Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

A Learning-based Iterative Control Framework for Controlling a Robot Arm with Pneumatic Artificial Muscles

Hao Ma, Dieter Büchler, Bernhard Schölkopf, Michael Muehlebach

***Abstract*—In this work, we propose a new learning-based iterative control (IC) framework that enables a complex soft-robotic arm to track trajectories accurately. Compared to traditional iterative learning control (ILC), which operates on a single fixed reference trajectory, we use a deep learning approach to generalize across various reference trajectories. The resulting nonlinear mapping computes feedforward actions and is used in a two degrees of freedom control design. Our method incorporates prior knowledge about the system dynamics and by learning only feedforward actions, it mitigates the risk of instability. We demonstrate a low sample complexity and an excellent tracking performance in real-world experiments. The experiments are carried out on a custom-made robot arm with four degrees of freedom that is actuated with pneumatic artificial muscles. The experiments include high acceleration and high velocity motion.**

I. Introduction

Since their invention, pneumatic artificial muscles (PAMs) have been widely used for various tasks, such as aerospace applications, [7, 32], medical applications, [26, 22], and industrial applications, [11, 31]. Due to their high power-to-weight ratio, PAMs enable very fast motions, [8]. Moreover, their compliance and low inertia improve safety in interaction with humans, which might enable PAMs to be used in warehouse robotics or factories, where robots work alongside humans. In our research, we use PAMs to actuate the robot arm shown in Figure 1, which generates high velocities and accelerations (as required in table tennis, for example). This indeed imposes challenging requirements on the speed and accuracy of our tracking/control algorithms. However, despite the high power-to-weight ratio and compliance that PAMs offer, they are inherently nonlinear due to the compressibility of air and the nonlinear force length and force velocity characteristics of the muscle, [2]. This motivates the current article that introduces a learning control framework, which can compensate for these nonlinearities in a reliable and data-efficient manner. The effectiveness of our approach is shown in this video: https://youtu.be/kR9jowEH7PY, where the robot arm is able to intercept ping-pong balls that are played to the machine with a success rate of close to 100%.

A. Related Work

We divide the related work section into two parts. The first part discusses the literature on trajectory tracking with PAMs, while the second part summarizes several works that have used machine learning for solving tracking/control problems for complex systems.

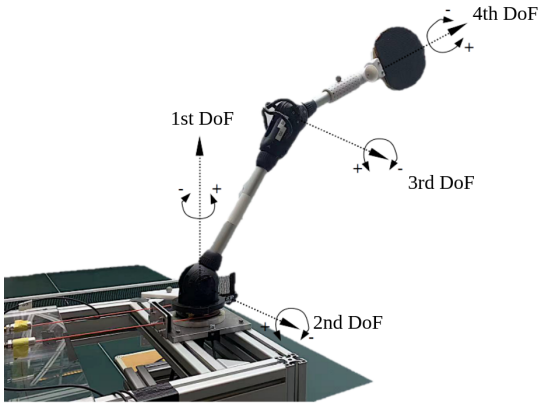


Fig. 1: The figure shows the structure of the robot arm. It has four rotational joints, and each joint is actuated by a pair of PAMs. For simplicity we consider only the first three degrees of freedom in this article. Note that DoF stands for degree of freedom.

Researchers have tried various methods to achieve precise control of PAMs. In the early stages of research, a theoretical model, see [30, 24], was used to design a controller. For example, Ganguly et al. [13] constructed a model based on the structural characteristics and aerodynamic models of PAMs and used traditional proportional-integral-derivative (PID) controllers to steer the muscles. Ba et al. [4] used a neural network to approximate the internal dynamics of PAMs and achieved precise control by accounting for nonlinearities. Kogiso et al. [18] and Hofer and D'Andrea [14] performed parameter identification of a structured model by using experimental data, thereby avoiding a purely white-box modeling. Büchler et al. [9] went even a step further, completely abandoned a first-principles model of PAMs, and used Bayesian optimization for trajectory tracking tasks.

Due to the high nonlinearity of PAMs, Hofer et al. [15] and Zughaibi et al. [34] proposed the use of ILC to improve the position tracking performance for an articulated soft robotic arm during aggressive maneuvers. ILC is a learning control method, which stands for the repeatability of operating a given system and the possibility of updating the control input based on previous operation data to improve the transient performance of systems over a fixed time interval, [3, 6, 1]. ILC as a feedforward control has been proven to have excellent

performance in trajectory tracking. For example, Mueller et al. [23] and Schoellig et al. [28] achieved high-performance tracking of quadrocopters using ILC; a similar performance was also achieved with other complex systems, [33, 17]. However, the most fatal flaw of ILC is that it only works for fixed reference trajectories. If the reference trajectory changes, ILC needs to be trained from scratch, which is not only time-consuming but also infeasible in situations where the reference trajectory frequently changes (for example when intercepting balls that are played to the robot). This motivates our work, which generalizes the ILC approach from fixed reference trajectories to varying reference trajectories.

A similar idea is pursued by Chen et al. [12], where deep learning is used to reduce the number of iterations by ILC. In contrast to their work, we use a convolutional neural network to directly learn a mapping from the reference trajectory to feedforward inputs. We also quantify the generalization capabilities of the network on different reference trajectories (we do not require retraining with ILC) and demonstrate excellent trajectory tracking on a challenging soft-robotic system.

Another common machine learning method that can deal with the tracking/control problems of complex systems is reinforcement learning. Hwangbo et al. [16] and Lee et al. [19] trained a neural network policy and then transferred it to a legged robot, which enabled the robot to overcome challenging terrain. Büchler et al. [10] used a reinforcement learning method for enabling a PAM-driven robot arm to play table tennis. Alternative approaches include guided policy search, [21, 20], which aims at finding a global feedback policy. However, this requires a parametrization over feedback policies. In contrast to all of these, our proposed framework learns only feedforward inputs and can compensate for repeatable disturbances in a non-causal way (it can anticipate repeatable disturbances, something which cannot be done when learning feedback policies).

B. Contribution

In this work, we abandon the theoretical modeling of PAMs and rely on data-driven methods to achieve precise control. However, in most of our methods, we could easily include (approximate) first-principle models as prior knowledge, which would speed up the learning and reduce the sample complexity further. In a first step, we will perform a system identification experiment, resulting in models that have a concrete interpretation as linear dynamical systems and can, in principle, be related to the underlying physics of the robot arm. Compared to the traditional identification method in frequency domain, we use a non-standard methodology, which allows us to not only measure the frequency response function, but also precisely characterize the degree of nonlinearity. In addition, the results characterize the actuation bandwidth of pneumatic actuators. Starting from our linear model, we will use ILC to learn and compensate repeatable disturbances when tracking trajectories. The ILC accounts for unmodeled nonlinearities and actuation biases, and achieves excellent tracking performance in our experiments. We will use deep learning to interpolate between the feedforward inputs of ILC (for different reference trajectories). This results in a nonlinear feedforward controller that can handle different reference trajectories and generalizes the excellent tracking performance of ILC to non-fixed reference trajectories. Our method is also one of the few papers that compares learning-based approaches to traditional control methodologies. We are able to demonstrate that for our soft-robotic system traditional control approaches do not yield satisfactory performance, due to the high nonlinearity and friction of the actuation.

Key advantages of our approach are a low number of hyperparameters, which have a physical interpretation and can be tuned in a principled way. Our approach also incorporates prior knowledge about the system dynamics, which guides the learning by providing closed-form gradient information. This not only avoids gradient computations via finite differences or sampling-based approaches, but also reduces the sample complexity. Due to the fact that deep learning is only used for computing feedforward controls, the method mitigates the risk of destabilizing the system. This will be further discussed in Section II.

C. Structure

This paper is structured as follows. In Section II, we will give an overview of the proposed learning control framework. Section III presents system identification results for each degree of freedom near its operating point. The resulting models provide important insights into the dynamic properties (such as the bandwidth) of each degree of freedom. However, this system identification step could also be skipped if a first-principle model is available. In Section IV, we will use ILC to learn feedforward controls for different fixed reference trajectories, whereby the identification results from Section III are incorporated as prior knowledge. The fixed reference trajectories arise from the specific task for which the robot arm is used. Since in our case the robot arm is used for playing ping-pong, the fixed reference trajectories arise from the interception of ping-pong balls that are played to the robot arm. In Section V, we will design a neural network to generalize the results obtained by ILC. We will also compare the results of the neural network with the results of ILC in experiments. The article concludes with a discussion in Section VI.

II. Overview

Our learning-based iterative control framework builds upon the traditional two degrees of freedom control structure, which is shown in Figure 2.

The concrete implementation steps of our framework are as follows:

1) Design a model that approximates the real system (either first-principles or system identification).
2) Sample suitable fixed reference trajectories for the given task (e.g. intercepting ping-pong balls).
3) For each fixed reference trajectory, run the ILC algorithm to learn feedforward inputs.

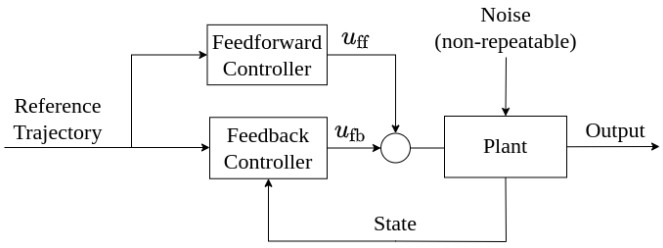


Fig. 2: The figure shows the block diagram of the traditional two degrees of freedom control design, which is used to build our learning-based IC framework.

4) Train a deep network on the fixed reference trajectories and the feedforward inputs. This yields the nonlinear feedforward block shown in Figure 2.

Unlike reinforcement learning, which improves system performance by learning a feedback policy, we propose to learn the feedforward block. This means that our framework is safe, since in many cases, it can be guaranteed that the learning cannot destabilize the system. For example, it is a control theoretic fact that for linear systems, feedforward cannot lead to instability. This fundamental observation also holds to some extent to nonlinear systems, such as the ones which are input to state stable or passive. More precisely, if our feedback controller is designed in such a way that the closed-loop system is input-to-state stable (in the presence of constraints such a feedback controller might not exist), we can ensure that the state trajectory $x(t)$ of the plant, satisfies the following bound ([29])

$$|x(t) - x_{\text{e}}| \leq \rho(t, |x(0) - x_{\text{e}}|) + \gamma(||x_{\text{ref}}||_\infty + ||u_{\text{ff}}||_\infty),$$

where ρ is a class $\mathcal{KL}$ function and γ is a class $\mathcal{K}_\infty$ function, $t > 0$ refers to time, x_{ref} to the reference trajectory, u_{ff} to the feedforward input, x_{e} denotes the equilibrium, $|\cdot|$ the Euclidean norm, and $||\cdot||_\infty$ the supremum norm.[1] We will ensure that the output of the neural network is bounded by an appropriate design (for example using the hyperbolic tangent as activation function in the last layer). This means that even if we apply our deep iterative controller to an unseen reference trajectory, we can guarantee from the outset that the state trajectory of our system will remain bounded. Such an a-priori bound cannot be easily established if the learning algorithm optimizes over feedback policies.

The remainder of the article will illustrate and evaluate this framework on the task of trajectory tracking with the robot arm shown in Figure 1.

We note that our framework requires the plant to be stable, since, as will be discussed in the following, the ILC optimizes over feedforward trajectories in open loop. However, in case the plant is unstable, it can be prestabilized with a feedback controller and our learning-based IC framework can be applied nonetheless.

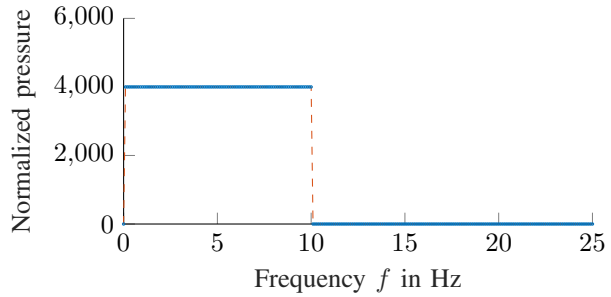
[1]A class $\mathcal{K}_\infty$ function is a function $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$, which is continuous, strictly increasing, unbounded and satisfies $\gamma(0) = 0$. A class $\mathcal{KL}$ function is a function $\rho : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ such that $\rho(\cdot, t) \in \mathcal{K}_\infty$ for each t and $\rho(\zeta, t) \to 0$ for each ζ and for $t \to \infty$.

III. Identification of Linear Baseline Model

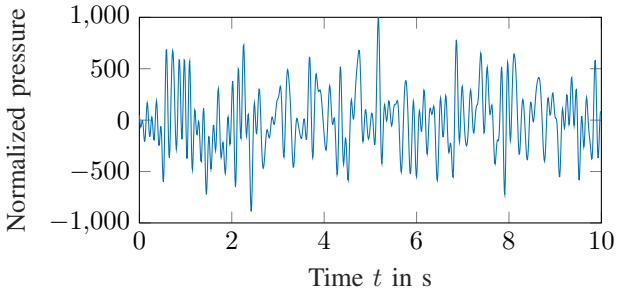
Throughout the article, we decide not to introduce a first-principle model of PAMs, but work with data-driven approaches instead. We will start by identifying a non-parametric model, as this gives us insights about the physical characteristics of our actuation and the robotic arm. In a second step we will also fit a parametric model, which will be used as prior knowledge for our learning-based control approach.

A. Non-parametric Model

We first identify a (dynamic) linear model of each degree of freedom near its operating point. We do the identification in the frequency domain and for simplicity, we regard each joint as an independent single-input single-output system. We use a random-phase multisine signal for exciting the system. By choosing a multisine signal we avoid leakage (due to the periodicity) and excite only the relevant frequencies. The amplitude spectrum is shown in Figure 3a, and excites the frequency lines $0.1\,\text{Hz}, 0.2\,\text{Hz}, \ldots, 10\,\text{Hz}$. The phase is chosen randomly, which allows us to generate excitation signals that evolve differently in the time domain but have the same amplitude spectrum in the frequency domain. We use these different signals to characterize the nonlinearity of the system.



(a) Signal in frequency domain



(b) Signal in time domain

Fig. 3: The figure shows the random multisine signal in the frequency domain and the time domain. We note that in (a) the DC component is set to be zero and only the frequencies $0.1\,\text{Hz}, 0.2\,\text{Hz}, \ldots, 10\,\text{Hz}$ are excited.

We excite each degree of freedom individually. For steering the robot arm we used the application programming interface

developed by Berenz et al. [5]. Each excitation signal will be applied for ten periods continuously, and the results of the first two periods will be discarded in order to eliminate the effect of transients. Let $U^i(j\omega_k)$ and $\Theta^i(j\omega_k),\ i = 1, \ldots, p$ and $k = 1, \ldots, N_\mathrm{f}$ be the discrete Fourier transform (DFT) of the input and output of the i-th period, where p denotes the total number of periods, $j = \sqrt{-1}$ the imaginary number and N_f the number of frequencies. For obtaining a non-parametric transfer function model we first average the input and output signals in the frequency domain over the different periods,

$$\hat{\Theta}\left(j\omega_k\right) = \frac{1}{p}\sum_{i=1}^{p}\Theta^i\left(j\omega_k\right),\ \hat{U}\left(j\omega_k\right) = \frac{1}{p}\sum_{i=1}^{p}U^i\left(j\omega_k\right). \quad (1)$$

The variance of the output signal $\hat{\sigma}_\theta^2\left(j\omega_k\right)$ is given by

$$\frac{1}{p-1}\sum_{i=1}^{p}\left[\Theta^i\left(j\omega_k\right) - \hat{\Theta}\left(j\omega_k\right)\right]\left[\Theta^i\left(j\omega_k\right) - \hat{\Theta}\left(j\omega_k\right)\right]^*, \quad (2)$$

where $(\cdot)^*$ represents complex conjugation, and the average frequency response function (FRF) $\widehat{G}$ and its uncertainty $\hat{\sigma}_\mathrm{n}$ due to measurement noise, see [25], can be obtained as follows:

$$\widehat{G}(j\omega_k) = \frac{\hat{\Theta}(j\omega_k)}{\hat{U}(j\omega_k)},\ \hat{\sigma}_\mathrm{n}^2(k) = \frac{|\widehat{G}(j\omega_k)|^2}{p}\frac{\hat{\sigma}_\theta^2(j\omega_k)}{|\hat{\Theta}(j\omega_k)|^2}. \quad (3)$$

These quantities refer to a single identification experiment with a fixed excitation signal. If the identified system is linear, then applying excitation signals with different phase realizations should not affect the average FRF. Thus, the discrepancy between the measured average FRFs that arise when having excitation signals with different phase distributions, provides a means to quantify the nonlinearities. In our experiments, we designed ten excitation signals with different phase spectra and applied each excitation signal again for ten periods. For each excitation signal, we compute an average FRF and estimate the noise level according to (3), which leads to $\widehat{G}^i$ and $\hat{\sigma}_\mathrm{n}^i$, where the superscript i refers to the different excitation signals.

We then calculate the average FRF over all excitation signals as follows:

$$\widehat{G}_\mathrm{BLA}(j\omega_k) = \frac{1}{l}\sum_{i=1}^{l}\widehat{G}^i(j\omega_k), \quad (4)$$

where l denotes the number of excitation signals (here $l = 10$), and the subscript BLA refers to "best linear approximation".

As discussed, the discrepancy between $\widehat{G}^i$ and $\widehat{G}_\mathrm{BLA}$ gives us an estimation of the nonlinearity of the system:

$$\hat{\sigma}_\mathrm{nl}^2(j\omega_k) = \frac{1}{l(l-1)}\sum_{i=1}^{l}\left|\widehat{G}^i(j\omega_k) - \widehat{G}_\mathrm{BLA}(j\omega_k)\right|^2, \quad (5)$$

which should be compared to the noise level (discrepancy between the different periods):

$$\hat{\sigma}_\mathrm{n}^2(j\omega_k) = \frac{1}{l^2}\sum_{i=1}^{l}\left(\hat{\sigma}_\mathrm{n}^i(j\omega_k)\right)^2. \quad (6)$$

The non-parametric identification results are shown in Figure 4. We note that the noise level is extremely low (signal to noise ratio of about $40\,\mathrm{dB}$), but the uncertainty due to the nonlinearities of the system is relatively high with a signal to noise ratio of $10\,\mathrm{dB}$. This emphasizes once more the need to depart from a linear control framework, as will be discussed in the following sections. The results also highlight the speed of the actuation; the PAMs offer a bandwidth of about $40\,\mathrm{rad\,s^{-1}}$ ($\sim 6\,\mathrm{Hz}$), which is extremely high compared to other actuators. This also shows that our choice of the excitation signal, which excites frequencies below $10\,\mathrm{Hz}$ is reasonable.

B. Parametric Model

To obtain a parametric model for each degree of freedom, we start with the following ansatz:

$$G(s) = \frac{a_n s^n + \cdots + a_0}{s^m + b_{m-1}s^{m-1} + \cdots + b_0}\cdot e^{-TN_\mathrm{d}s}, \quad (7)$$

where $n \in \mathbb{Z}^+$ and $m \in \mathbb{Z}^+$ denote the order of the numerator and denominator of the transfer function, respectively, whereby $m \geqslant n$ is required to ensure causality. The non-negative integer N_d denotes the number of time delays and T is the sampling time (here $T = 0.01\,\mathrm{s}$). All unknown parameters of the numerator and denominator form a parameter vector $\psi^\mathrm{T} = [a_0, \ldots, a_n, b_0, \ldots, b_{m-1}] \in \mathbb{R}^{m+n+1}$, which we determine by solving the following optimization problem

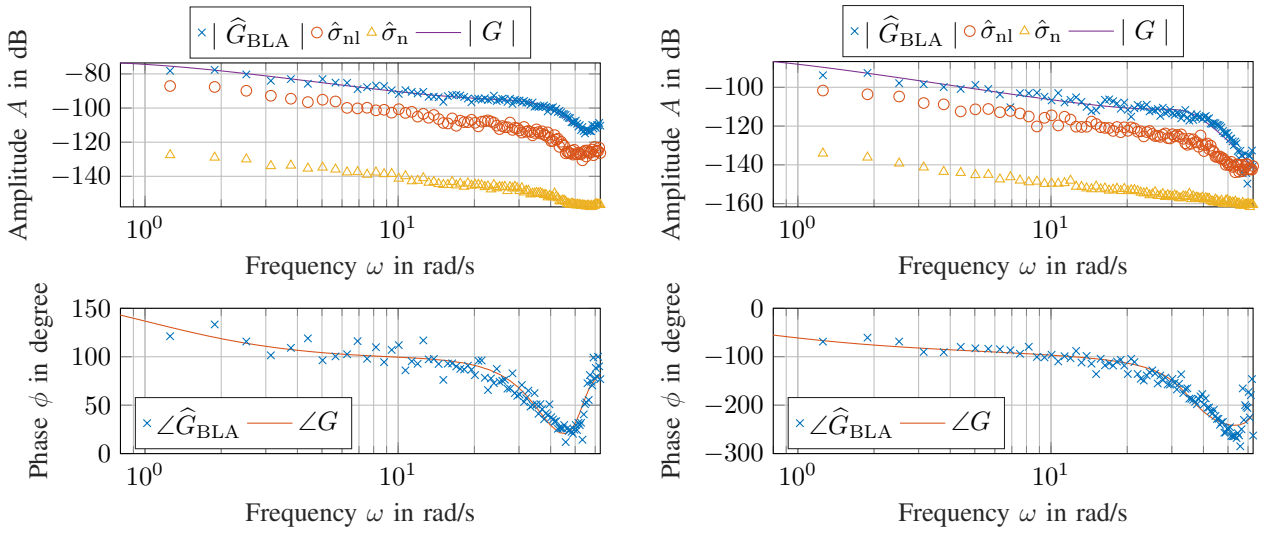
$$\min_{\substack{N_\mathrm{d}\in\mathbb{N}\\ m,n\in\{1,\ldots,5\}}}\ \min_{\psi}\sum_{k=1}^{N_\mathrm{f}}\left|G\left(j\omega_k \mid \psi, N_\mathrm{d}\right) - \widehat{G}_\mathrm{BLA}(j\omega_k)\right|^2. \quad (8)$$

For fixed values of N_d, m and n, we can multiply by the denominator of G, which yields a least squares problem that combines real and imaginary parts. We therefore obtain a solution estimate of (8) by enumerating all possible combinations of N_d, m and n, solving the least squares problem for each, and picking the solution that achieves the lowest cost in (8). We restrict m and n to be below five to avoid overfitting. The results of the parametric model are also shown in Figure 4. From the figure, it can be seen that for both, the amplitude spectrum and the phase spectrum, our parametric transfer function matches the non-parametric estimate well.

IV. Iterative learning control for fixed reference trajectories

A. Motivation

We motivate the introduction of our learning-based IC framework by first discussing the tracking performance that is obtained with traditional linear control methods. Figure 5 shows the tracking result of a traditional two degrees of freedom control design, where we used plant inversion for the feedforward control and a PID controller for the feedback block. The gains for the PID controller have been found by minimizing the $\mathcal{H}_\infty$-norm of the closed-loop sensitivity. Note that manual tuning of the PID gains (e.g. Ziegler Nichols method) leads to similar results. The resulting tracking performance is not satisfactory: Due to the large amounts of



(a) First degree of freedom

(b) Second degree of freedom

Fig. 4: The figures show the identification results. From left to right are the identification results for the first and second degrees of freedom, respectively. Each figure consists of two sub-figures. The upper sub-figure represents the amplitude spectrum and the lower sub-figure represents the phase spectrum. In the amplitude spectrum and the phase spectrum, the measured data are shown by blue crosses, and the fitted results G are shown by solid curves. In the amplitude spectrum, the level of the nonlinearity of the system is shown by circles, and the noise level is shown by triangles.

friction and nonlinear characteristics of PAMs, the accuracy of the linear model is very limited. We will show that with our learning-based IC framework the tracking performance can be improved by orders of magnitudes.

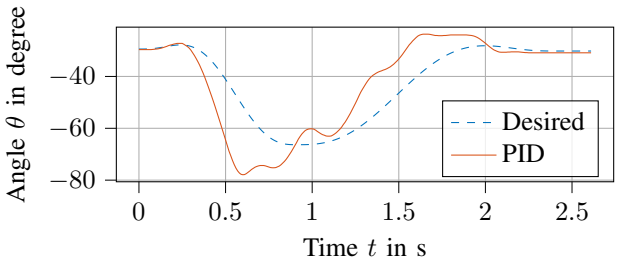


Fig. 5: The figure shows the tracking result of an example trajectory using a linear control framework. The dashed line denotes the fixed reference trajectory, and the solid line the actual trajectory. The tracking accuracy obtained by applying pure feedforward control are even worse.

B. Formulation

Our ILC formulation is inspired by Hofer et al. [15] and will be based on a discrete-time model. For illustration purpose, we will use the discretization of (7) as our model (denoted by G^{d}). In many cases, a nonlinear and more accurate model could be used instead. We will discuss our approach in the case where $n = m = 3$ and $N_{\mathrm{d}} = 2$ in (7). All other cases are treated in a similar manner. By discretizing (7) we obtain the following discrete-time transfer function

$$\Delta\theta(z) = \underbrace{\frac{\alpha_0 + \alpha_1 z^{-1} + \alpha_2 z^{-2}}{1 + \beta_1 z^{-1} + \beta_2 z^{-2}} z^{-2}}_{G^{\mathrm{d}}(z)} u(z), \tag{9}$$

where $\Delta\theta(z) \in \mathbb{C}$ denotes the z-transform of the angle value deviating from the reference posture, and $u(z) \in \mathbb{C}$ represents the z-transform of the normalized input (pressure value) sent to the PAMs. By introducing the state variable $r^{\mathrm{T}}[k] = [\Delta\theta[k-2], \Delta\theta[k-1], u[k-4], u[k-3]]\,, k = 0, \ldots,$ we can rewrite (9) in state-space form:

$$\begin{aligned} r[k+1] &= Ar[k] + b_{\mathrm{u}} u[k-2] + b_{\mathrm{d}} \underbrace{(d[k] + n_{\mathrm{w}}[k])}_{\text{disturbance}}, \\ \Delta\theta[k-1] &= \underbrace{[0\ 1\ 0\ 0]}_{c^{\mathrm{T}}} r[k], \end{aligned} \tag{10}$$

where A and b_{u} are related to the coefficients α_0, α_1, α_2, β_1 and β_2, and the disturbance is assumed to act directly on $\Delta\theta[k]$, which means that b_{d} is equal to c. The model (10) includes additionally the disturbance $(d[k] + n_{\mathrm{w}}[k])$ where $d[k]$ represents the repeatable part (e.g. delays, friction and nonlinearity) and $n_{\mathrm{w}}[k]$ the non-repeatable part (e.g. process noise). The disturbance $d[k]$ is in many cases implicitly dependent on the state r and this dependence could, in principle, be arbitrarily complex (even non-smooth, [27]). The disturbance $d[k]$ also contains interactions between the different degrees of freedom. We can convert (10) into the lifted state-space:

$$\Delta\theta = A_0 r_0 + B_{\mathrm{u}} \tilde{u} + B_{\mathrm{d}} (d + n_{\mathrm{w}})\,, \tag{11}$$

where $r_0 = r[0]$ denotes the initial state, $\Delta\theta \in \mathbb{R}^q$ denotes the outputs of the system in the interval $k \in \{0, 1, \ldots, q-1\}$,

$d \in \mathbb{R}^q$ and $n_\mathrm{w} \in \mathbb{R}^q$ capture the repeatable and non-repeatable disturbances, respectively. The vectors $\Delta\theta$, d, and n_w contain a trajectory of the system of length q, that is, $\Delta\theta^\mathrm{T} = [\Delta\theta[0], \cdots, \Delta\theta[q-1]]$ for example. The input $\tilde{u}^\mathrm{T} = [u[-2], \cdots, u[q-3]] \in \mathbb{R}^q$ takes the time delays into account (here the time delay is two).

The matrices $A_0 \in \mathbb{R}^{q\times 4}$, $B_\mathrm{u} \in \mathbb{R}^{q\times q}$ and $B_\mathrm{d} \in \mathbb{R}^{q\times q}$ in (11) can be expressed as follows:

$$A_0^\mathrm{T} = \left[A^\mathrm{T}c,\ \cdots,\ (A^q)^\mathrm{T}c\right],$$

$$B_\mathrm{u} = \begin{bmatrix} c^\mathrm{T}b_\mathrm{u} & & & & \\ c^\mathrm{T}Ab_\mathrm{u} & c^\mathrm{T}b_\mathrm{u} & & & \\ \vdots & \vdots & \ddots & & \\ c^\mathrm{T}A^{q-2}b_\mathrm{u} & c^\mathrm{T}A^{q-3}b_\mathrm{u} & \cdots & c^\mathrm{T}b_\mathrm{u} & \\ c^\mathrm{T}A^{q-1}b_\mathrm{u} & c^\mathrm{T}A^{q-2}b_\mathrm{u} & \cdots & \cdots & c^\mathrm{T}b_\mathrm{u} \end{bmatrix},$$

$$B_\mathrm{d} = \begin{bmatrix} c^\mathrm{T}b_\mathrm{d} & & & & \\ c^\mathrm{T}Ab_\mathrm{d} & c^\mathrm{T}b_\mathrm{d} & & & \\ \vdots & \vdots & \ddots & & \\ c^\mathrm{T}A^{q-2}b_\mathrm{d} & c^\mathrm{T}A^{q-3}b_\mathrm{d} & \cdots & c^\mathrm{T}b_\mathrm{d} & \\ c^\mathrm{T}A^{q-1}b_\mathrm{d} & c^\mathrm{T}A^{q-2}b_\mathrm{d} & \cdots & \cdots & c^\mathrm{T}b_\mathrm{d} \end{bmatrix}.$$

The iterative learning control algorithm aims at learning the repeatable disturbances d by applying the following principle:

1) Apply the input signal $\tilde{u}$ to the system and record the angle trajectories of the degrees of freedom $1-3$.
2) Update the estimate for the repeatable disturbances d in (10) and/or (11).
3) Update the input signal $\tilde{u}$.

The repeatable disturbances d are learned with a Kalman filter, which is based on the following process equation

$$d^{i+1} = d^i + n_\mathrm{d}^i,\ n_\mathrm{d}^i \sim \mathcal{N}(0, \Pi_\mathrm{d}),\ d^0 \sim \mathcal{N}(0, \Pi), \quad (12)$$

and measurement equation

$$\underbrace{\Delta\theta^i - A_0 r_0^i - B_\mathrm{u}\tilde{u}^i}_{\text{measurement data}} = B_\mathrm{d}d^i + n^i,\ n^i \sim \mathcal{N}(0, \Pi_\mathrm{mix}), \quad (13)$$

where $(\cdot)^i$ denotes the number of ILC iterations. We use the following forms for the variance of n_d^i, the variance of d_0, and the variance of n^i:

$$\Pi_\mathrm{d} = \sigma_\mathrm{d}^2 I,\ \Pi = \sigma^2 I,\ \Pi_\mathrm{mix} = \sigma_\mathrm{w}^2 B_\mathrm{d}B_\mathrm{d}^\mathrm{T} + \sigma_\mathrm{y}^2 I,$$

where $I \in \mathbb{R}^{q\times q}$ denotes the identity matrix, and σ_y^2 the measurement uncertainty. The concrete numerical values for each degree of freedom are shown in Table I.

We then use the mean value of the Kalman filter estimate denoted by $\hat{d}^i$ at the i-th iteration to update the feedforward input $\tilde{u}^{i+1}$ for the next iteration. More precisely, we update $\tilde{u}$ in the following way:

$$\tilde{u}^{i+1} = \arg\min_u \frac{1}{2}\left|\Delta\theta_\mathrm{des} - A_0 r_0 - B_\mathrm{u}u - B_\mathrm{d}\hat{d}^i\right|^2, \quad (14)$$

where $\Delta\theta_\mathrm{des}^\mathrm{T} = [\Delta\theta_\mathrm{des}[0], \cdots, \Delta\theta_\mathrm{des}[q-1]] \in \mathbb{R}^q$ denotes a fixed reference trajectory.

TABLE I: Numerical values of the different variances for each degree of freedom. We note that only the relative magnitudes of the different variances are important not their absolute values. The Kalman filter provides an intuitive way of tuning by trading off measurement uncertainty with process uncertainty.

DoF	σ_d^2	σ^2	σ_y^2
1	10^{-10}	10^{-7}	10^{-2}
2	10^{-10}	10^{-7}	10^{-2}
3	10^{-8}	10^{-7}	10^{-3}

Although there are pressure constraints on the input of the robot arm, we do not consider these when solving the optimization problem in (14). This leads to the following closed-form solution of the optimization problem (14):

$$\tilde{u}^{i+1} = B_\mathrm{u}^\dagger\left(\Delta\theta_\mathrm{des} - A_0 r_0 - B_\mathrm{d}\hat{d}^i\right), \quad (15)$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo inverse.

C. Learning Results

We apply the above ILC method to our robot arm to track the same trajectory as in Figure 5. We conduct 40 learning iterations for ILC, and the results of the last three iterations are shown in Figure 6. From the figure, we can see that there are only minor differences in the results of the last three learning iterations, which indicated that the algorithm has converged. By comparing with the linear control framework we find that ILC can achieve high tracking accuracy even with feedforward control alone: The last iterations consistently result in small steady-state and tracking errors. Compared to the results in Figure 5 there are no oscillations.

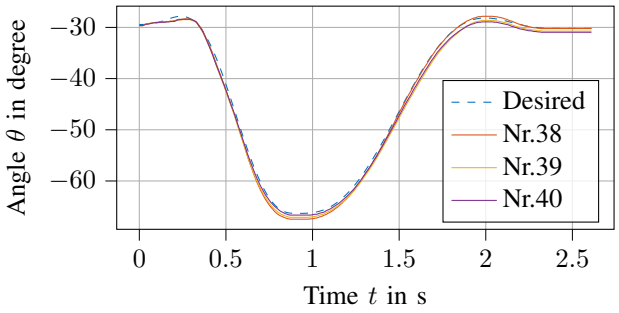


Fig. 6: The figure shows the learning results of ILC. The dashed line is the fixed reference trajectory and the solid lines are the results of the last three learning iterations.

V. Generalizing Iterative Learning Results with Deep Learning

Given the excellent tracking performance of ILC, it is natural to replace the feedforward block in Figure 2 with ILC. However, a fatal flaw of ILC hinders this idea: ILC can only be applied to fixed reference trajectories and requires retraining from scratch if the reference trajectory changes. However, in many applications reference trajectories do frequently change

and retraining from scratch would be infeasible or too time-consuming. This motivates us to develop a new learning-based IC framework based on a CNN. The CNN will generalize the feedforward inputs obtained with ILC, thus enabling tracking of unseen reference trajectories with a similar accuracy.

A. Table Tennis Reference Trajectories

Our final goal is to intercept table tennis balls, and therefore we choose reference trajectories that are typical for the task of intercepting balls that are played to the robot. We select 44 typical ball motions (these are recorded from actual measurements) and for each ball motion, we plan a minimum jerk reference trajectory such that the robot intercepts the ball. We plan the trajectory for the end-effector in a polar coordinate system, which gives us reference trajectories for each joint. These joint trajectories are then tracked by our learning-based approach. Since the position of the end-effector is not affected by the last degree of freedom, and its motion range is very small, the last degree of freedom can be controlled with a simple PID controller. Hence, in this paper we focus on controlling the degrees of freedom $1-3$. We intercept the ball at the highest point after its first collision with the table, since at this point its kinetic energy is minimal. The time duration from the start of the ball's motion to its arrival at the interception point is considered to be the ball's flight time. During this time window, we plan the reference trajectory of the end-effector: From a fixed rest position the robot arm follows a minimum jerk trajectory to the interception point and then returns to the rest position. Figure 7 shows typical reference trajectories and their ILC learning results.

As can be seen from the figures, ILC is a very effective method for trajectory tracking. It is worth emphasizing that these results only rely on feedforward control. Although there is still a small steady error in Figure 7b, this error decreases as the number of training iterations increases. However, some errors cannot be reduced by increasing the number of iterations, which is likely due to the fact that the physical limits are reached.

B. Generalizing with Nonlinear Feedforward Block

As mentioned before, we will use a CNN to generalize the feedforward inputs obtained by ILC for tracking unseen trajectories with similar accuracy. When intercepting the balls that are played to the robot arm, we need to perform the inference in real-time. This motivates us to use a CNN instead of fully connected layers or recurrent neural networks. Moreover, the CNN was also found to be beneficial for handling the coupling between the various degrees of freedom and the temporal correlations. We use a CNN with a simple structure to avoid overfitting since the size of our dataset (44 trajectories) is limited. We divide all trajectories into a training and validation dataset according to the ratio of $7:3$. To speed up the convergence of the neural network and improve accuracy, all inputs are regularized (mean value is subtracted) and all outputs are normalized.

Our neural network consists of six convolutional layers and four fully connected layers. The convolutional layers do not contain any pooling layers, and the fully connected layers do not have a dropout. We use `Tanh` as the activation function for the last layer to ensure that the output is between -1 and 1, and therefore respects the actuation limits of the system. The rest of the layers use `ReLU` as an activation function, which mitigates the risk of gradient disappearance or explosion.

The input of the neural network contains two channels. The first is a window of the given reference trajectory of length $2h+1$ centered at time point k. The second one is the output predicted by a linear regression model, which is also a data segment of length $2h+1$ centered at time point k. We would like the neural network to capture the coupling between all degrees of freedom, so the width of the input is three, one for each degree of freedom. The output of the network is the feedforward command u_{ff} (see block diagram) at time point k. We train an individual neural network for each degree of freedom, where each network has the same structure.

The approach for generating the input data to the CNN is shown in Figure 8. The same approach is used to generate the first and second input channels.

We use the prediction results of a linear regression model as the second input channel of the neural network, since we found in experiments that this improves prediction slightly. The linear prediction model is obtained from the same training data as the neural network, and the input is processed in very similar ways (see Figure 9 for a graphical sketch). The difference is that we stack the data from each degree of freedom vertically instead of horizontally, and introduce a bias term.

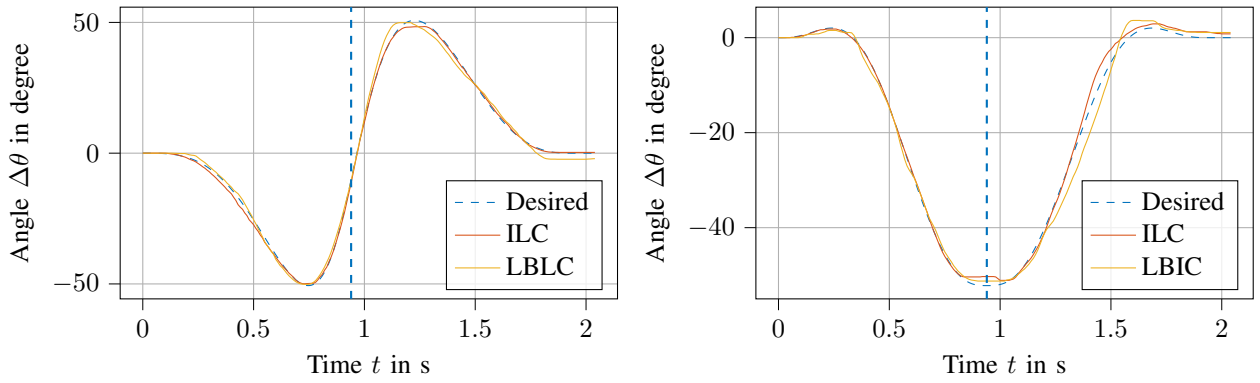
The parameters for the linear regression model are obtained by solving:

$$\eta_\tau = \Xi^\dagger u_\tau,\ \tau = 1,2,3, \tag{16}$$

where $\eta_\tau \in \mathbb{R}^{6h+4}$ is the parameter vector for the linear regression model which includes a bias term, and $u_\tau \in \mathbb{R}^{N_\text{t}}$ contains the training data (obtained by ILC). The subscript τ denotes the index of the degrees of freedom since a linear regression model is generated for each degree of freedom separately. The dimension N_t represents the number of data points used for training the CNN, and the matrix $\Xi \in \mathbb{R}^{N_\text{t}\times(6h+4)}$ is a stack of different data points as shown in Figure 9. We experimented with different choices of h and found that a value of $h = 100$ yields good results. We note that $h = 100$ implies that the CNN operates on a window of length $2\,\text{s}$, meaning that it can take future values of the reference trajectories (up to $1\,\text{s}$) into account.

C. Generalizing Results

We use the neural network architecture from the previous subsection and train it on 31 ball trajectories. We then use the CNN as the feedforward block in Figure 2 and implement the resulting two degrees of freedom controller. We retain the same feedback controller as in Section III, apply the learning-based IC framework to our robot arm and evaluate the tracking



(a) First degree of freedom

(b) Second degree of freedom

Fig. 7: The figure shows the results of the last iteration of ILC and learning-based IC framework. The fixed reference trajectories $\Delta\theta_{\text{des}}$ are shown in dashed curves. The results of the last iteration $\Delta\theta$ obtained by different methods are shown in solid curves. "ILC" denotes the results of traditional ILC method and "LBIC" the learning-based IC framework. The hitting time point is shown in a vertical dashed line. Both trajectories are from the validation dataset. We note that the movement range is large ($+/-50°$ in the first degree of freedom and $-40°$ in the second degree of freedom) and that the motion is dynamic reaching $5\,\mathrm{m\,s^{-1}}$ at the interception point. The right sub-figure shows the small steady-errors.

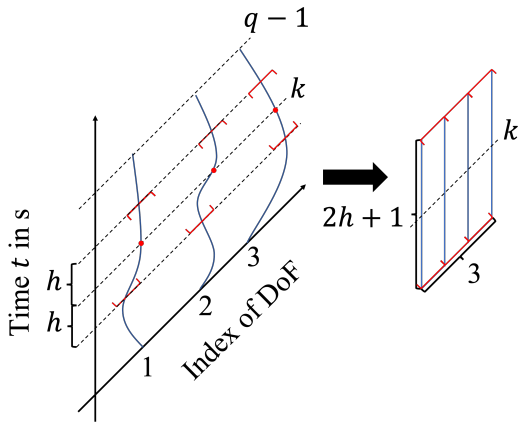


Fig. 8: The figure shows the approach for generating the input data for CNN. We use a window of length $2h+1$ at time point k. The input data for the CNN has two channels, the first channel contains the desired trajectories for different degrees of freedom and the second channel is generated from the prediction results of a linear regression model. We emphasize that the length of the reference trajectory is not necessarily fixed.

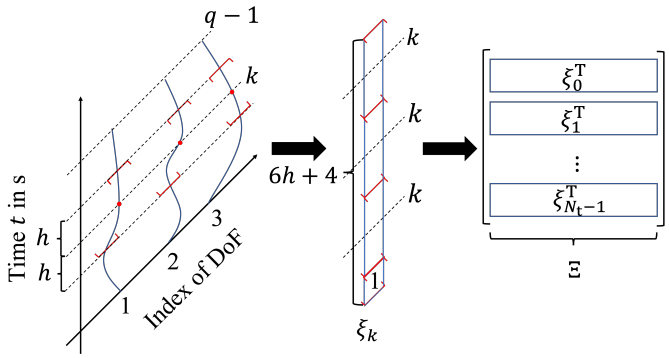


Fig. 9: The figure shows the method for generating the input data for the linear regression model. We also use a window of length $2h+1$ at each point in time and for each degree of freedom and introduce a bias term (the last element of ξ_k is set to 1). Finally, we stack all the segments ξ_k^{T} to form the data matrix Ξ used in the linear regression model.

performance on all trajectories. We use the root-mean squared tracking error of the end-effector as our performance metric:

$$\delta_i = \frac{1}{q_i} \sum_{k=0}^{q_i} \left| e_i^\star[k] - e_i[k] \right|, \tag{17}$$

where $e_i^\star[k] \in \mathbb{R}^3$ and $e_i[k] \in \mathbb{R}^3$ denote the i-th reference trajectory and actual trajectory at time point k, respectively, $i = 1, \ldots, 44$.

The values of δ_i for the different trajectories are shown in Figure 10. As shown in the figure, when relying only on feedforward control, the tracking accuracy of the neural network is worse than ILC even though the neural network has excellent prediction results (there is almost no performance difference between training and test trajectories). Therefore, we introduce feedback to compensate for initial errors and non-repeatable disturbances, which results in the two degrees of freedom structure shown in Figure 2. The final tracking accuracy is even better than ILC, reaching an average error of under $0.04\,\mathrm{m}$.

In Figure 7 we compare the tracking results of the learning-based IC framework and ILC. It is worth mentioning that both trajectories are from the validation dataset. As can be seen from the figure, the generalization results of the learning-based IC framework are very close to the tracking results of ILC. It is also worth noting that the average root-mean squared tracking error over all ball trajectories obtained by the learning-based

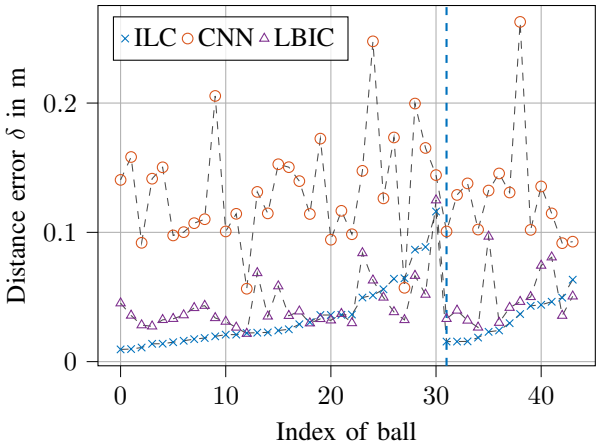


Fig. 10: The figure shows the average Euclidean distance to the desired trajectory in Cartesian space of all ball trajectories with different methods. The left side of the dashed line is the index of the training trajectories and the right side is the index of the test trajectories. The index of ball is sorted in ascending order according to the results of ILC on the training dataset and validation dataset respectively

IC framework is better than the learning results of ILC in both the training and validation datasets, see Figure 10, which indicates that our learning-based IC framework successfully generalizes the results from ILC.

VI. Conclusion

In summary, we conclude that by identifying a linear dynamical model of the system near the operating point and then combining ILC with deep learning techniques, we can achieve outstanding tracking performance even on very complex systems and when performing dynamic motions. We found that ILC converges even when the initial linear model is a poor approximation of the true underlying dynamics. However, a more accurate model is likely to reduce the iterations needed for ILC to converge.

The traditional ILC framework can only accurately track fixed reference trajectories and requires retraining from scratch if the reference changes. In contrast, our learning-based IC framework generalizes the results and enables accurate tracking of previously unseen trajectories. We also found that including a linear regression model as input to the CNN can effectively improve the convergence speed and accuracy of the latter. Finally, we have shown that our two degrees of freedom approach with the CNN as feedforward controller results in excellent tracking results on the real robot (see also https://youtu.be/kR9jowEH7PY).

In the future, we aim at collecting a larger dataset, which might enable us to improve the performance further. We would also like to use some of the ILC data to quantify the uncertainty of our disturbances estimates.

Acknowledgments

We thank Vincent Berenz and Simon Guist for numerous fruitful discussions and for providing excellent software and hardware support. Hao Ma and Michael Muehlebach thank the German Research Foundation, the Branco Weiss Fellowship, administered by ETH Zurich, and the Center for Learning Systems for the generous support.

References

[1] H.-S. Ahn, Y. Chen, and K. L. Moore. Iterative Learning Control: Brief Survey and Categorization. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(6):1099–1121, 2007.

[2] G. Andrikopoulos, G. Nikolakopoulos, and S. Manesis. A Survey on Applications of Pneumatic Artificial Muscles. In *Proceedings of the Mediterranean Conference on Control Automation*, pages 1439–1446, 2011.

[3] S. Arimoto, S. Kawamura, and F. Miyazaki. Convergence, Stability and Robustness of Learning Control Schemes for Robot Manipulators. In *Proceedings of the International Symposium on Robot Manipulators on Recent Trends in Robotics: Modeling, Control and Education*, pages 307–316, 1986.

[4] D. X. Ba, T. Q. Dinh, and K. K. Ahn. An Integrated Intelligent Nonlinear Control Method for a Pneumatic Artificial Muscle. *Transactions on Mechatronics*, 21(4): 1835–1845, 2016.

[5] V. Berenz, M. Naveau, F. Widmaier, M. Wüthrich, J.-C. Passy, S. Guist, and D. Büchler. The o80 c++ templated toolbox: Designing customized python apis for synchronizing realtime processes. *Journal of Open Source Software*, 6(66):2752, 2021.

[6] D. Bristow, M. Tharayil, and A. Alleyne. A Survey of Iterative Learning Control. *IEEE Control Systems Magazine*, 26(3):96–114, 2006.

[7] G. Brown, R. Haggard, R. Almassy, R. Benney, and S. Dellicker. The Affordable Guided Airdrop System (AGAS). In *Proceedings of the Aerodynamic Decelerator Systems Technology Conference*, pages 316–325. 1999.

[8] D. Büchler, H. Ott, and J. Peters. A Lightweight Robotic Arm with Pneumatic Muscles for Robot Learning. In *Proceedings of the International Conference on Robotics and Automation*, pages 4086–4092, 2016.

[9] D. Büchler, R. Calandra, and J. Peters. Learning to Control Highly Accelerated Ballistic Movements on Muscular Robots. *arXiv:1904.03665 [cs]*, 2019.

[10] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters. Learning to Play Table Tennis From Scratch Using Muscular Robots. *IEEE Transactions on Robotics*, 2020.

[11] D. Caldwell, G. Medrano-Cerda, and M. Goodwin. Braided Pneumatic Actuator Control of a Multi-Jointed Manipulator. In *Proceedings of IEEE Systems Man and Cybernetics Conference*, pages 423–428, 1993.

[12] Z. Chen, X. Liang, and M. Zheng. Deep Iterative Learning Control for Quadrotor's Trajectory Tracking. In

Proceedings of the American Control Conference, pages 1408–1413, 2021.

[13] S. Ganguly, A. Garg, A. Pasricha, and S. K. Dwivedy. Control of Pneumatic Artificial Muscle System through Experimental Modelling. *Mechatronics*, 22(8):1135–1147, 2012.

[14] M. Hofer and R. D'Andrea. Design, Modeling and Control of a Soft Robotic Arm. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1456–1463, 2018.

[15] M. Hofer, L. Spannagl, and R. D'Andrea. Iterative Learning Control for Fast and Accurate Position Tracking with an Articulated Soft Robotic Arm. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 6602–6607, 2019.

[16] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning Agile and Dynamic Motor Skills for Legged Robots. *Science Robotics*, pages 1–13, 2019.

[17] Y. Jian, D. Huang, J. Liu, and D. Min. High-Precision Tracking of Piezoelectric Actuator Using Iterative Learning Control and Direct Inverse Compensation of Hysteresis. *IEEE Transactions on Industrial Electronics*, 66(1): 368–377, 2019.

[18] K. Kogiso, K. Sawano, T. Itto, and K. Sugimoto. Identification Procedure for McKibben Pneumatic Artificial Muscle Systems. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3714–3721, 2012.

[19] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. *Science Robotics*, pages 1–13, 2020.

[20] S. Levine and V. Koltun. Guided Policy Search. In *Proceedings of the International Conference on Machine Learning*, pages 1–9, 2013.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-End Training of Deep Visuomotor Policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[22] J. J. Misuraca and C. Mavroidis. Lower Limb Human Muscle Enhancer. In *Dynamic Systems and Control*, pages 963–969, 2001.

[23] F. L. Mueller, A. P. Schoellig, and R. D'Andrea. Iterative Learning of Feed-Forward Corrections for High-Performance Tracking. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 3276–3281, 2012.

[24] V. L. Nickel, J. Perry, and A. L. Garrett. Development of Useful Function in the Severely Paralyzed Hand. *Journal of Bone and Joint Surgery*, 45(5):933–952, 1963.

[25] R. Pintelon and J. Schoukens. *System Identification: A Frequency Domain Approach*. 2012.

[26] S. D. Prior and A. S. White. Measurements and Simulation of a Pneumatic Muscle Actuator for a Rehabilitation Robot. *Simulation Practice and Theory*, 3(2):81–117, 1995.

[27] R. Quintanilla and J. T. Wen. Iterative Learning Control for Nonsmooth Dynamical Systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 245–251, 2007.

[28] A. P. Schoellig, F. L. Mueller, and R. D'Andrea. Optimization-Based Iterative Learning for Precise Quadrocopter Trajectory Tracking. *Autonomous Robots*, 33(1):103–127, 2012.

[29] E. D. Sontag. Input to state stability: Basic concepts and results. *Nonlinear and Optimal Control Theory*, pages 163–213, 2008.

[30] B. Tondu and S. Zagal. McKibben Artificial Muscle Can Be in Accordance with the Hill Skeletal Muscle Model. In *Proceedings of the International Conference on Biomedical Robotics and Biomechatronics*, pages 714–720, 2006.

[31] M. van Damme, F. Daerden, and D. Lefeber. A Pneumatic Manipulator Used in Direct Contact with an Operator. In *Proceedings of the International Conference on Robotics and Automation*, pages 4494–4499, 2005.

[32] N. Wereley, C. Kothera, E. Bubert, B. Woods, M. Gentry, and R. Vocke. Pneumatic Artificial Muscles for Aerospace Applications. In *Proceedings of the ASC Structures, Structural Dynamics, and Materials Conference*. 2009.

[33] Y. M. Zhao, Y. Lin, F. Xi, and S. Guo. Calibration-Based Iterative Learning Control for Path Tracking of Industrial Robots. *IEEE Transactions on Industrial Electronics*, 62 (5):2921–2929, 2015.

[34] J. Zughaibi, M. Hofer, and R. D'Andrea. A Fast and Reliable Pick-and-Place Application with a Spherical Soft Robotic Arm. *Proceedings of the International Conference on Soft Robotics*, pages 599–606, 2021.