reference signal.

simulated reference profile being the solution to (60) for the sample time 0.03 s is presented in Fig. 4 (the plot is normalized to rad/s). The objective of the control task is to accurately follow the reference profile of the output angular velocity $\omega(t)$ by proper update of the input signal of the magnetic flux $B_0(t)$ generated by an external electromagnet, simultaneously taking into account potential faults occurring in the system.

To investigate fault tolerant properties of the proposed approach, a number of faulty scenarios were prepared, including both sensor and actuator faults, as well as multiple faults. Specification of the faults is included in Table 1. All faults are of abrupt and additive nature.

Table 1: Faulty scenarios specification.

scenario	type	intensity	trial	time instant
f_1	sensor	$+0.05$	5th	100th
f_2	sensor	-0.05	10th	100th
f_3	actuator	$+0.002$	10th	95th
f_4	sensor then actuator	$+0.05/+0.002$	10th	100th/200th

6.2. Forward and inverse model design

Forward multi-model. In this study we applied a multi-model developed in our previous work (Patan et al., 2020), and the details can be found therein. Briefly, to design the model, seven operating points were considered (from the interval $[0.003 - 0.095]$) and, consequently, seven state-space neural models (17) were

derived. The final output of the forward model was calculated using the membership functions (18) of the spread guaranteeing covering the entire range of the operating points and at the same time satisfying the criterion (14). Each sub-model had a relatively simple architecture with five hidden hyperbolic tangent neurons and a low model order (2nd or 3rd).

Inverse model ensemble. The ensemble of inverse models used in this research was derived in the previous work of the authors (Patan and Patan, 2022). To achieve a robust model, a mixture of five neural inverse neural networks of the form (22) was used. Each model had a different number of hidden non-linear neurons (varying from 3 to 12) and a different number of past inputs m and outputs n (3 or 4 in both cases). The final output of the inverse model was calculated as the mean value of sub-models' outputs according to (21). Details concerning the design process as well as training are presented by Patan and Patan (2022).

6.3. Learning controller synthesis

The initial parameters of the learning controller were set as small random values. After that the controller was applied to control the magnetic brake and at the same time trained with the algorithm described by the equations (10)–(13). The number of neurons in the hidden layer was set to $n_h = 15$, and the activation function was a hyperbolic tangent. The output activation function σ_o was a linear one. To obtain proper generalization abilities of the controller, the learning rate was set to be exponentially decreasing:

$$\eta(i) = \eta_0 e^{-0.05i}, \tag{62}$$

where i stands for the training iteration. The maximum number of training iterations was equal to 1000, and the initial value of the training rate at each trial was $\eta_0 = 0.001$. The convergence of the root mean square of the tracking error is shown in Fig. 5. It is clear that the tracking error norm was established at the value of about 0.01, which is, in fact, the value of the norm of the noise affecting the system output. Then, further reduction of the tracking error norm is not possible. After training the controller, parameters are stored for further use. According to Remark 1, a crucial issue for the convergence is to keep

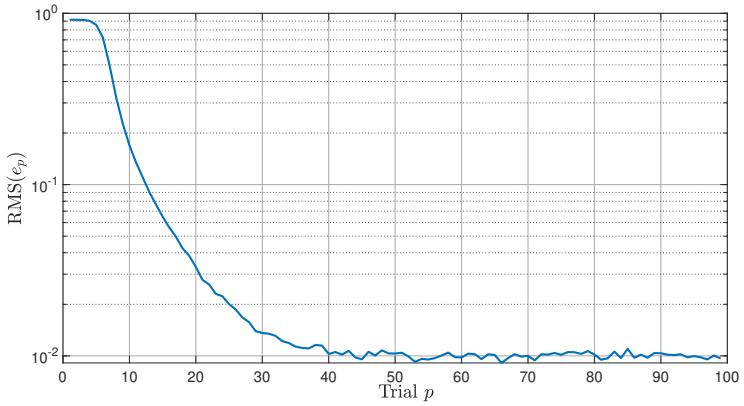

Figure 5: Convergence of the ILC: normal operating conditions.

parameter γ_1 below the value of 1. From Fig. 6 it is clear that γ_1 has the correct value all the time, guaranteeing the convergence of the control system. An important characteristic of the control system is its self fault tolerance. In

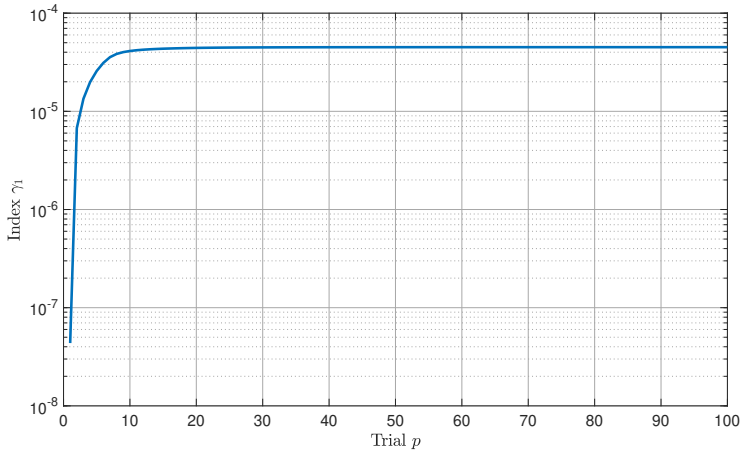

Figure 6: Evolution of the parameter γ_1.

this framework, two experiments were conducted and investigated. The first one was the reaction of the learning controller to the faulty scenario f_1. The convergence of the tracking error for this case is presented in Fig. 7 with the blue-solid line. Obviously, at the 5th trial there is an abrupt change in the RMS value of the tracking error after which it settles at the level nearing 0.05. Clearly, ILC itself is not able to compensate the sensor fault, but the RMS of the tracking error is upper bounded. The boundedness of the tracking error can

be explained directly from the results provided by Theorem 1. Assuming a zero initial state ($\epsilon_x = 0$) and no actuator fault ($\epsilon_a = 0$), parameter γ_5 is equal to zero. Moreover, one can find such a value of λ that the parameter γ_4 is very close to zero. Then, (55) can be rewritten as

$$\lim_{p\to\infty} \|y_d(k) - y_p(k)\|_\lambda \leq \epsilon_s + \psi, \tag{63}$$

where ψ is a very small value close to zero. Fault intensity was equal to 0.05. In the case considered, $\epsilon_s = 0.8958$, but its RMS is $\sqrt{\frac{1}{321}\epsilon_s^2} = 0.05$, which is a value very close to the upper bound of the tracking error observed in Fig. 7. Summarizing this part of the research, ILC is not able to compensate the sensor fault and an additional fault accommodation mechanism is required to achieve fault-tolerant control.

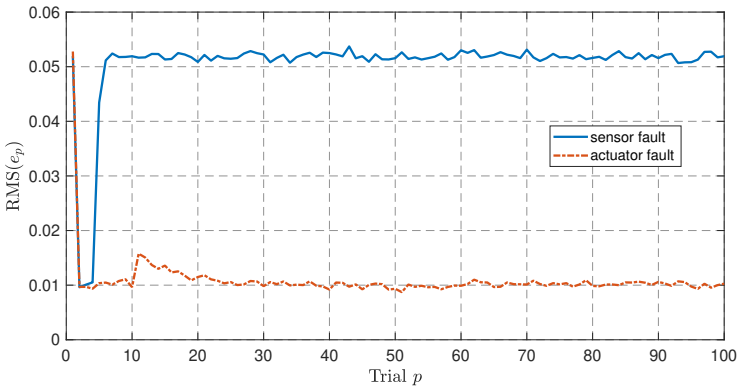

Figure 7: Convergence of ILC in the case of faults.

The second experiment was to analyze the behavior of the learning controller in the case of the actuator fault f_3. The convergence of the tracking error is shown in Fig. 7 with the red-dashed line. After fault occurrence (10th trial), the convergence of the tracking error is violated and the RMS of the tracking error starts to increase. However, very quickly the learning controller was able to properly react and retrain its parameters to yet again converge to a minimum value of the tracking error after approximately 15 trials. To explain this phenomenon, let us analyze again the condition (55). Assuming a zero initial state ($\epsilon_x = 0$) and no sensor fault ($\epsilon_s = 0$), one can find such a value of λ that the parameters γ_4 and γ_5 are very close to zero. Then, (55) can be rewritten as

$$\lim_{p\to\infty} \|y_d(k) - y_p(k)\|_\lambda \leq \psi, \tag{64}$$

where ψ is a very small value close to zero. To conclude, the proposed ILC has the actuator fault tolerance property. However, we believe that developing actuator fault accommodation will help to improve the convergence rate of the tracking error after fault occurrence.

6.4. Fault detection and isolation

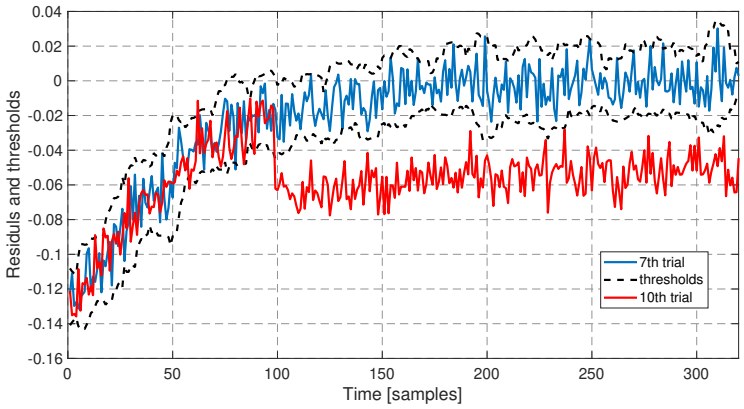

Figure 8: Output residual thresholding.

The fault diagnosis system is based on the evaluation of two residual signals as was preliminary portrayed in Section 4. A fault is detected using the residual (24). In the case considered, during each trial we can observe transient behavior in magnetic brake work, so the residual at the beginning has some value which vanishes as time passes (see Fig. 8, with the residual marked with the blue line). Then, to make a decision about faults, we decided to apply an adaptive threshold. It can be determined on the basis of the statistical parameters of the residual derived in normal operation conditions. Due to the transient behavior observed in the residuals, a moving mean and moving standard deviation were calculated over the past $n \in (0, k)$ samples using the formulas (Patan, 2008)

$$\bar{r}_y(k) = \frac{1}{n} \sum_{i=k-n}^{k} r_y(i), \tag{65}$$

where $\bar{r}_y$ is the moving mean of the residual r_y at the time instant k, n is the window length, and

$$s_{r_y}(k) = \sqrt{\frac{1}{n-1} \sum_{i=k-n}^{k} (r_y(i) - \bar{r}_y(k))}, \tag{66}$$

where $s_{r_y}(k)$ is the standard deviation of the residual at the time instant k. Using (65) and (66), upper and lower thresholds can be easy calculated using $2s_{r_y}$ rule-of-thumb as follows:

$$T_{u,l}^{r_y} = \bar{r}_y(k) \pm 2s_{r_y}, \tag{67}$$

where $T_u^{r_y}$ and $T_l^{r_y}$ represent the upper and lower thresholds, respectively. In the present work, the window length was set to $n = 10$ and vectors $\bar{r}_y$ and s_{r_y} were calculated based on residuals derived for normal operating conditions. After that, these vectors were stored for further use. The derived thresholds are depicted in Fig. 8 (black-dashed lines). Such derived thresholds can be effectively used for fast and reliable fault detection. In Fig. 8, one can observe the residual derived for the 10th cycle of magnetic brake operation (red line). An alarm is raised at the 100th time instant. This is consistent with the scenario f_2, according to which the sensor fault is introduced during the 10th trial at the 100th time step. In this case, the false detection rate $r_{fd} = 3.69\%$ and the detection moment $t_{dm} = 1$ time instant. Clearly, fault detection was carried out quickly and reliably.

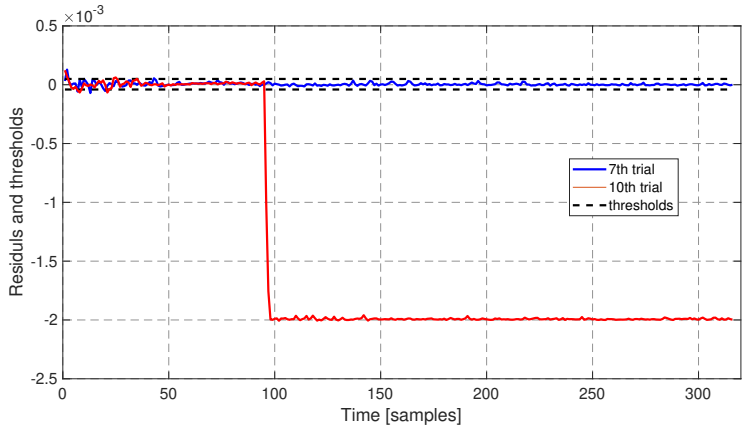

Figure 9: Input residual thresholding.

However, to definitely point out that a fault is a sensor one, another residual (25) should be processed. This time the residual is centered (see Fig. 9, with the residual marked with blue line), and for decision making simple thresholds can be used. To calculate the thresholds the mean value and standard deviation of the residual $r_u(k)$ determined for normal operating conditions are required.

In the case considered, we achieved the following values: $\bar{r}_u = 8.78 \cdot 10^{-6}$ and $s_{r_u} = 3.43 \cdot 10^{-5}$. Finally, the thresholds are calculated using the $3s_{r_y}$ rule:

$$T_{u,l}^{r_u} = \bar{r}_u \pm 3s_{r_u}, \tag{68}$$

where $T_u^{r_u}$ and $T_l^{r_u}$ represent the upper and lower thresholds, respectively. In the study considered, $T_u^{r_u} = 1.12 \cdot 10^{-4}$ and $T_l^{r_u} = -0.94 \cdot 10^{-4}$. Threshold values are stored and used in the fault diagnosis process. In Fig. 9, thresholds are marked with the black-dashed lines. In normal operating conditions the residual is inside the region limited by thresholds, but when an actuator fault occurs the residual almost immediately exceeds the thresholds, as illustrated in Fig. 9 (the residual marked with the red line). In this case, an alarm is raised at the 95th time instant. This is consistent with the scenario f_3, according to which the actuator fault is introduced during the 10th trial at the 95th time step. In this case, the false detection rate $r_{fd} = 1.25\%$ and the detection moment $t_{dm} = 1$ time instant, which means that the fault was quickly and surely detected.

6.5. Fault estimation and accommodation

After fault diagnosis, a fault can be estimated and accommodated. First, let us analyze the scenario f_2. The results of fault tolerant control are presented in Fig. 10. Clearly, the sensor fault was reliably accommodated in one trial using the sensor fault estimate (28). This was possible due to fast fault detection, as illustrated in Fig. 8. In the case considered, the mean value of $\hat{f}_s(k)$ was -0.0506, which means that sensor fault was accurately estimated with the absolute value of the reconstruction error $\| - 0.0506 - (-0.05)\| = 0.0006$.

The second experiment was to check fault-tolerant control quality in the case of the actuator fault f_3. The control results are shown in Fig. 10. Once again, the fault was reliably accommodated using the actuator fault estimate (26). It should be stressed that the proposed learning controller has actuator fault tolerance properties as illustrated in Fig. 7 (red-dashed line). However, by introducing the actuator fault accommodation mechanism (27), control convergence was significantly improved. In this case, the mean value of $\hat{f}_a(k)$ was 0.0022, which is very close value to the real fault intensity equal to 0.002 (see Table 1). The absolute value of the reconstruction error is $\|0.0022 - 0.002)\| = 0.0002$.

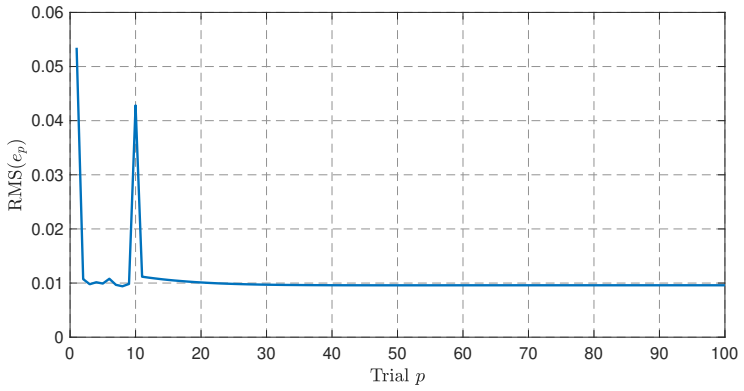

Figure 10: Sensor fault accommodation: scenario f_2.

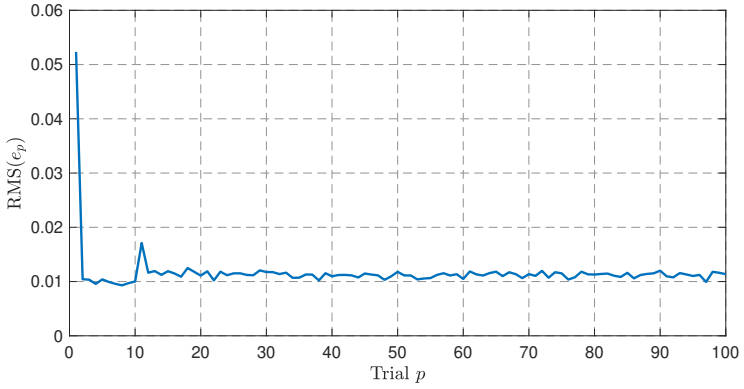

Figure 11: Actuator fault accommodation: scenario f_3.

The last investigated scenario, f_4, consists in accommodation of two faults appearing one by one. The control results are presented in Fig. 12. Fault accommodation is pretty effective. In Fig. 13 we can observe the process of fault detection and isolation. Firstly, the sensor fault was signalled at the 100th time step as the residual r_y permanently exceeded the upper threshold $T_u^{r_y}$. At the same time, the residual r_u remains inside the region bounded by thresholds. After that, at the 200th time step, the actuator fault was signalled as the residual r_u exceeded the lower threshold $T_l^{r_u}$. Simultaneously, the residual r_y still retained outside the region bounded by thresholds.

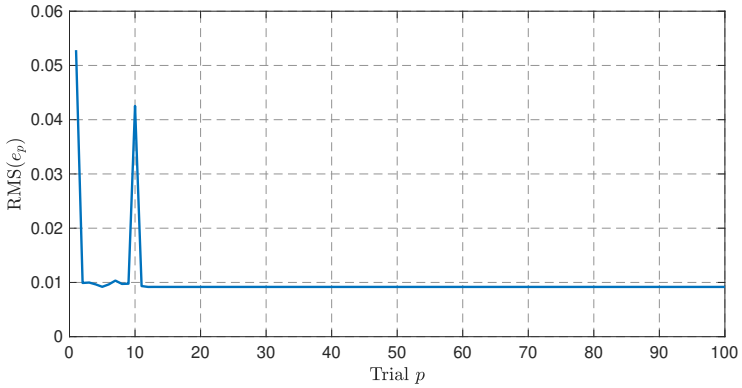

Figure 12: Multiple faults accommodation: scenario f_4.

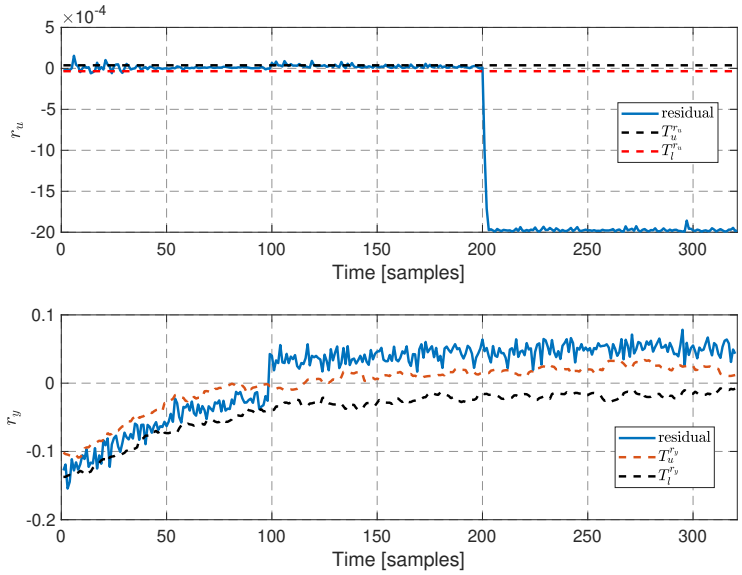

Figure 13: Multiple faults detection.

7. Conclusions

The paper proposed fault-tolerant iterative learning control carried out by means of neural network models. To design this control scheme, three kinds of neural networks were applied. Due to the class of systems considered, a forward model was built upon a mixture of state-space neural network submodels. Then, the forward model was used to carry out fault detection. The second class of neural networks applied included a neural network with external dynamics, used to build an ensemble of inverse models. The ensemble was used in a twofold way: to perform fault isolation and to generate the desired control estimate for the purpose of controller synthesis. The third class of neural networks included a feed-forward network used to design the static learning controller. The proposed fusion of neural networks is very useful in the cases when the mathematical model of a system could not be derived and data-driven methods are the only alternative. Moreover, the application of neural networks renders it possible to design a learning controller of the adaptive kind. The paper provides sufficient conditions guaranteeing the convergence of the proposed control strategy in the presence of both actuator and sensor faults. An important result from the convergence analysis is that the proposed open-loop control has the inherent tolerance regarding actuator faults. Anyway, this work proposed fault estimation and accommodation mechanisms to improve control quality deteriorated by faults. Simulation results confirm the effectiveness of the proposed fault-tolerant control. Summarizing, the decided advantages of the proposed approach in comparison to existing contributions are as follows:

- very extensive treatment of the subject including different types of faults, model uncertainty and control design supplemented with careful mathematical characterization and convergence analysis;
- there is no linearization required at any stage of control design, i.e., modeling, fault detection and accommodation, and controller synthesis, which broaden the area of applicability.

There still remain open problems, which require more attention. Each inverse model candidate was trained using the so-called series-parallel setting, which means that during training the neural model was treated as a feed-forward

one. More accurate modeling is possible using recursive training via a parallel identification model. Secondly, a very important issue is to extend the proposed fault accommodation mechanism onto the class of incipient faults. Finally, a closer look into robustness properties of the proposed control scheme to external disturbances as well as to a model mismatch is needed.

Acknowledgements

This research was funded in part by National Science Centre in Poland, grant No. 2020/39/B/ST7/01487. For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

Ahn, H.S., Moore, K.L., Chen, Y., 2007. Iterative learning Control. Robustness and Monotonic Convergence for Interval Systems. Communications and Control Engineering, Springer-Verlag, London.

Arimoto, S., Kawamura, S., Miyazaki, F., 1984. Bettering operation of robots by learning. Journal of Robotic systems 1, 123–140.

Bartyś, M., Patton, R., Syfert, M., de las Heras, S., Quevedo, J., 2006. Introduction to the DAMADICS actuator FDI benchmark study. Control Engineering Practice 14, 577–596. doi:10.1016/j.conengprac.2005.06.015.

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., 2016. Diagnosis and Fault-Tolerant Control. Springer-Verlag, New York.

Bristow, D.A., Tharayil, M., Alleyne, A.G., 2006. A survey of iterative learning control: A learning-based method for high-performance tracking control. IEEE Control Systems Magazine 26, 96–114.

Bu, X., Hou, Z., Yu, Q., Yang, Y., 2020. Quantized data driven iterative learning control for a class of nonlinear systems with sensor saturation. IEEE Transactions on Systems, Man, and Cybernetics: Systems 50, 5119–5129. doi:10.1109/TSMC.2018.2866909.

Bu, X., Xu, F., Hou, Z., Yang, H., 2012. Robust iterative learning control for nonlinear systems with measurement disturbances. Journal of Systems Engineering and Electronics 23, 906–913.

Chen, Y., Chu, B., Freeman, C.T., 2022. Iterative learning control for path-following tasks with performance optimization. IEEE Transactions on Control Systems Technology 30, 234–246. doi:10.1109/TCST.2021.3062223.

Chi, R., Hou, Z., 2009. A new neural network-based adaptive ILC for nonlinear discrete-time systems with dead zone scheme. Journal of Systems Science and Complexity 22, 435–445.

Chien, C.J., 1998. A discrete iterative learning control for a class of nonlinear time-varying systems. IEEE Transactions on Automatic Control 43, 748–752.

Chow, T.W.S., Li, X.D., Fang, Y., 2000. A real-time learning control approach for nonlinear continuous-time system using recurrent neural networks. IEEE Transactions on Industrial Electronics 47, 478–486.

Conchas, R.F., Sanchez, E.N., Ricalde, L.J., Alvarez, J.G., Alanis, A.Y., 2023. Sensor fault-tolerant control for a doubly fed induction generator in a smart grid. Engineering Applications of Artificial Intelligence 117, 105527. doi:10.1016/j.engappai.2022.105527.

Ducard, G.J.J., 2009. Fault-tolerant Flight Control and Guidance Systems: Practical Methods for Small Unmanned Aerial Vehicles. Advances in Industrial Control, Springer, London.

Freeman, C., Rogers, E., Burridge, J., Hughes, A.M., Meadmore, K., 2015. Iterative Learning Control for Electrical Stimulation and Stroke Rehabilitation. SpringerBriefs in Control, Automation and Robotics, Springer-Verlag, London.

Haykin, S., 2009. Neural Networks and Learninig Systems. Prentice-Hall, New Jersey.

Huang, J., Wang, W., Su, X., 2021. Adaptive iterative learning control of multiple autonomous vehicles with a time-varying reference under actuator

faults. IEEE Transactions on Neural Networks and Learning Systems 32, 5512–5525. doi:10.1109/TNNLS.2021.3069209.

Leith, D.J., Leithead, W.E., 2000. Survey of gain-scheduling analysis and design. International Journal of Control 73, 1001–1025.

Li, L., Lina, Y., Hong, W., Zhiwei, G., 2022. Iterative learning fault diagnosis and fault tolerant control for stochastic repetitive systems with brownian motion. ISA Transactions 121, 171–179. doi:10.1016/j.isatra.2021.03.030.

Liu, G., Hou, Z., 2022. Adaptive iterative learning fault-tolerant control for state constrained nonlinear systems with randomly varying iteration lengths. IEEE Transactions on Neural Networks and Learning Systems doi:10.1109/TNNLS.2022.3185080. early access.

Miao, Y., Li, C., 2017. Robust adaptive iterative learning control for discrete-time nonlinear systems with time-iteration-varying parameters. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47, 1737–1745.

Moore, K.L., 1993. Iterative Learning Control for Determinisic Systems. Advances in Industrial Control, Springer-Verlag, London.

Nørgaard, M., Ravn, O., Poulsen, N., Hansen, L., 2000. Networks for Modelling and Control of Dynamic Systems. Springer-Verlag, London.

Noura, H., Theilliol, D., Ponsart, J., Chamseddine, A., 2003. Fault-Tolerant Control Systems: Design and Practical Applications. Springer-Verlag, Berlin.

Oomen, T., Rojas, C.R., 2017. Sparse iterative learning control with application to a wafer stage: Achieving performance, resource efficiency, and task flexibility. Mechatronics 47, 134–147.

Owens, D.H., 2016. Iterative Learning Control. An Optimization Paradigm. Advances in Industrial Control, Springer-Verlag, London.

Patan, K., 2008. Artificial neural networks for the modelling and fault diagnosis of technical processes. volume 377 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, Berlin-Heidelberg.

Patan, K., 2019. Robust and Fault-Tolerant Control. Neural-Network-Based Solutions. Springer-Nature, Cham, Switzerland.

Patan, K., Patan, M., 2020. Neural-network-based iterative learning control of nonlinear systems. ISA Transactions 98, 445–453.

Patan, K., Patan, M., 2021. Neural-network-based nonlinear iterative learning control: Magnetic brake study, in: 2021 IJCNN (IJCNN), pp. 1–7. doi:10.1109/IJCNN52387.2021.9533709.

Patan, K., Patan, M., 2022. Actuator fault-tolerant iterative learning control of the magnetic brake system. IFAC-PapersOnLine 55, 266–271. doi:10.1016/j.ifacol.2022.07.140.

Patan, K., Patan, M., Klimkowicz, K., 2020. Sensor fault-tolerant control design for magnetic brake system. Sensors 20, 1–18.

Rouabah, B., Toubakh, H., Kafi, M.R., Sayed-Mouchaweh, M., 2022. Adaptive data-driven fault-tolerant control strategy for optimal power extraction in presence of broken rotor bars in wind turbine. ISA Transactions 130, 92–103. doi:10.1016/j.isatra.2022.04.008.

Sollich, P., Krogh, A., 1996. Learning with ensembles: How over-fitting can be useful, in: Proc. of the 1996 Conference on Advances in Neural Information Processing System, pp. 190–196.

Tao, H., Paszke, W., Rogers, E., Yang, H., Gałkowski, K., 2017. Iterative learning fault-tolerant control for differential time-delay batch processes in finite frequency domains. Journal of Process Control 56, 112–128. doi:10.1016/j.jprocont.2016.12.007.

Wei, J., Zhang, Y., Sun, M., Geng, B., 2017. Adaptive iterative learning control of a class of nonlinear time-delay systems with unknown backlash-like hysteresis input and control direction. ISA Transactions 70, 79–92.

Xiong, W., Ho, D.W.C., Yu, X., 2016. Saturated finite interval iterative learning for tracking of dynamic systems with HNN-structural output. IEEE Transactions on Neural Networks and Learning Systems 27, 1578–1584.

Xu, J.X., Tan, Y., 2003. Linear and Nonlinear Iterative Learning Control for Determinisic Systems. volume 291 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin.

Yu, X., Hou, Z., Polycarpou, M., Duan, L., 2021. Data-driven iterative learning control for nonlinear discrete-time mimo systems. IEEE Transactions on Neural Networks and Learning Systems 32, 1136–1148. doi:10.1109/TNNLS.2020.2980588.

Zhang, D., Wang, Z., Masayoshi, T., 2021. Neural-network-based iterative learning control for multiple tasks. IEEE Transactions on Neural Networks and Learning Systems 32, 4178–4190. doi:10.1109/TNNLS.2020.3017158.