Control-relevant neural networks for feedforward control with preview

Control-Relevant Neural Networks for Feedforward Control with Preview: Applied to an Industrial Flatbed Printer ⋆

Leontine Aarnoudse [a], Johan Kon [a], Wataru Ohnishi [b], Maurice Poot [a], Paul Tacx [a], Nard Strijbosch [a], Tom Oomen [a,c]

[a] *Dept. of Mechanical Engineering, Control Systems Technology, Eindhoven University of Technology, Eindhoven, The Netherlands*

[b] *Graduate School of Engineering, The University of Tokyo, Tokyo, Japan*

[c] *Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands*

Abstract

The performance of feedforward control depends strongly on its ability to compensate for reproducible disturbances. The aim of this paper is to develop a systematic framework for artificial neural networks (ANN) for feedforward control. The method involves three aspects: a new criterion that emphasizes the closed-loop control objective, inclusion of preview to deal with delays and non-minimum phase dynamics, and enabling the use of an iterative learning algorithm to generate training data in view of addressing generalization errors. The approach is illustrated through simulations and experiments on an industrial flatbed printer.

Key words: Feedforward control; Neural networks; Iterative learning control

1 Introduction

Feedforward control is essential for the performance of control applications since it can attenuate known disturbances before they effect the system. Through model-based approaches [13], approaches that combine measured data and approximate models [8, 11], and fully data-driven approaches [9], high performance can be achieved for mechatronic systems such as wafer stages [3]. The key challenge in feedforward control is the combination of performance and flexibility, i.e., to achieve high performance for a range of disturbances.

Many techniques for feedforward control have been developed that differ in terms of the three basic entities involved in the construction of models from data [27]: the model structure, the data set and the criterion for assessment of the model. Each of these entities is essential for the performance and flexibility properties of feedforward control. In particular, feedforward control for motion systems typically requires non-causal models [44], data sets based on the required flexibility and control setting, and assessment criteria that take into account the aim of achieving high performance in terms of the closed-loop error [6].

Regarding the model structure, to achieve good performance the models used for feedforward control should be non-causal to enable pre-actuation, and they should be non-linear in order to approximate system nonlinearities. Existing approaches range from signal-based with optimal feedforward signals for predefined basis tasks [20] to parameterized methods including polynomial and rational basis functions [4, 7, 43], and non-parametric methods such as Gaussian processes [37] and artificial neural networks (ANNs). Rational basis functions as well as Gaussian processes are capable of generating non-causal feedforward signals [5], but they are mostly linear or limited to pre-specified nonlinearities like friction. Artificial neural networks are universal nonlinear function approximators [21] that have large potential for the flexibility and performance of feedforward control of motion systems.

ANNs have indeed been explored for feedforward control, which has lead to several improvements, yet the success of

⋆ This work is part of the research programme VIDI with project number 15698, which is (partly) financed by the NWO.

Email addresses: l.i.m.aarnoudse@tue.nl (Leontine Aarnoudse), j.j.kon@tue.nl (Johan Kon), ohnishi@koseki.t.u-tokyo.ac.jp (Wataru Ohnishi), m.m.poot@tue.nl (Maurice Poot), p.j.m.m.tacx@tue.nl (Paul Tacx), n.w.a.strijbosch@tue.nl (Nard Strijbosch), t.a.e.oomen@tue.nl (Tom Oomen).

these ANNs is still limited; the abilities of ANNs for preview and nonlinear feedforward are not yet fully exploited because existing implementations limit the size and structure of the networks. Feedback error learning (FEL) [23,39], similar to [34], successfully uses ANNs to map references to causal feedforward signals. However, it depends strongly on optimization through experiment-based gradient descent and as such only allows for small, shallow ANNs. In [42], a forward NARX model is trained and inverted to find reference-based feedforward signals, and a two-sample preview is successfully added. However, this approach depends on invertible ANNs with monotonic mappings, which strongly limits the available ANN structures.

Regarding the tuning in terms of data sets and assessment criteria, which is a completely different aspect of ANNs for feedforward control, developments in traditional feedforward for control systems have led to key insights. When a nonlinear system is approximated, the approximation will depend on the specific set of input-output data that is used since nonlinearities manifest themselves along the used trajectories [29]. In addition, it is essential that the assessment criterion for the feedforward model is control-relevant, i.e., it relates to the criteria based on which the system performance is assessed [6]. While important steps have been made for ANNs for feedforward control, existing approaches do not consider suitable input-output data or control-relevant assessment criteria. FEL achieves good performance for, e.g., robotic systems, but it uses the feedback controller output as criterion to train the ANN [31], which is neither efficient, because every iteration requires an experiment, nor optimal, because it is not directly aimed at minimizing the tracking error. In [36], an ANN-based learning controller based on error and feedback controller output data is successfully implemented, but the approach does not take into account the dependence of feedforward control on the reference. In [47] and [15], plant input and output data is used to train a forward ANN-based model, to which ILC is applied to obtain feedforward signals. Again, the approach is not optimal because the purpose of feedforward control is not reflected in the training process.

Although artificial neural networks are promising for feedforward control, the selection of suitable model structures, data, and assessment criteria are not yet clear. The aim of this paper is to develop a systematic framework for ANNs for feedforward control that explicitly considers these entities in order to achieve both flexibility and high performance. The contribution consists of four cornerstones:

(C1) a control-relevant loss function that directly aims at minimizing the closed-loop error;
(C2) non-causal neural networks with respectively finite and infinite preview;
(C3) the use of iterative learning control to inexpensively generate suitable training data; and
(C4) illustration of the approach through several simulations and through experiments on an industrial flatbed printer.

In addition, guidelines regarding data acquisition, network design and implementation aspects are provided. Preliminary research appeared in [1]. The present paper extends [1] with extended theory, explanations and implementation guidelines.

In the broader picture of ANNs for control and identification, significant developments have been made. In the field of system identification, ANNs are used to model nonlinear systems [2]. Regarding control, [30] suggests using ANNs as nonlinear controllers for nonlinear systems, [38] uses ANNs with an additional sensor to detect and compensate external disturbances, and in the field of model-predictive control (MPC) ANNs are used to approximate optimal MPC strategies, see, e.g., [14, 17] or the overview in [18].

This paper is structured as follows. In Section 2 the problem considered in this paper is formulated. In Section 3, the need for control-relevant training of ANNs is explained and a control-relevant loss function is designed. The importance of non-causal feedforward and the network structures enabling finite and infinite preview for feedforward control are discussed in Section 4. In Section 5 the use of iterative learning control to generate training data is proposed. In Section 6, a systematic framework for ANNs for feedforward control is introduced, design considerations aimed at motion feedforward are discussed, and some implementation aspects are considered. Experimental results are shown in Section 7. Lastly, conclusions are given in Section 8.

2 Problem formulation

In this section, the considered problem is formulated. First, feedforward control is introduced. Then, artificial neural networks are introduced and their potential for motion feedforward is discussed.

2.1 Feedforward control

Feedforward control aims to compensate for known disturbances. Consider an LTI control system as shown in Figure 1 (left). The discrete-time error signal $e(t)$ with is given by

$$e(t) = Sr(t) - SPf(t) \tag{1}$$

with reference $r(t)$, feedforward signal $f(t)$ and system sensitivity $S = (1+PC)^{-1}$ with plant P and controller C. The reference $r(t)$ is a known disturbance that can be compensated by the feedforward signal $f(t)$. Zero error is achieved for

$$f^*(t) = (SP)^{-1}Sr(t) = P^{-1}r(t). \tag{2}$$

Feedforward signal $f^*(t)$ can, in general, not be determined directly since it depends on both the known reference signal $r(t)$ and the plant P, which is typically not known exactly.

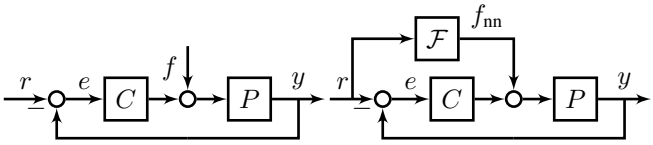

Fig. 1. Standard control setup (left) and control setup with a neural network $\mathcal{F}(r)$ that maps r tof_{nn} (right).

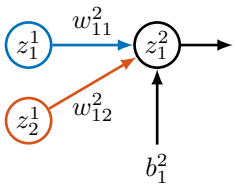

Fig. 2. Example of a neuron z_1^2, the inputs for which are given by the outputs of the neurons in the previous layer, z_1^1 and z_2^1, with weights w_{kj}^l and bias b_k^l.

Existing approaches to approximate $f^*(t)$ include model-based estimations of P or P^{-1} and data-driven methods such as iterative learning control (ILC) that learn a feedforward signal $\hat{f}(t) \approx f^*(t)$ directly for one specific reference [11] or parameterized as a function of $r(t)$ to achieve flexibility for varying references [9, 10].

2.2 *Artificial neural networks*

Artificial neural networks are universal function approximators that are capable of representing complicated and nonlinear mappings [21]. A neural network, illustrated in Fig. 2, consists of neurons that are collected in layers, where each neuron has one or multiple inputs and a single output. The outputs of the neurons in one layer form the inputs for the neurons in the next layer. The output of a neuron z_k^l, the k^{th} neuron in layer l of a simple feedforward neural network (FNN) where the information moves in a single direction, is given by

$$z_k^l = \sigma\left(\sum_{j=1}^{n_{l-1}} w_{kj}^l z_j^{l-1} + b_k^l\right) \tag{3}$$

with $w_{kj}^l \in \mathbb{R}$ the weights, $b_k^l \in \mathbb{R}$ the biases, n_{l-1} the number of neurons in the previous layer and σ an activation function, which typically is a nonlinear mapping.

In this paper, three types of neural networks are considered: feedforward neural networks (FNN), time-delay neural networks (TDNN) [46], and recurrent neural networks (RNN) [41]. In an FNN, the information moves in one direction from the input nodes, through the neurons in the hidden layers, to the output nodes, see Fig. 3. In TDNNs the information moves similarly, but each neuron in the first layer of a TDNN receives, in addition to the current input, a window of time-delayed inputs.

RNNs can be interpreted as nonlinear state space systems

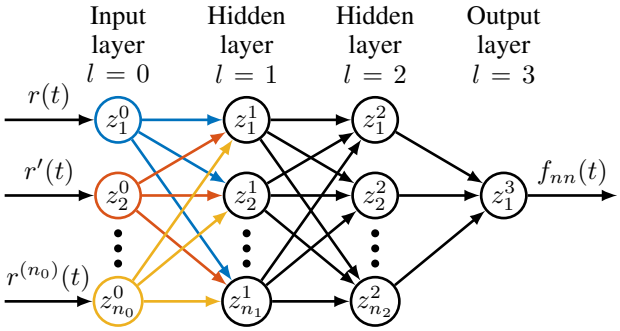

Fig. 3. FNN with two fully connected hidden layers.

[28], because they contain an recurrent state containing information from a distant past. In contrast to time-delay neural networks that only convey information from a limited window of past inputs, the recurrent neural network creates a feedback loop around each node in the net. The information contained in the state may remain available infinitely long.

2.3 *ANNs for feedforward control*

This paper aims to exploit the capabilities of neural networks in finding mappings without requiring a priori system knowledge, in order to achieve both high performance and flexibility for varying references in feedforward control. Specifically, the aim is to use neural networks to find a mapping $\mathcal{F}(r(t))$ for which

$$\hat{f}(t) = \mathcal{F}(r(t)), \tag{4}$$

such that feedforward signal $\hat{f}(t)$ minimizes the error e in (1) for $f(t) = \hat{f}(t)$. This results in the control structure shown in Fig. 1 (right).

Three important aspects of feedforward control are explicitly taken into account to achieve high performance for varying references: 1) a control-relevant loss function is introduced that reflects the aim of minimizing $e(t)$ in the training, 2) the neural networks are designed to enable non-causal feedforward signals with finite or infinite preview, and 3) iterative learning control is used to inexpensively generate closed-loop data to minimize the generalization error.

3 Control-relevant neural networks

In this section a control-relevant loss function is presented that directly reflects the aim of minimizing $e(t)$ in (1) through weighting with an estimate of the process sensitivity $\widehat{SP}$, leading to contribution C1. A simulation example is used to illustrate that typical mean-squared error-based training of neural networks, which is equivalent to using the estimate $\widehat{SP} = 1$, is not suitable for control purposes.

3.1 Control-relevant loss functions

The aim of feedforward control is to minimize the error $e(t)$ in (1). A control-relevant loss function that takes into account this aim is proposed in the following theorem.

Theorem 1 *The approximation $\hat{f}(t)$ of feedforward signal $f^*(t) = P^{-1}r(t)$ that results in the smallest error $e(t)$ in (1) for reference $r(t)$ in terms of the squared vector 2-norm is the minimizer of loss function*

$$\mathcal{J}(\hat{f}(t)) = \left\| SP\left(f^*(t) - \hat{f}(t)\right)\right\|_2^2, \tag{5}$$

with the vector 2-norm defined as $\|x\|_2 = \sqrt{x^\top x}$.

PROOF. Feedforward signal $f^*(t)$ is rewritten to

$$f^*(t) = (SP)^{-1}Sr(t). \tag{6}$$

Using (6), (1) is rewritten to

$$e(\hat{f}(t)) = SPf^*(t) - SP\hat{f}(t) = SP(f^*(t) - \hat{f}(t)), \tag{7}$$

Taking $\|e((\hat{f})t)\|_2^2$ results in loss function

$$\mathcal{J}(\hat{f}(t)) = \left\| SP\left(f^*(t) - \hat{f}(t)\right)\right\|_2^2. \tag{8}$$

In practice, SP is typically not know exactly and a model $\widehat{SP}$ is used instead. To provide robustness against model uncertainty a regularization term may be added as in [12], resulting in the regularized control-relevant loss function

$$\begin{aligned}\mathcal{J}_{\text{CR}}\left(\hat{f}(t)\right) =& w_1\left\|\widehat{SP}\left(f^*(t) - \hat{f}(t)\right)\right\|_2^2 \\ &+ \left\|f^*(t) - \hat{f}(t)\right\|_{W_2}^2,\end{aligned} \tag{9}$$

where w_1 is a scalar weight, and $\|x\|_{W_2}^2 = x^{\mathsf{T}}W_2x$ where W_2 may be chosen, for example, as w_2I. The regularization term also prevents constant offsets in $f^*(t) - \hat{f}(t)$ that may occur if SP is small at low frequencies due to integrators in C. The analysis assumes that P is LTI, but in practice most systems are nonlinear and therefore these offsets could cause undesirable effect. Note that the standard mean-squared error (MSE) loss function that is typically used in regression-based training of neural networks minimizes $\|f^*(t) - \hat{f}(t)\|_2^2$ and is equivalent to (9) with estimate $\widehat{SP} = 1$, which is not suitable for feedforward control in general as illustrated in the next example.

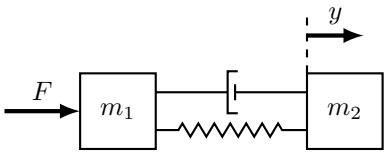

Fig. 4. Non-collocated two-mass spring damper system.

3.2 Example

The importance of the control-relevant loss function is illustrated using an example of two-mass spring damper system. The system is undermodeled and the model parameters are fitted using both mean-squared error and control-relevant loss functions to illustrate the importance of fitting the part of the signal that has the largest influence on the tracking error.

Consider a system consisting of two masses, connected by a spring and a damper, as shown in Fig. 4. An actuator that provides a force input is connected to the first mass, while the position output is measured at the second mass, resulting in a non-collocated system. The transfer function from input to output is given by

$$\begin{aligned}P &= \frac{ds + k}{m_1m_2s^4 + (m_1 + m_2)ds^3 + (m_1 + m_2)ks^2} \\ &= \frac{0.5s + 822.5}{5s^4 + 3s^3 + 4935s^2},\end{aligned} \tag{10}$$

with s the operator variable in the Laplace domain, masses $m_1 = 1$ and $m_2 = 5$, damper constant $d = 0.5$ and spring constant $k = 822.5$, resulting in the Bode diagram of P shown in Fig. 5. The system is put in feedback as shown in Fig. 1 (left), with controller C given by

$$C = \frac{9.549s + 10}{0.1061s + 1}, \tag{11}$$

i.e., a lead/lag filter with a zero on 0.167 Hz, a pole on 1.5 Hz and a gain of 10. The system is discretized using zero-order hold with a sampling frequency of 1 kHz, and a reference given by $r(t) = 0.1(1 - \cos(\pi t)) + 10^{-4}(1 - \cos(40\pi t))$ for $t = 0, 1/1000, ..., 4$, see Fig. 6, is applied.

It is assumed that a feedforward signal $f^*(t)$ is available, which results in zero error for reference $r(t)$. This signal is given by $f^*(t) = P^{-1}r$ according to (2), and can be found using, for example, iterative learning control as shown in Section 6. Signal $f^*(t)$ is projected onto a set of reference-dependent basis functions $\psi(r(t))$, which is equivalent to training an ANN that is linear in the parameters. In particular, a basis function is chosen that approximates the system by a mass, i.e., $\psi(r(t)) = \ddot{r}(t)$. Thus, plant P is approximated by

$$\hat{P} = \frac{1}{\hat{m}s^2}. \tag{12}$$

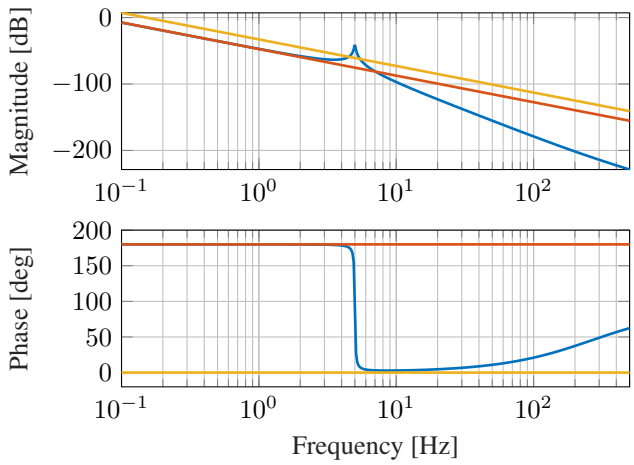

Fig. 5. Bode diagram of a non-collocated two-mass spring damper system (—) and of estimates $\hat{P} = \frac{1}{\hat{m}s^2}$ with $\hat{m}$ based on projected norm-optimal ILC without (—) and with SP-weighting (—). Note the phase difference between the projections.

Note that the system P is undermodeled by $\hat{P}$, and that the estimation holds only for low frequencies before the resonance, where the system behaves as a rigid body. The projected feedforward signal is given by

$$\hat{f}(t) = \hat{P}^{-1} r(t) = \psi(r(t))\hat{m}. \tag{13}$$

Parameter $\hat{m}$ is estimated using both standard MSE training with $\widehat{SP} = 1$ and control-relevant training with $\widehat{SP} = SP$ to illustrate the importance of control-relevant training.

3.2.1 *Standard (unweighted) MSE training*

The unweighted least-squares estimate of $\hat{m}$ based on $f^*(t)$ is given by

$$\begin{aligned}\hat{m}_{\text{MSE}} &= \arg\min_{m\in\mathbb{R}} \|f^*(t) - \psi(r(t))m\|_2^2 \\ &= \left(\psi(r(t))^{\mathsf{T}}\psi(r(t))\right)^{-1}\psi(r(t))^{\mathsf{T}} f^*(t).\end{aligned} \tag{14}$$

This estimation results in the feedforward signal shown in Fig. 7. As shown in the figure, $f^*(t)$ contains much more energy at 20 Hz than the reference signal, due to the high gain of P^{-1} at that frequency. This is reflected in the projected feedforward signal, which also contains most energy at 20 Hz, with the corresponding phase. However, the estimate that follows from (14) is given by $\hat{m}_{\text{MSE}} = -1.12$, while the actual mass for the rigid-body mode is $m_1 + m_2 = 6$. In Fig. 6 it is shown that this estimation, with a phase opposite to that of the system at low frequencies as shown in Fig. 5, results in a high error. Note that in the error signal, most signal content is at 0.5 Hz. This illustrates that the estimate (14) is not suitable for feedforward control in general, since the energy distribution over frequencies in $f^*(t)$ does not necessarily reflect that in the error.

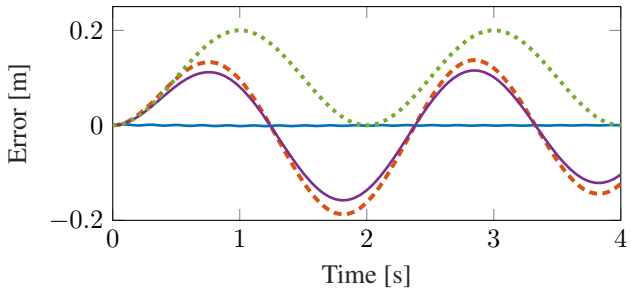

Fig. 6. For a non-collocated two mass-spring-damper system with reference (·····), projecting $f^*(t)$ on a mass basis function without weighting results in a error (- -) larger than the error without any feedforward (—). When the projection is weighted with SP instead, the error is almost 0 (—).

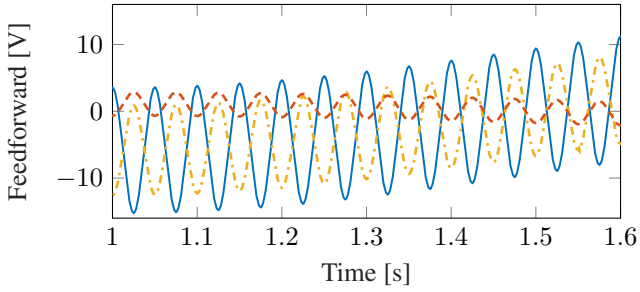

Fig. 7. The feedforward signal $f^*(t)$ used for training (-·-) and its unweighted projection on a mass basis function (- -) are in phase for high frequencies, and in anti-phase for low frequencies. When the projection is weighted with SP, the resulting feedforward signal (—) is in phase with the training signal at low frequencies and in anti-phase at high frequencies.

3.2.2 *Control-relevant training*

Now, estimate $\hat{m}_{\text{CR}}$ is determined using weighting with SP according to (5), such that

$$\begin{aligned}\hat{m}_{\text{CR}} &= \arg\min_{m\in\mathbb{R}} \|SP(f^*(t) - \psi(r(t))m)\|_2^2 \\ &= \left((SP\psi(r(t)))^{\mathsf{T}} SP\psi(r(t))\right)^{-1}(SP\psi(r(t)))^{\mathsf{T}} SP f^*(t).\end{aligned} \tag{15}$$

This results in the estimate $\hat{m}_{\text{CR}} = 5.94$ and the corresponding Bode diagram shown in Fig. 5, which approximates P well at low frequencies. The resulting feedforward signal is shown in Fig. 7, and while it is in anti-phase with $f^*(t)$ at high frequencies, it is in phase at low frequencies and the resulting error shown in Fig. 6 is almost zero.

The example illustrates the importance of using a loss function that reflects the purpose of minimizing the closed-loop tracking error. The best fit in terms of the mean-squared difference between $f^*(t)$ and $\hat{f}(t)$ results in an increase in error compared to not using feedforward. By weighting with SP, an estimate $\hat{f}(t)$ is found that best approximates the part of $f^*(t)$ that has the largest influence on $e(t)$, resulting in good tracking performance.

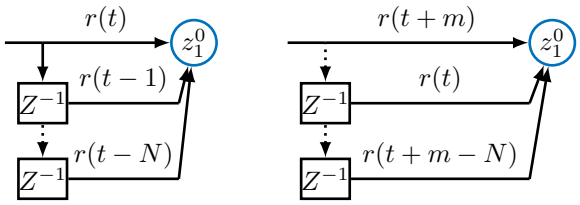

Fig. 8. Input node for causal (left) and non-causal (right) time-delay neural networks with N input delays and m samples preview.

4 Non-causal neural networks

Feedforward signals for motion systems typically require preview to achieve zero error, for example due to plant delays or in the presence of non-minimum phase zeros. The feedforward signal for which zero-error is achieved is given by $f^* = P^{-1}r$ according to (2). For typical discrete-time systems, the plant P contains delays and is therefore strictly proper. As a result, the inverse plant P^{-1}, which is to be approximated by the neural network, is non-causal.

In this section, two types of neural networks are considered that are capable of finding non-causal mappings between the reference and the feedforward signal, respectively with finite and infinite preview: non-causal time-delay neural networks (TDNN) and recurrent neural networks (RNN) with bi-directional long short-term memory (BiLSTM) layers. First, non-causal network structures are introduced, leading to contribution C2. Then, the importance of non-causal feedforward is illustrated using a benchmark non-minimum phase system to which both causal and non-causal neural networks are applied.

4.1 Non-causal neural networks

A non-causal feedforward signal with finite preview can be generated using a time-delay neural network with shifted inputs. A standard TDNN [46] is a causal operator that maps a finite time sequence of input samples $\{r(t), r(t-1), ..., r(t-N)\}$ to a single output $f_{\text{nn}}(t)$. To enable non-causal feedforward signals with finite preview, the input of a standard TDNN is shifted. A preview of m samples is obtained by taking as input $r_{\text{shift}}(t) = r(t+m)$, such that $\{r_{\text{shift}}(t), r_{\text{shift}}(t-1), ..., r_{\text{shift}}(t-N)\} = \{r(t+m), r(t+m-1), ..., r(t+m-N)\}$. Input nodes for causal and non-causal TDNNs are illustrated in Fig. 8.

In typical feedforward control, it is also possible to include infinite preview, see e.g. [44] on inversion for feedforward. Recurrent neural networks with bi-directional long short-term memory layers can be used to obtain feedforward signals with infinite preview. A standard LSTM layer [19] contains a hidden 'state' containing information on prior inputs and outputs. In a biLSTM layer, two LSTM layers are combined. One of the layers receives standard forward data, while the other receives time-reversed data. As such, the biLSTM layer has access to all past and future data, resulting in infinite preview equal to the reference length.

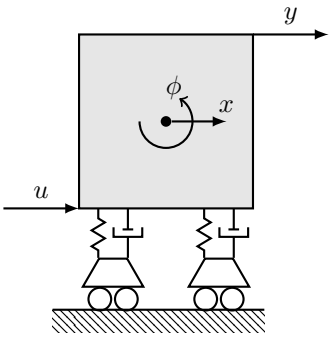

Fig. 9. Non-minimum phase system consisting of a mass that can translate in x-direction and rotate in ϕ-direction with input force u and output y.

4.2 Motivating example: non-minimum phase systems

The importance of preview in feedforward control is illustrated using the non-minimum phase benchmark system shown in Fig. 9, see also [44]. Consider the open loop system from force u [N] to position y [m], given in discrete time by

$$P(z) = \frac{-3 \cdot 10^{-8}(z+0.9632)(z-0.9447)(z-1.1410)}{(z-1)^2(z^2-1.9595z+0.9632)}. \tag{16}$$

The system is sampled at 1 kHz. With the feedback controller $C(z) = \frac{925(z-0.9979)}{z-0.9813}$, the process sensitivity $SP(z)$ becomes

$$SP(z) = \frac{-3 \cdot 10^{-8}(z+0.9632)(z-0.9447)(z-1.1410)(z-0.9813)}{(z-0.9901)(z^2-1.9903z+0.9903)(z^2-1.9605z+0.9640)} \tag{17}$$

The systems P and SP have one non-minimum phase zero $z = 1.1410$, which causes undershoot, i.e., the step response of the system initially moves in the 'wrong' direction [45].

Because of the non-minimum phase behavior of the system, the bounded feedforward signal $f^* = P^{-1}r$ that achieves zero error is non-causal and provides pre-actuation, as shown in Fig. 10. In order to generate a feedforward signal of this form, a non-causal neural network is required.

For the simulated system (17) a data set containing feedforward signals for five different references is generated. One reference is kept apart as test data, the others are used to train several causal and non-causal neural networks with finite and infinite preview as listed in Table 1. Each neural net is trained as explained in Section 6, using a mean-squared error loss function and with the reference and its four derivatives as inputs. In Table 1, the performance of each network for the test reference is expressed by the error 2-norm.

As shown in Fig. 11, non-causal neural networks can provide the pre-actuation required for a small error, in contrast to

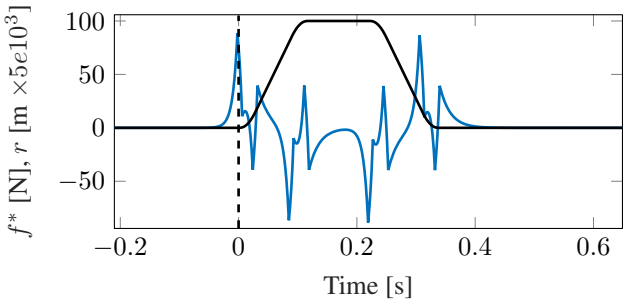

Fig. 10. For a non-minimum phase system and a reference r (—) which is zero for all $t < 0$, the bounded feedforward signal f^* (—) that achieves zero error is non-causal.

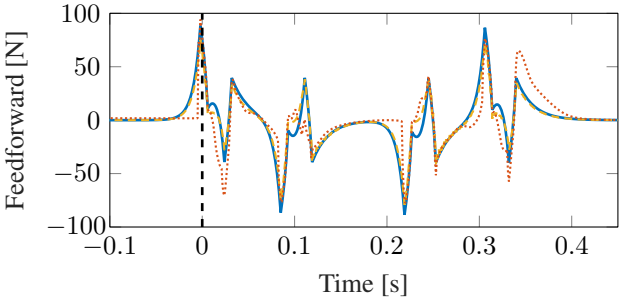

Fig. 11. Feedforward signals for the non-minimum phase system. For a reference starting at $t = 0$, the non-causal feedforward f^* (—) that achieves zero error is nonzero for $t < 0$. The feedforward generated by a bi-directional LSTM RNN (- -) is also non-causal due to infinite preview, while a causal LSTM RNN (·····) has a bias for $t < 0$ and only shows dynamic behavior when the fourth derivative of the reference becomes nonzero.

causal neural networks that, apart from a bias, only start actuating when the reference or its derivatives are nonzero. The results in Table 1 illustrate that non-causality is required to achieve small errors. In addition, it is shown that while non-causal TDNNs are capable of achieving small errors, this depends strongly on the size of the finite preview. As such, these TDNNs have an extra design parameter compared to RNNs with biLSTM layers that have infinite preview.

5 Data acquisition

In this section, it is proposed to use iterative learning control (ILC) to generate closed-loop training data. Specifically, ILC is used to generate feedforward signals for a set of references, leading to contribution C3.

When modeling systems, especially systems with nonlinearities that manifest themselves along the used trajectories, it is essential that the data on which the model is based is maximally informative for the purpose of the model. In the case of ANNs for feedforward, this means that the data should reflect the aim of minimizing $e(t)$ in (1) through feedforward $f(t)$ for various references $r(t)$. Therefore, the ANNs are trained using training data consisting of references and

Table 1
Overview of the causal and non-causal networks used for simulation. The size of each layer is given between parentheses, where 'out' is a linear output layer with one neuron.

Network	$\|e\|_2$
No feedforward	0.278
TDNN, causal (20 samples backview), ReLU(50,50,out)	0.292
TDNN, non-causal (50 samples preview, 20 samples backview), ReLU(50,50,out)	0.083
TDNN, non-causal (20 samples preview, 20 samples backview), ReLU(50,50,out)	0.152
RNN, causal LSTM (features hidden state=5, 2 recurrent layers + output)	0.183
RNN, non-causal biLSTM (features hidden state=5, 2 recurrent layers + output)	0.080

corresponding feedforward signals for which $e(t) \approx 0$. The N references $r_i(t)$, $i = 1, 2, ..., N$ are chosen such that they cover the set of possible references well in terms of positions, directions and position derivatives including velocity, acceleration and jerk.

For each of the references, a feedforward signal is required such that Assumption 2 holds.

Assumption 2 (Perfect training data) *For each reference r_i, the corresponding feedforward signal $f_{\text{train},i}$ results in zero error, i.e.,*

$$e_i(t) = Sr_i(t) - SPf_{\text{train},i}(t) = Sr_i(t) - SPf_i^*(t) = 0. \tag{18}$$

This can be achieved by iterative learning control (ILC) [11], provided that there are no iteration-varying disturbances. In ILC, a feedforward signal that compensates for a repeating disturbance, e.g., a reference signal, is learned iteratively. Each iteration j, the feedforward signal $f_j(t)$ is updated according to

$$f_{j+1}(t) = Q(f_j(t) + \alpha L e_j(t)). \tag{19}$$

The error $e_j(t)$ for feedforward $f(t) = f_j(t)$ is determined experimentally and is defined according to (1). Learning filter L is chosen to approximate SP^{-1} using a suitable inversion approach [44]. Robustness against modeling errors is provided by the filter Q and learning gain α ensures that iteration-varying disturbances are not amplified by the learning. ILC requires few iterations to converge, it is easy to implement and it is capable of attenuating repeating disturbances completely. As such, it is a suitable technique to inexpensively generate training data for neural networks for motion feedforward.

6 Implementation and design considerations

In this section several aspects of ANN design and implementation are considered. First, the approach proposed in this paper is summarized in Algorithm 1. Secondly, some considerations regarding the design of ANNs for motion feedforward are given. Thirdly, the implementation of the control-relevant loss function of Section 3 and the data processing are explained.

Algorithm 1 Neural Networks for Feedforward Control

1: Design a representative set of references (Section 5).
2: Use ILC to find the optimal feedforward input for each of the references (Section 5).
3: Process the data to improve the conditioning, and separate the data in training and testing sets (Section 6.2).
4: Choose a suitable neural network in terms of non-causal properties and other hyperparameters (Section 4, 6.1).
5: Train the network using a control-relevant loss function (Section 3, 6.2).
6: First validation step: evaluate the loss $\mathcal{J}_{\text{CR}}$ for the testing data set.
7: Second validation step: evaluate the experimental tracking error for the testing data set.
8: Use the neural network to generate feedforward signals for arbitrary references.

6.1 Network design for motion feedforward

Several aspects of the design of ANNs for motion feedforward are discussed based on properties of motion systems. First, suitable input signals are selected. Secondly, the network structure and activation functions are discussed.

6.1.1 Input selection

For feedforward control of typical motion systems, it is recommended to use not only the reference but also its derivatives as inputs to the neural network, since the derivative terms determine essential parts of the feedforward signal. In particular, the acceleration term is required for mass feedforward, the velocity term is needed to overcome viscous friction and the snap term is used to compensate for flexible dynamics.

Theoretically, the neural network should be able to compute derivatives of the input itself, as is shown for a very simple TDNN with linear activation functions in [1]. However, this seldom works well in practice due to the different orders of magnitude of the derivatives of the reference, which result in bad conditioning. Due to this conditioning, combined with extensive nonlinear networks with many local minima in the cost landscape, solvers are often not able to find the minimum corresponding to a situation in which the input is differentiated. Therefore, it is recommended to provide the reference derivatives as inputs to the network.

6.1.2 Hyperparameters: layers, neurons and activation functions

The selection of hyperparameters, such as the number of layers and neurons per layer, is crucial for the performance of neural networks. In the case of motion feedforward, the aim is to find a mapping $f = \mathcal{F}(r)$, where the resulting error is zero if $\mathcal{F}(r)$ is equal to the inverse plant P^{-1}. Therefore, the requirements for the neural network depend on the plant. The network should be extensive enough, in terms of the number of layers as well as the neurons and functions within these layers, to approximate P^{-1} well. However, as is typical in most identification problems, there is a trade-off between bias and variance. While a large network may be able to find a mapping close to the true system, i.e., with a small bias, a network with many parameters will have a larger variance error [28] and might suffer from overfitting. In addition, such a network will take longer to train.

Extensive selection of optimal hyperparameters is outside of the scope of this paper and requires further research. However, there are some system properties that can be taken into account when selecting a suitable network. First of all, although LTI systems are assumed at some points in this paper, motion systems typically contain nonlinearities such as friction that can be modeled by ANNs that use nonlinear activation functions. Secondly, in general these nonlinear characteristics can often be described by relatively simple functions, such that relatively small ANNs are often found to give good approximations. This is illustrated in Section 7, where TDNNs with one hidden layer containing twenty neurons achieve good performance.

6.2 Implementation

Nowadays, many tools in various programming languages are available for training neural networks. This section aims to point out some implementation aspects that are of importance regardless of the specific tools that are used. In particular, the implementation of control-relevant loss functions and the processing of the data are discussed.

6.2.1 Filtering through dynamic systems

In many machine learning frameworks, fast training of neural networks is enabled by the use of GPUs instead of CPUs and by automatic differentiation. Both of these steps limit the available predefined functions significantly. While this is typically not a problem for training neural networks, since all relevant functions are available, it does become problematic when, for example, functions from a control systems toolbox are used. Therefore, the straightforward approach of determining the loss (9) by filtering $f_{\text{train}}(t) - \hat{f}(t)$ through a dynamic model of SP using functions such as `lsim` results in extremely slow training of the neural network.

To take advantage of the fast calculations available for specific functions in the field of machine learning, the control-

relevant loss function of Section 3 can be implemented using convolutions. The response $y(k)$ of a stable, causal linear discrete-time dynamic system at sample k is given by

$$y(k) = \sum_{i=0}^{k-1} h(i) f(k-i) \tag{20}$$

with $f(k)$ the input and $h(i)$ the Markov parameters of the system. Note that the closed-loop process sensitivity SP of a physical system is causal and, in general, stable due to the closed-loop controller C. The Markov parameters of the system SP follow from its impulse response, which only needs to be simulated or measured once. During training, (20) can be implemented using the convolution functions available in typical machine learning toolboxes. Note, however, that these functions are typically aimed at image processing for classification, and as such may employ cross-correlation rather than actual convolution. In this case, either the sequence of Markov parameters or the output signal should be flipped to obtain the correct system response.

Remark 3 *Using (20) allows for a procedure that does not require parametric models. The Markov parameters of the system can be determined from a frequency-response measurement of the system. For data generation, it is possible to use lifted ILC [11] which only requires the same Markov parameters.*

6.2.2 Data processing

The training data for the neural network is conditioned to enable efficient training. The magnitudes of the reference derivatives that are used as inputs for the network, especially for higher-order derivatives such as jerk and snap, are typically much higher than that of the reference itself. Therefore, it is advantageous to normalize all input signals as well as the training output. Each input and output signal is normalized by determining the maximum absolute value over all references, and scaling the signal with that value.

To validate the performance of the neural network after training it is important to exclude several of the references with corresponding feedforward signals from the training data and use them as testing data instead. The separation of the available data in training and testing sets is a trade-off between the quality of training and validation. However, when the data is generated using an inexpensive method such as ILC, both data sets can be made sufficiently large. In the particular case of neural networks for motion feedforward, the validation may consist of two steps: First, the loss (9) is determined for the testing data. Secondly, if the calculated loss is found to be acceptable, the feedforward signal is implemented on the system to determine the experimental performance.

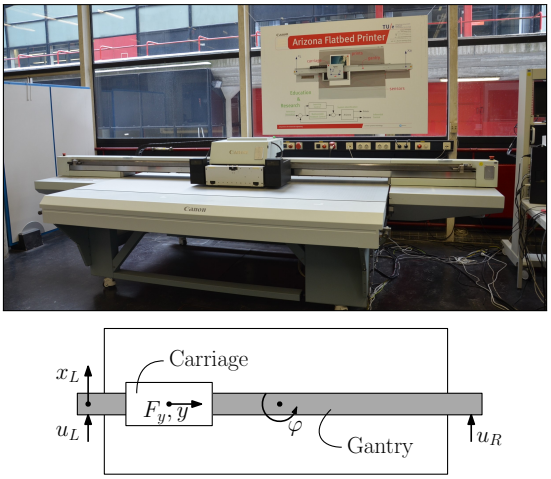

Fig. 12. Photo (top) and schematic overview (bottom) of the Arizona flatbed printer.

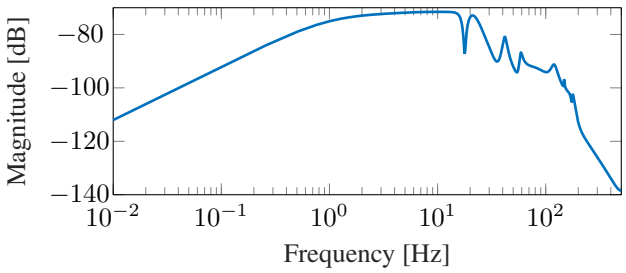

Fig. 13. Process sensitivity SP of the y-axis of the printer.

7 Experimental results

In this section the proposed framework for neural networks for feedforward control is illustrated using an industrial flatbed printer. First, the experimental setup and implementation are introduced. Then, the results for different types of neural networks are compared and discussed.

7.1 Experimental setup

The approach is illustrated using an industrial Arizona flatbed printer, shown in Fig. 12. A single-input single-output situation is considered, in which only the carriage is moved. The input F_y is the force on the carriage, and the output y is its position. The translation and rotation of the gantry have a reference equal to zero, and no feedforward is applied in these directions. A Bode magnitude plot of the process sensitivity of the y-axis of the flatbed printer is shown in Fig. 13. In the remainder of this section the data acquisition is described, and the training and implementation using PyTorch are described.

Remark 4 *Experimental setups may be subject to non-reproducible disturbances such as noise. This influences the performance of feedforward control, and Assumption 2 may not be met. Since feedforward control does not influence the stability of the closed-loop system the method remains*

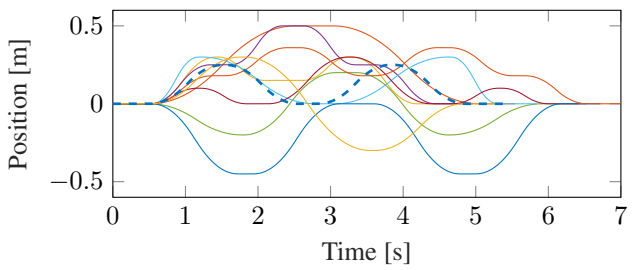

Fig. 14. Nine training references (solid) and the testing reference (dashed).

feasible as long as the main disturbances are reproducible. Two approaches to reducing the influence of noise are as follows.

(1) To reduce the noise in the ILC training data, ILC should be applied with a small learning gain which averages the noise over multiple realizations [33].
(2) The influence of noisy data on ANN training is analyzed in [26], which investigates instrumental variables as a solution.

7.1.1 Data acquisition using ILC

In accordance with Algorithm 1, a representative set of ten fourth-order references is designed that covers a set of positions, velocities, accelerations, jerk and snap values, see Fig. 14. The length of the references varies between 5-7 s, sampled at 1000 Hz. Thus, each reference provides between 5000 and 7000 data points.

Standard frequency-domain ILC [11] is used to find a feedforward signal for each of the references, requiring around ten iterations for each reference. One of the references for which the feedforward signal is known and that is comparable to the training references in terms of the derivative values, is reserved as testing data. This signal is used for a first evaluation step, i.e., evaluating the theoretical cost for a signal for which the optimal feedforward signal is known, as well as for the second evaluation step, i.e., experimental validation.

7.1.2 Neural network training and implementation

The neural networks are trained using PyTorch [35] software in Python, which uses automatic differentiation to allow for fast backpropagation and training. First, the training data is normalized to improve the conditioning during training. Then, different types of neural networks are designed and trained. While there are many algorithms available for training neural networks, the Adam algorithm [24] is preferred. It was observed that the Adam algorithm results in relatively fast training without stability issues, as opposed to, for example, stochastic gradient descent, which converges slowly, or LBFGS, which can converge fast but in this case tends to diverge easily. The ANN weights are initialized using Kaiming Uniform or He initialization [16], which is the default initialization in PyTorch.

Although the time required for training varies depending on the network, in general the training is fast because of the relatively small data set used in this paper. For each network, the training process consists of 1000 epochs with a higher learning rate of $5e^{-2}$, followed by 15000 epochs at a learning rate of $3e^{-3}$. The training is conducted on a standard workstation: a Lenovo Thinkpad P1 laptop running Windows 10, with a NVIDIA Quadro T1000 graphics card and an Intel Core i7 9750H processor. 100 epochs with nine training references take approximately 6 seconds for the nonlinear FNN (see entry FNC in Table 2) and approximately 60 seconds for the much larger biLSTM RNN (see entry RNC in Table 2). Since the training is relatively fast, even for large networks, this shows that there is room to increase the size of the data set or the networks.

Remark 5 *In this paper the feedforward signals based on the ANNs are computed offline, such that real-time computation requirements are of no concern. Computing the feedforward signal for the complete test reference of 5500 samples takes approximately* $0.5\,\mathrm{s}$ *for the nonlinear FNN, and* $0.6\,\mathrm{s}$ *for the non-causal RNN. In general, for sufficiently small ANNs that employ limited preview, real-time computation of feedforward signals can be feasible, see, e.g., [34, 39].*

7.2 Overview of the networks used in experiments

Different types of neural networks (FNNs, TDNNs and RNNs) are tested on the experimental setup to illustrate the influence of finite and infinite preview, and the choice of activation functions and other hyperparameters. All networks use as input the 4th-order reference with its first four derivatives (velocity, acceleration, jerk, snap), in accordance with Section 6.1.1. The networks that are used are given in Table 2. Each network configuration is trained using the control-relevant loss function

$$\mathcal{J}\left(\hat{f}\right) = w_1 \left\| SP\left(f_{\text{train}} - \hat{f}\right)\right\|_2^2 + \left\| f_{\text{train}} - \hat{f}\right\|_{W_2}^2, \quad (21)$$

with $w_1 = 1$, $W_2 = 0.1I$. Additionally, for comparison two of the network configurations are also trained with $w_1 = 0$, $W_2 = I$, resulting in a mean-squared error loss function.

After training, the neural nets are used to generate a feedforward signal for a reference outside of the training set. The resulting feedforward signals are each tested on the setup five times, and the experimental loss is determined by replacing the term $SP\left(f_{\text{train}} - \hat{f}\right)$ in the loss function by the average squared 2-norm of the error of these five experiments. The results of the different networks are summarized in Table 3.

Table 2
Overview of the networks tested in experiments. The size of each layer is given between parentheses, where 'out' is a linear output layer with one neuron. ANNs are trained with a control-relevant (CR) or mean-squared error (MSE) loss function.

Name	ANN design	Loss
FLCC	FNN, Linear (20,20,out), Causal	CR
FLCM	FNN, Linear (20,20,out), Causal	MSE
FNCC	FNN, Nonlinear: ReLU (50,50,out), Causal	CR
TNCC	TDNN, Nonlinear: ReLU(50,50,out), Causal (20 samples backview)	CR
TNNC	TDNN, Nonlinear: ReLU(50,50,out), Noncausal (20 samples backview, 10 preview)	CR
TNNM	TDNN, Nonlinear: ReLU(50,50,out) Noncausal (20 samples backview, 10 preview)	MSE
RNCC	RNN, Nonlinear, Causal: LSTM (features hidden state=50, 2 recurrent layers + output)	CR
RNNC	RNN, Nonlinear, Noncausal: biLSTM (features hidden state=50, 2 recurrent layers + output)	CR

Table 3
The loss on the training data and the experimental loss on the test reference according to (21) with $w_1 = 1$, $W_2 = 0.1I$ for each of the ANNs in Table 2. Control-relevant training and preview in TDNNs improve performance.

Net	Loss training	Loss test exp.
f_{train}	0	4.606×10^1
FLCC	3.355×10^3	3.130×10^2
FLCM	3.451×10^3	3.351×10^2
FNCC	1.201×10^3	1.246×10^2
TNCC	1.173×10^3	1.232×10^2
TNNC	1.174×10^3	1.015×10^2
TNNM	1.486×10^3	1.082×10^2
RNCC	2.640×10^2	9.789×10^2
RNNC	2.234×10^2	1.042×10^3

7.3 Discussion of experimental results

In the following paragraphs, different aspects of the results shown in Table 3 are discussed. The main observations are the following:

- Using nonlinear activation functions results in a loss reduction of at least 50% compared to a linear network that is equivalent to polynomial basis functions.
- The improvements due to training with a control-relevant loss function and adding preview to TDNNs are visible in the experimental results. Even though for this setup other phenomena dominate the performance, the results confirm the relevance of these steps.
- RNNs are very sensitive to overfitting and using them may lead to reduced performance.

Next, each of the observations is further explained.

7.3.1 Influence of nonlinear activation functions

As explained in Section 6.1.2, typical motion systems shown nonlinear behavior, such as nonlinear friction characteristics. Therefore, it is beneficial to include nonlinear activation functions in the networks, which also increases the flexibility of the networks significantly. This is best illustrated by comparing the performance of a linear FNN (FLCC) with a nonlinear FNN (FNCC). Note that a linear FNN is equivalent to the existing method of feedforward control with polynomial basis functions [32]. Using a neural network nonlinear activation functions instead is shown to result in a performance improvement of more than 50%.

7.3.2 Control-relevance

The performance improvements due to the control-relevant loss function are small yet visible. This follows from the comparison of the linear FNNs with control-relevant and MSE training, respectively FLCC and FLCM, and from the comparison of non-causal TDNN with CR and MSE training, respectively TNCC and TNCM. In Table 3, it is shown that CR training results in a lower loss on both the training data and the testing reference in experiments.

7.3.3 Causal and non-causal TDNNs

Taking into account a trail of previous reference samples is not shown to have a significant effect on performance. This follows from the comparison of nonlinear FNNs (FNCC) and nonlinear, causal TDNNs (TNCC) in Table 3.

Adding a finite preview of samples shows some improvements when comparing causal (TNCC) and noncausal (TNNC) TDNNs, but the effect is small. This is most likely related to the fact that the setup does not suffer from undershoot, and most of the preview required to compensate for the sampling delays is already provided by the reference derivatives.

7.3.4 The performance of RNNs

Based on the losses during training, it could be expected that the recurrent neural networks would result in the best performance. However, as shown in Table 3, this is not the case. The most likely cause is overfitting, which would lead to decreased performance for references that are not part of the training set, especially since small inaccuracies are often amplified by RNNs.

Comparison of Fig. 15 and 16 shows that while the RNN fits a signal within the training set much better than other nets, it does not generalize well for the testing reference. This

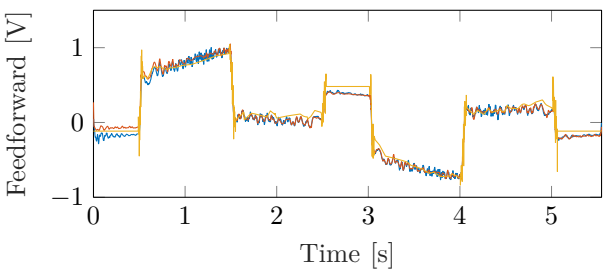

Fig. 15. For references within the training set, the feedforward signal generated by a non-causal RNN (RNNC, —) approximates the training signal (f_{train}, —) much better than the signal from a non-causal TDNN (TNNC, —).

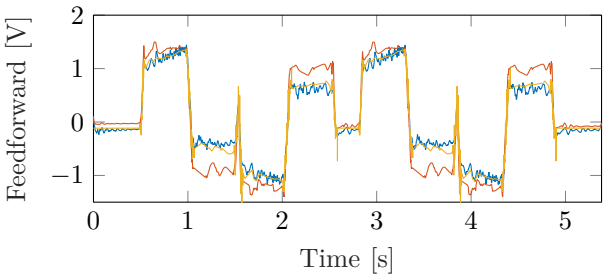

Fig. 16. For the testing reference, the feedforward signal generated by a non-causal RNN (RNNC, —) deviates strongly from the training signal (f_{train}, —), while the signal from a non-causal TDNN (TNNC, —) still gives a reasonably good approximation.

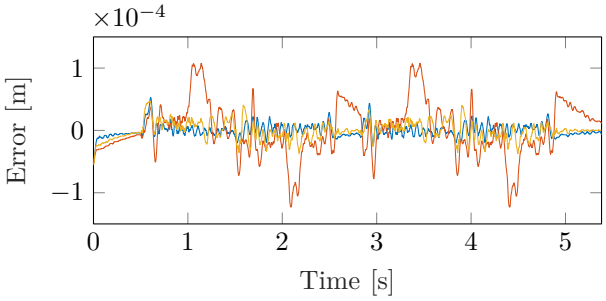

Fig. 17. Errors for the testing reference resulting from the training data (f_{train}, —), a non-causal TDNN (TNNC, —) and a non-causal RNN (RNNC, —). The sections where the error of the RNN is large correspond to the parts where f_{train} is not well approximated, cf. Fig. 16.

indicates that overfitting may be the cause, and illustrates that a more complicated net does not necessarily result in better performance. In contrast, the TDNN shown in the figures may not approximate the signal in the training set as well as the RNN, but its performance for signals inside and outside of the training set is comparable. In Fig. 17 it is shown that the mismatches in the feedforward signal generated by the RNN result in large errors in the same time intervals.

8 Conclusion

In this paper a systematic framework to artificial neural networks for feedforward control is proposed that can achieve high performance and flexibility through choices in model structure, data and assessment criteria. A control-relevant loss function is introduced that is aimed explicitly at minimizing the tracking error, and it is shown that standard loss functions are in general not suitable for feedforward control. In addition, non-causal ANNs with respectively finite and infinite preview are introduced and the importance of pre-actuation in feedforward control is illustrated using simulations. In the proposed framework, closed-loop iterative learning control techniques are used to inexpensively gather training data. Different design and implementation aspects for motion feedforward are discussed, and the approach is illustrated experimentally, where different types of ANNs are used to generate feedforward signals for an industrial flatbed printer.

In future work, the design and performance of ANNs for motion feedforward should be investigated further. The experimental results show the feasibility of the overall approach, but regarding the use of control-relevant loss functions and non-causal networks the observed performance improvements are small and these techniques require further experimental validation. In addition, the design of optimal neural networks for motion feedforward in terms of network types and hyperparameters requires more research. The risk of overfitting when using RNNs may be reduced through, e.g., early stopping and increasing the number of training references. In addition, the use of stable RNNs, see, e.g., [40], could be investigated.

Ongoing work is also aimed at using model knowledge in parallel to ANNs [22], resulting in the application of physics-guided neural networks to feedforward control in [25], where regularization is used to ensure that the contributions of model and ANN are orthogonal. In addition, the criteria used in this paper aim at minimizing the fit error and assume deterministic data. In [26], noisy data is assumed and the stochastic errors, which could lead to closed-loop issues in training of ANNs, are analyzed.

Acknowledgements

The authors gratefully acknowledge the contributions to this paper through a new challenge-based learning project by Lennard Ceelen, Spyros Chatzizacharias, Mathyn van Dael, Yves Elmensdorp, Gijs Herings, Jilles van Hulst, Laurens Kools, Martijn Kortenhoeven, Mike Mostard, Bart Reijnen, Pim Scheers, Matthijs Schotman, Stan Verbeek, Joey Verdonschot, Peter Verheijen, and Jelle de Vries.

The authors also wish to thank Koen Tiels and Sjirk Koekebakker for their contributions.

References

[1] Leontine Aarnoudse, Wataru Ohnishi, Maurice Poot, Paul Tacx, Nard Strijbosch, and Tom Oomen. Control- relevant neural networks for intelligent motion feedforward. In *2021 IEEE Int. Conf. Mechatronics*, 2021.

[2] Carl Andersson, Antônio H. Ribeiro, Koen Tiels, Niklas Wahlström, and Thomas B. Schön. Deep convolutional networks in system identification. In *IEEE Conf. Decis. Control*, pages 3670–3676, Nice, France, 2019. IEEE.

[3] Lennart Blanken, Frank Boeren, Dennis Bruijnen, and Tom Oomen. Batch-to-batch rational feedforward control: From iterative learning to identification approaches, with application to a wafer stage. *IEEE/ASME Trans. Mechatronics*, 22(2):826–837, 2017.

[4] Lennart Blanken, Goksan Isil, Sjirk Koekebakker, and Tom Oomen. Data-driven feedforward tuning using non-causal rational basis functions: With application to an industrial flatbed printer. In *Am. Control Conf.*, Wisconsin Cennter, Milwaukee, USA, 2018.

[5] Lennart Blanken and Tom Oomen. Kernel-based identification of non-causal systems with application to inverse model control. *Automatica*, 114:108830, 2020.

[6] Frank Boeren, Abhishek Bareja, Tom Kok, and Tom Oomen. Frequency-Domain ILC Approach for Repeating and Varying Tasks: With Application to Semiconductor Bonding Equipment. *IEEE/ASME Trans. Mechatronics*, 21(6):2716–2727, 2016.

[7] Frank Boeren, Lennart Blanken, Dennis Bruijnen, and Tom Oomen. Optimal Estimation of Rational Feedforward Control via Instrumental Variables: With Application to a Wafer Stage. *Asian J. Control*, 20(3):975–992, 2018.

[8] Frank Boeren, Tom Oomen, and Maarten Steinbuch. Iterative motion feedforward tuning: A data-driven approach based on instrumental variable identification. *Control Eng. Pract.*, 37:11–19, 2015.

[9] Joost Bolder, Stephan Kleinendorst, and Tom Oomen. Data-driven multivariable ILC: Enhanced performance by eliminating L and Q filters. *Int. J. Robust Nonlinear Control*, 28(12):3728–3751, 2018.

[10] Joost Bolder and Tom Oomen. Rational Basis Functions in Iterative Learning Control—With Experimental Verification on a Motion System. *IEEE Trans. Control Syst. Technol.*, 23(2):722–729, 2015.

[11] D. A. Bristow, M. Tharayil, and A. G. Alleyne. A survey of iterative learning control. *IEEE Control Syst.*, 26(3):96–114, 2006.

[12] Douglas A Bristow. Weighting Matrix Design for Robust Monotonic Convergence in Norm Optimal Iterative Learning Control. In *Am. Control Conf.*, pages 4554–4560, Seattle, Washington, USA, 2008.

[13] J. A. Butterworth, L. Y. Pao, and D. Y. Abramovitch. Analysis and comparison of three discrete-time feedforward model-inverse control techniques for nonminimum-phase systems. *Mechatronics*, 22(5):577–587, 2012.

[14] Steven W. Chen, Tianyu Wang, Nikolay Atanasov, Vijay Kumar, and Manfred Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022.

[15] Zhu Chen, Xiao Liang, and Minghui Zheng. Deep Iterative Learning Control for Quadrotor ' s Trajectory Tracking. In *2021 Am. Control Conf.*, pages 1404–1409, New Orleans, USA, 2021.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proc. IEEE Int. Conf. Comput. Vis.*, 2015 Inter:1026–1034, 2015.

[17] Michael Hertneck, Johannes Kohler, Sebastian Trimpe, and Frank Allgower. Learning an Approximate Model Predictive Controller with Guarantees. *IEEE Control Syst. Lett.*, 2(3):543–548, 2018.

[18] Lukas Hewing, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annu. Rev. Control. Robot. Auton. Syst.*, 3:269–296, 2020.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[20] David J. Hoelzle, Andrew G. Alleyne, and Amy J. Wagoner Johnson. Basis task approach to iterative learning control with applications to micro-robotic deposition. *IEEE Trans. Control Syst. Technol.*, 19(5):1138–1148, 2011.

[21] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[22] Anuj Karpatne, Gowtham Atluri, James H. Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Trans. Knowl. Data Eng.*, 29(10):2318–2331, 2017.

[23] Mitsuo Kawato, Yoji Uno, Michiaki Isobe, and Ryoji Suzuki. Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics. *IEEE Control Syst. Mag.*, 8(2):8–15, 1988.

[24] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd Int. Conf. Learn. Represent.*, 2015.

[25] Johan Kon, Dennis Bruijnen, Jeroen van de Wijdeven, Marcel Heertjes, and Tom Oomen. Physics-Guided Neural Networks for Feedforward Control: An Orthogonal Projection-Based Approach. In *Am. Control Conf.*, 2022.

[26] Johan Kon, Marcel Heertjes, and Tom Oomen. Neural Network Training Using Closed-Loop Data: Hazards and an Instrumental Variable (IVNN) Solution. In *14th IFAC Work. Adapt. Learn. Control Syst.*, 2022.

[27] Lennart Ljung. *System Identification*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[28] Lennart Ljung, Carl Andersson, Koen Tiels, and Thomas B. Schön. Deep learning and system identification. In *21st IFAC World Congr.*, 2020.

[29] Ola Markusson, Håkan Hjalmarsson, and Mikael Norrlöf. A general framework for iterative learning control. *IFAC Proc. Vol.*, 15(1):387–392, 2002.

[30] Kevin L. Moore. *Iterative Learning Control for Deterministic Systems*. Springer-Verlag London Limited, 1993.

[31] Magnus Nilsson and Bo Egardt. A clarifying analysis of feedback error learning in an LTI framework. *Int. J. Adapt. Control Signal Process.*, 22:875–901, 2008.

[32] Tom Oomen. Control for precision mechatronics. In John Baillieul and Tariq Samad, editors, *Encycl. Syst. Control*. Springer, London, 2nd edition, 2019.

[33] Tom Oomen and Cristian R. Rojas. Sparse iterative learning control with application to a wafer stage: Achieving performance, resource efficiency, and task flexibility. *Mechatronics*, 47:134–147, 2017.

[34] Gerco Otten, Theo J. A. De Vries, Job Van Amerongen, Adrian M. Rankers, and Erik W. Gaal. Linear motor motion control using a learning feedforward controller. *IEEE/ASME Trans. Mechatronics*, 2(3):179–187, 1997.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

[36] Krzysztof Patan and Maciej Patan. Neural-network-based iterative learning control of nonlinear systems. *ISA Trans.*, 98:445–453, 2019.

[37] Maurice Poot, Jim Portegies, Noud Mooren, Max Van Haren, Max Van Meer, and Tom Oomen. Gaussian Processes for Advanced Motion Control. *IEEJ J. Ind. Appl.*, 11(3), 2022.

[38] Xuemei Ren, Frank L. Lewis, and Jingliang Zhang. Neural network compensation control for mechanical systems with disturbances. *Automatica*, 45(5):1221–1226, 2009.

[39] F. Resquín, J. Gonzalez-Vargas, J. Ibáñez, F. Brunetti, I. Dimbwadyo, L. Carrasco, S. Alves, C. Gonzalez-Alted, A. Gomez-Blanco, and J. L. Pons. Adaptive hybrid robotic system for rehabilitation of reaching movement after a brain injury: A usability study. *J. Neuroeng. Rehabil.*, 14(1):1–15, 2017.

[40] Max Revay, Ruigang Wang, and Ian R Manchester. A Convex Parameterization of Robust Recurrent Neural Networks. *IEEE Control Syst. Lett.*, 5(4):1363–1368, 2021.

[41] D.E. Rumelhart and J.L. McClelland. Schemata and Sequential Thought Processes in PDP Models. In *Parallel Distrib. Process.*, pages 224–249. The MIT Press, 1986.

[42] Ole Sørensen. Additive feedforward control with neural networks. In *14th Trienn. IFAC World Congr.*, pages 1378–1383, Beijing, China, 1999.

[43] Stan H. Van Der Meulen, Rob L. Tousain, and Okko H. Bosgra. Fixed structure feedforward controller design exploiting iterative trials: Application to a wafer stage and a desktop printer. *J. Dyn. Syst. Meas. Control. Trans. ASME*, 130(5):0510061–05100616, 2008.

[44] Jurgen van Zundert and Tom Oomen. On inversion-based approaches for feedforward and ILC. *Mechatronics*, 50(November 2016):282–291, 2018.

[45] M. Vidyasagar. On Undershoot and Nonminimum Phase Zeros. *IEEE Trans. Automat. Contr.*, 31(5):440, 1986.

[46] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Trans. Acoust.*, 37(3):328–339, 1989.

[47] Zhihua Xiong and Jie Zhang. A batch-to-batch iterative optimal control strategy based on recurrent neural network models. *J. Process Control*, 15(1):11–21, 2005.