Interpolation and neural network based iterative learning control for coping with new references without relearning

Erens, G.P.N.

Award date:
2021

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Final Report of the Graduation Project

Interpolation and neural network based iterative learning control for coping with new references without relearning

Master:	Systems & Control
Department:	Electrical Engineering
Research Group:	Control Systems
Student:	G.P.N. Erens
Identitiy Number:	0997906
Thesis Supervisors:	Dr. M. Lazar
	Prof. Dr. H. Butler
Coach:	MSc. M. Bolderman
Date:	August 31, 2021

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date 29/08/2021

Name Gerben Prins Nicolaas Erens

ID-number 0997906

Signature 

Insert this document in your Master Thesis report (2nd page) and submit it on Sharepoint

[i] See: http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU

Contents

Abstract - **Linear motors are widely used actuators for high precision systems. This high precision is achieved through a combination of feedback and feedforward control. Iterative Learning Control (ILC) is a form of feedforward control that can achieve a superior precision on a repeating reference trajectory. In this work we investigate three approaches to use learned ILC sequences to obtain a general feedforward controller for reference trajectories different from the learned trajectories. Of the three approaches considered in this work, two focus on interpolating the input sequences learned through ILC. The first scales the learned input sequences and the reference trajectories in time, and then interpolates the time-scaled input sequences based on their reference trajectories and the target reference trajectory. The second interpolation approach splits a reference trajectory into segments that can be described by a polynomial and then interpolates the learned input sequences based on the polynomial coefficients of the reference trajectory. In the last approach feedforward neural networks are used to approximate the relationship between the reference trajectory values and the input sequence. On a motor model with a small nonlinear disturbance the interpolation of time-phase scaled signals achieves a** 99.4 % **tracking error suppression on average with respect to mass-acceleration feedforward. The interpolation based on polynomial coefficients achieves a** 94.3 % **tracking error reduction. Lastly the neural network approach achieves a tracking error reduction of** 99.5 % **These are all favorable results and a close approximation of the near** 100 % **tracking error reduction that is obtained through ILC.**

I. Introduction

Linear motors are high precision actuators mainly used in various semiconductor fabrication and inspection processes [1][2]. The main advantages of linear motors over their rotary counterpart are that they have no mechanical limitation on velocity and acceleration in the form of gears or belts and that their mechanical simplicity results in higher reliability, longer lifetime, less wear, and less maintenance required.

To achieve precise tracking with linear motors a combination of feedback and feedforward control is used. The feedback controller compensates for the disturbances encountered during the operation of the motor and the feedforward controller compensates for the reference and disturbances that can be calculated before it affects the output, which significantly improves performance when compared to only feedback control [3]. In this work we will look at a specific strategy for feedforward control: Iterative Learning Control (ILC). ILC achieves superior performance by iteratively updating an input sequence for a repeating reference trajectory. In doing so ILC can achieve high performance despite large model uncertainties and repeating disturbances [4][5].

While an input sequence obtained using ILC can result in a low tracking error on the reference it was learned for, this input sequence does not work for any other reference trajectories. Not only does it not provide the inputs necessary to track the new reference, it still provides the inputs required for an old reference trajectory, resulting in a feedforward input sequence that is detrimental to performance [6]. For this reason anytime the reference changes, the ILC should be reinitiated and relearned. This is an expensive and time-consuming process during which the plant can not operate. Therefore we aim to formulate a general feedforward controller for a range of references, adapting the ILC generated feedforward sequences as simple and as fast as possible.

Interpolation between ILC learned input sequences has been done already in [7]. Here the interpolation is done between learned inputs based on a slowly changing model parameter. The choice of interpolation coefficients bears similarities to gain scheduling control, with common options given in [8].

A different interpolation has been done in [9], where the learned ILC input sequence is assigned to a period of constant jerk. By shifting this period in time and flipping the sign of the input a set of third order reference trajectories could be made.

Instead of directly interpolating the data a set of basis function can also be used. Using an adaptation of the norm optimal ILC algorithm the interpolation coefficients of the basis functions are learned instead of the input sequence [10]. Interesting choices for basis functions are rational basis functions [11] and FIR models [9], which both approximate a model inverse using the ILC algorithm.

A different idea is to approximate the relationship between the reference trajectory and the feedforward input sequence obtained through ILC. Neural networks are known to be good approximators [12] and have been used in control before [13][14]. These approaches have also been linked to ILC. In [15] a neural network was used to approximate the relation between a reference trajectory and the ILC obtained inputs. In [16] an ILC like structure was used, where a neural network was used as a model to predict the error in the next time step, and a different neural network was used to calculate a feedforward input based on the predicted error.

In this work we develop two methods to interpolate between input sequences learned through ILC by explointing knowledge about the reference trajectory.

The third approach is to train a neural network to approximate the input sequence obtained by using ILC based on the reference trajectory is implemented. This approach is taken from [15], because it is an interesting approach that has been succesfully implemented before, and should provide a fair benchmark to compare the other methods against.

The structure of this thesis is as follows. In section II brief preliminaries about iterative learning control are

Table I: Parameter values used in the simulation of the linear motor model.

m	20 kg
f_v	135.75 $^{Ns}/_{m}$

discussed. The problem, using iteartive learning control on different references without relearning, is formulated in section III. In section IV we present two different approaches using interpolation to construct input sequences for different references. In section V we present a neural network as function approaximator to construct the input sequences. The three presented methods are compared against eachother in section VI. The conclusion and recommendation is summarized in section VII.

II. Preliminaries

This section provides the necessary knowledge about the type of system used, the reference trajectory design, and iterative learning control.

A. System dynamics and control structure

The methodology introduced is valid for linear systems subject to partially known and partially unknown nonlinear disturbances. As a guiding example we consider a coreless linear motor as shown in Figure 1, where known disturbances are friction forces, and unknown ones are parasitic forces due to electromagnetics. A simplified dynamical model is obtained via Newton's second law of motion as

$$\ddot{y}(t) = \frac{1}{m}\left(F_u - f_v\dot{y}(t)\right), \tag{1}$$

where m is the identified mass, and f_v is the viscous friction coefficient. The values used for these parameters is given in Table I. F_u is the actual input force identified as a function of the desired input u and the position y

$$F_u(u, y) = \alpha(y)u + \beta(y). \tag{2}$$

The actual force is different from the desired force due to force ripple, and is caused by imperfection of the commutation algorithm. This effect is modelled using harmonic series. For details on the modelling process and $\alpha(y)$ and $\beta(y)$ see [17].

This system is controlled in closed-loop by the feedback controller

$$C_{fb}(s) = \frac{320s^2 + 6912s + 2.388 \times 10^4}{3.029 \times 10^{-6}s^3 + 1.658 \times 10^{-3}s^2 + 0.2315}, \tag{3}$$

designed in [17]. This controller results in a bandwidth of 10.7 Hz. The full control scheme is shown in Figure 2. A zero-order hold is present before the system model P_{ME}, and a sampler is used to obtain the output, but these are omitted for brevity. The controller is discretized using zero-order hold and a sampling time $T_s = 10^{-3}$ s.

The control loop tries to make the system output y track a reference trajectory r. To do this, the feedback controller C_{fb} calculates a feedback input u_{fb} based on the error signal e, and an eventual feedforward controller C_{ff} calculates a feedforward input u_{ff}. Based on these inputs and possible disturbances d the system moves, resulting in an output position y.

In this work a mass acceleration (MA) feedforward controller is implemented as a base line feedforward controller. The feedforward input u_{ff} at time t is calculated as

$$u_{ff}(t) = m\, a(t), \tag{4}$$

where m is the identified mass of the plant, and $a(t)$ is the acceleration of the reference trajectory. The value of m is the same as the one used in the model, and given in Table I. An example of the resulting tracking error when using the mass acceleration feedforward is shown in Figure 3.

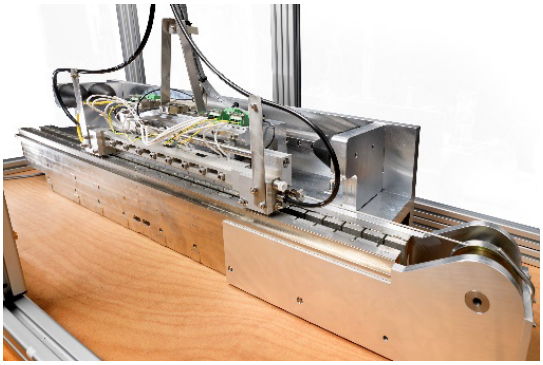

Figure 1: Picture of the coreless linear motor considered in this work.

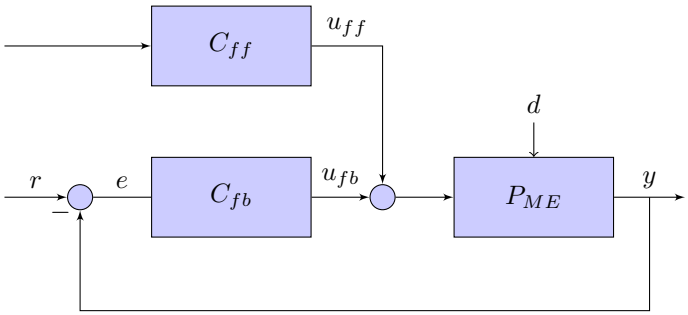

Figure 2: A basic negative feedback control loop with feedforward control.

B. Reference trajectory design

The reference trajectories used in this project are third-order reference trajectories satisfying velocity, acceleration, and jerk constraints [18]. The reference trajectory consists of a back and forth motion of 0.1 m. An example of the reference trajectories used is shown in Figure 6. Since the position step is equal in all reference trajectories, each reference trajectory $\mathbf{r}$ is fully characterized by parameters p_1, p_2, and p_3, describing maximum jerk, maximum acceleration, and maximum velocity respectively. This approach is in principle applicable for any finite number of parameters. Such that $\mathbf{r} = \pi(p_1, p_2, p_3)$, with $\pi : \mathbb{R}^3 \to \mathbb{R}^N$, and N is the amount of samples in

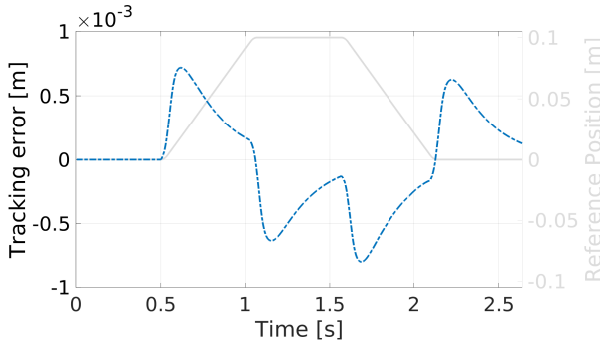

Figure 3: Tracking error obtained using mass acceleration feedforward

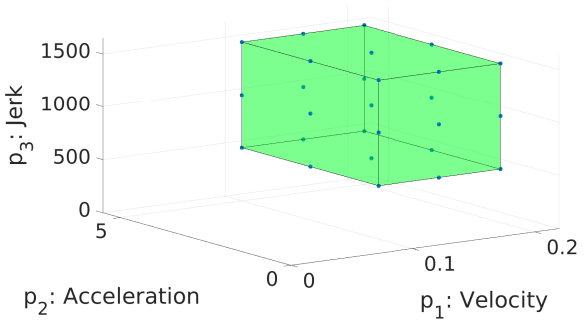

Figure 4: The reference parameters of R shown in space $\mathcal{P}$.

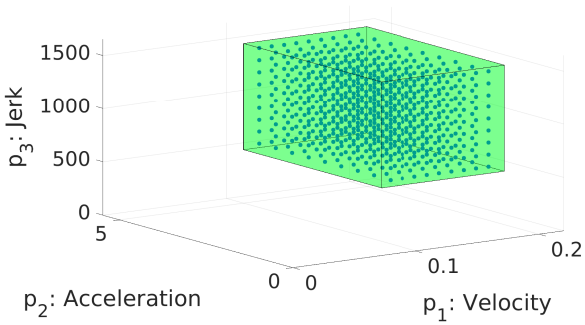

Figure 5: The reference parameters of T shown in space $\mathcal{P}$.

a reference trajectory. A parametrized space $\mathcal{P} \subset \mathbb{R}^3$ is defined as $\mathcal{P} : \prod_i^3 [\mathcal{L}_i, \mathcal{U}_i]$ where $\mathcal{L}_i, \mathcal{U}_i \subset \mathbb{R}$ with $i = 1, 2, 3$ are the lower and upper bounds on parameter p_i, describing all reference trajectories of interest. The values of the lower and upper bounds used in this work are given in Table II. We define two sets of references, $R, T \in \mathcal{P}$, which are respectively the set of original references $\mathbf{r}_o$ used to obtain input sequences via ILC, and the set of target references $\mathbf{r}_t$ used to test the effectiveness of the approaches:

$$R = \{\mathbf{r}_{o,1}, \cdots, \mathbf{r}_{o,n_R}\}, \tag{5}$$

$$T = \{\mathbf{r}_{t,1}, \cdots, \mathbf{r}_{t,n_T}\}. \tag{6}$$

In (5) R consists of 27 references, with their parameters shown in Figure 4. T in (6) consists of 729 reference trajectories in $\mathcal{P}$, as shown in Figure 5.

Table II: Lower and upper bounds on reference parameters.

Parameter	Lower bound $\mathcal{L}$	Upper bound $\mathcal{U}$
Velocity	0.1 $^m/_s$	0.2 $^m/_s$
Acceleration	1 $^m/_{s^2}$	5 $^m/_{s^2}$
Jerk	500 $^m/_{s^3}$	1500 $^m/_{s^3}$

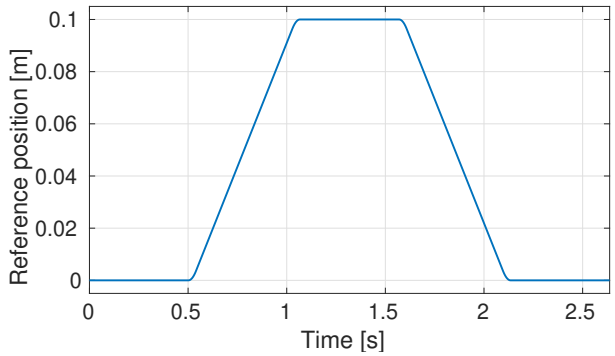

Figure 6: Example of a reference trajectory used to visuallize the results of the different approaches.

C. Generating input sequences using ILC

The basic ILC structure is shown in Figure 7. The system is controlled by a feedback controller C_{fb} and a feedforward input sequence $\mathbf{u_{ff}}$ stored in memory. For an iteration of a reference trajectory, all tracking errors will be stored in memory and will be used after the iteration to update $\mathbf{u_{ff}}$ for the next iteration.

The ILC approach used in this project is model inverse ILC [19]. This approach is more reliable to tune using the inverse of the linear model, than the manually tuning of a P(I)D-type ILC. This approach is less memory intensive than a design based on the lifted system form such as quadratically optimal ILC, which uses matrixes of size $N \times N$ for both the system model and the cost function [4].

In this case the update rule for the input sequence u_{ff} is given as

$$u_{ff,k+1} = Q\left(u_{ff,k} + Le_k\right), \tag{7}$$

where k is the iteration index, e_k is the tracking error, L is a learning filter, and Q is a robustness filter. Assuming the reference $r_k = 0$, we can use the transfer function from feedforward input to tracking error of a system P_{ME} controlled in closed-loop by controller C_{fb},

$$e_k = \frac{-P_{ME}}{1 + P_{ME}C_{fb}} u_{ff,k}, \tag{8}$$

which is the negative of the process sensitivity. We can

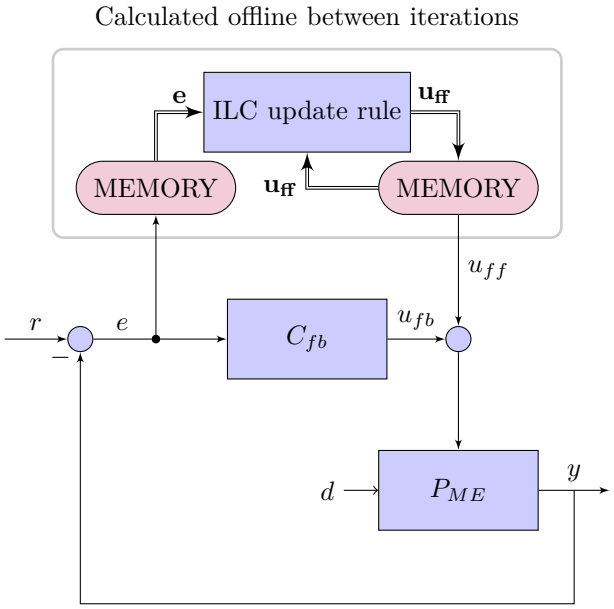

Figure 7: Control loop structure when using ILC as a feedforward controller.

elimite u_{ff}:

$$e_{k+1} = \frac{-P_{ME}}{1 + P_{ME}C_{fb}} u_{ff,k+1}, \tag{9}$$

$$= \frac{-P_{ME}}{1 + P_{ME}C_{fb}} Q\left(u_{ff,k} + Le_k\right), \tag{10}$$

$$= Q\left(1 + \frac{-P_{ME}}{1 + P_{ME}C_{fb}} L\right) e_k. \tag{11}$$

This shows the propagation of the error signal from run to run. Convergence will take place if:

$$\left\| Q\left(1 - \frac{P_{ME}}{1 + P_{ME}C_{fb}} L\right) \right\|_\infty < 1. \tag{12}$$

This makes the inverse of the process sensitivity $L = \frac{1+P_{ME}C_{fb}}{P_{ME}}$ seem like a good candidate. Since the process sensitivity is normally strictly causal, this inverse is non-causal. This is not a problem, since the whole error signal is known already, so it can be evaluated noncausally. If the process sensitivity contains non-minimum phase zeros there is a problem, since the inverse will be unstable. In that case a stable approximate inverse is used, such as the ZPETC or ZMETC [20][21]. The robustness filter Q is designed such that the inequality (12) is satisfied even with approximate inverse, and such that high frequent noise is filtered out of the input. Non repeating disturbances can be handled by adding a scaling factor the learning filter $L = \alpha\frac{1+P_{ME}C_{fb}}{P_{ME}}$, where $\alpha \in (0, 1]$. This results in inputs that are based more on the average error over multiple iterations, with a higher α learning more quickly, and a lower α resulting in a better averaging and less effect of nonrepeating disturbances.

In this work the design was as follows:

- The discretized version of the plant

$$P_{ME}(z^{-1}) = \frac{2.494 \times 10^{-8} z^{-1} + 2.489 \times 10^{-8} z^{-2}}{1 - 1.993 z^{-1} + 0.9932 z^{-2}}$$

- The discretized version of the feedback controller

$$C_{fb}(z^{-1}) = \frac{\begin{array}{r} 8.128 \times 10^{4} z^{-1} - 1.608 \times 10^{5} z^{-2} \\ + 7.954 \times 10^{4} z^{-3} \end{array}}{1 - 2.52 z^{-1} + 2.098 z^{-2} - 0.5785 z^{-3}}$$

- $\alpha = 0.8$
- Q is a second order butterworth lowpass filter with a breaking frequency of 300 Hz

$$Q(z^{-1}) = \frac{0.3913 + 0.7827 z^{-1} + 0.3913 z^{-2}}{1 + 0.3695 z^{-1} + 0.1958 z^{-2}}$$

- The process sensitivity $\frac{P_{ME}}{1+P_{ME}C_{fb}}$ is nonminimum phase, making the resulting learning filter stable. Thus it was not neccesary to use a stable approximation of the inverse.

This method of ILC is used on all reference trajectories in R and T for 20 iterations, converging to an input sequence. These input sequences are used to construct 2 sets. The set of input sequences learned for the original references:

$$U_R = \{\mathbf{u}^*_{o,1}, \cdots, \mathbf{u}^*_{o,n_R}\}, \tag{13}$$

which is used to generate new input sequences. The set of input sequences learned for the target references:

$$U_T = \{\mathbf{u}^*_{t,1}, \cdots, \mathbf{u}^*_{t,n_T}\}, \tag{14}$$

which is only used to validate the methods used, and is not used by the methods as a basis to generate new input sequences from.

The input sequence obtained for the reference shown in Figure 6 is plotted to visualize a typical result in Figure 8. Applying this input to the system model resulted in the tracking error shown in Figure 9. The maximum tracking error during the motion is 4.54×10^{-8} m, which is significantly smaller than the maximum tracking error with mass acceleration feedforward of 8.00×10^{-4} m.

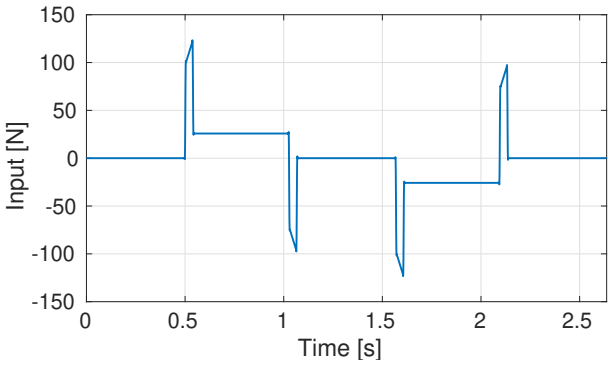

Figure 8: Input sequence learned using ILC on a reference trajectory.

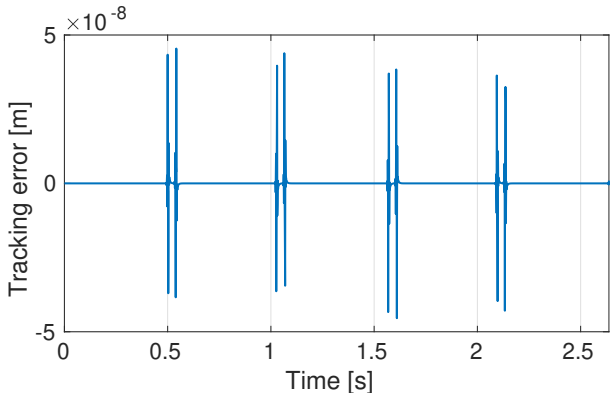

Figure 9: Resulting tracking error when using the input sequence obtained through ILC.

D. Changing the reference trajectory while using ILC

ILC is a very powerful control technique, generating inputs for close to perfect tracking in a few iterations, but it does come with a big drawback: if your reference trajectory changes, the learned feedforward input sequence is not correct for the new reference trajectory. An example of this is given for the situation in which the reference changes to one with a higher constant velocity as in Figure 10. Figure 11 shows that using an input sequence obtained with ILC on a different reference trajectory can result in a higher average tracking error, than not using feedforward control at all.

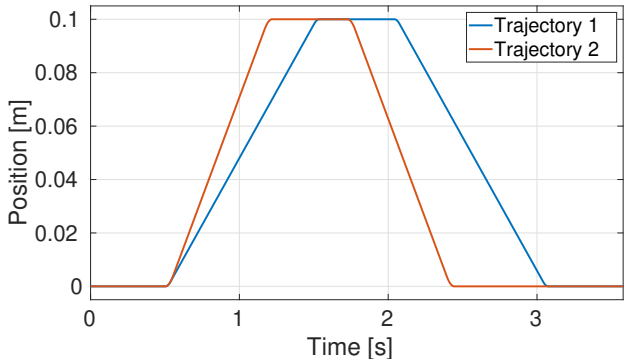

Figure 10: The two references used for visializing the effect of changing references on ILC.

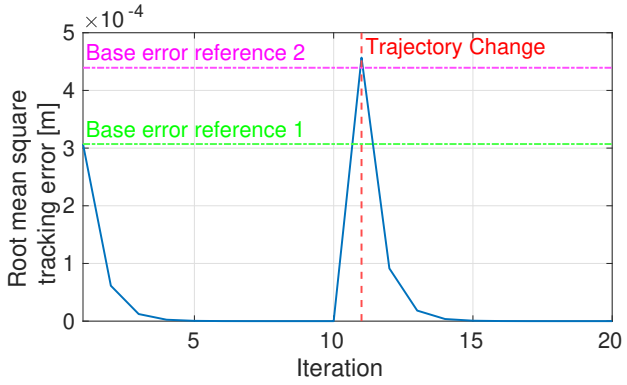

Figure 11: Error over iterations when changing the reference trajectory.

III. Problem formulation

While using an ILC as feedforward controller results in near-perfect tracking on a learned reference trajectory it is not beneficial for slightly different reference trajectories. Obtaining an ILC input sequence on every possible reference trajectory in parameter space $\mathcal{P}$ is not a viable approach, but it would still be desirable to have ILC like performance for all reference trajectories in $\mathcal{P}$. Therefore investigating a method to construct an input sequence resulting in ILC like performance for references in $\mathcal{P}$ based on a finite amount of input sequences obtained through ILC on references in $\mathcal{P}$ would be interesting. This results in the main problem considered in this thesis.

Problem 1 (Find a generalized feedforward controller): Construct a function $f : (\mathbb{R}^{N\times 1}, \mathbb{R}^{N\times n_R}, \mathbb{R}^{N\times n_R}) \to \mathbb{R}^{N\times 1}$ such that

$$\hat{\mathbf{u}} = f(\mathbf{r_t}, R, U_R), \tag{15}$$

which should generate a feedforward input sequence $\hat{u}$ for a target reference $\mathbf{r_t} \in \mathcal{P}$ based on learned ILC inputs U_R and the corresponding reference trajectories in R.

In this work we approach this problem in three different ways:

- Two methods for interpolating learned input sequences.
- One method approximating the relationship between reference trajectories and the input sequences obtained from ILC using neural networks.

A. Interpolating learned input sequences

The first way of tackling this problem is to use the reference trajectories in R and the set of input sequences learned through ILC U_R, and interpolate between the input sequences U_R to create a new feedforward $\hat{\mathbf{u}}$. This is easier said than done, as it introduces two new problems.

The first problem is practical. Due to different velocities, accelerations, and jerk two different trajectories might not have the same length or might be behaving differently at the same time step (one might be accelerating while the other is moving at a constant speed). For the interpolated input sequence to be sensible the behaviour of the signals to be interpolated should be the same.

Problem 2 (Interpolating signals with different amounts of samples): Construct a function h:

$$x_{o,s} = h(x_o, x_t), \tag{16}$$

where h maps an original signal x_o to a new signal $x_{o,s}$, which matches a target signal x_t in length and behaviour in terms of constant velocity, acceleration or jerk.

The second problem is an optimization problem. Since we want to interpolate inputs such that the approximation is as close as possible to the input obtained through ILC, without knowing the true input obtained through ILC.

Problem 3 (Calculating the interpolation coefficients): Calculate interpolation coefficients stacked in a vector

$[k_1(t), k_2(t), \cdots k_{n_R}(t)]$, sucht that the input approximation is given by:

$$\hat{\mathbf{u}}(t) = \sum_i k_i(t)\mathbf{u}_i^*(t) \tag{17}$$

such that the approximation $\hat{\mathbf{u}}$ is within a small bound ϵ of the ILC input $\mathbf{u}^*$:

$$||\hat{\mathbf{u}} - \mathbf{u}^*||_2 \leq \epsilon. \tag{18}$$

B. Approximating the relationship between reference trajectory and input sequence using neural networks

A different approach from analytically calculating a new feedforward input sequence for a reference $\mathbf{r}_t$ would be to approximate the relationship between reference trajectory and input sequence. Feedforward neural networks can approximate any continuous function with arbitrary accuracy within a domain [12]. We want to use this property to approximate the relationship between the reference trajectories $\mathbf{r}_o$ and the input sequence obtained through ILC $\mathbf{u}_o^*$. Then this relationship is used to approximate new input sequences $\hat{u}$ for a target reference trajectory $\mathbf{r}_t \in \mathcal{P}$.

Problem 4 (Create a neural approximator based on obtained ILC data): Use the reference data $\mathbf{r}_o$ and the calculated ILC input sequences $\mathbf{u}_o^*$ to obtain a relationship g minimizing the approximation error

$$\sum_{i=1}^{n_R} ||\mathbf{u}_{o,i} - g(\mathbf{r}_{o,i})||_2. \tag{19}$$

This relationship can then be used for nonlearned $\mathbf{r}_t \in \mathcal{P}$ to construct an approximate input sequence

$$\hat{\mathbf{u}_t} = g(\mathbf{r}_t). \tag{20}$$

C. Performance metrics used to validate methods

The developed methods are tested against all reference trajectories and learned input sequences in the target set T. For each of the methods, an example of the difference in input sequence compared to the sequence learned through ILC, and the resulting tracking error compared to the tracking error obtained through ILC is shown. This reference trajectory is shown in Figure 6, and the input sequence obtained through ILC is shown in Figure 8.

We also define four key metrics, summarizing the performance of a method for a certain reference, as follows:

- The average approximation of the input sequence:

$$\zeta_u = \max\left(0, \left(1 - \frac{||\mathbf{u}^* - \hat{\mathbf{u}}||_2^2}{||\mathbf{u}^*||_2^2}\right) \cdot 100\%\right). \tag{21}$$

- The maximum input difference:

$$\Delta_{u,max} = \max\left(|\mathbf{u}^* - \hat{\mathbf{u}}|\right). \tag{22}$$

- The root mean square error:

$$e_{RMS} = ||\mathbf{e}||_2. \tag{23}$$

- The average error reduction:

$$\zeta = \max\left(0, \left(1 - \frac{||\mathbf{e}||_2}{||\mathbf{e}_{MA}||_2}\right) \cdot 100\%\right). \tag{24}$$

  Where $\mathbf{e}$ is the tracking error obtained with the feedforward input sequence, and $\mathbf{e}_{MA}$ is the tracking error with mass acceleration feedforward control.
- The peak error over the signal samples:

$$e_{max} = \max\left(|\mathbf{e}|\right) \tag{25}$$

IV. Interpolation of ILC sequences

The following two methods aim to perform interpolation on the input sequences obtained through ILC on the learned reference set in order to obtain a new input sequence for a non-learned reference trajectory.

A. Interpolation of time-phase scaled signals

A third-order reference trajectory can be split into different phases, categorized by the lowest constant reference derivative active. This results in position, velocity, acceleration, and jerk phases. When the maximum values for reference derivatives p_1, p_2, and p_3 are different, the duration and starting time of these phases do not match anymore. For a sensible interpolation, the reference trajectories should be in the same phase at each time step. A possible way to align the phases is to scale each phase in time such that the phases' duration and starting point match up.

Given a signal x, a function pi is defined, which assigns a phase to each time index i of a signal x

$$p_{i,x} : \{i \in \mathbb{N} | 1 \leq i \leq N_x\} \rightarrow \{y \in \mathbb{N} | 1 \leq y \leq n_{phase}\}, \tag{26}$$

where n_{phase} is the total amount of phases in a third order profile and N_x is the length of signal x. A function d is defined, which assigns a duration to each phase

$$d_x : \{i \in \mathbb{N} | 1 \leq i \leq n_{phase}\} \rightarrow \{y \in \mathbb{R}_+\}. \tag{27}$$

Lastly a function s is defined, which assigns a starting index to each phase

$$s_{i,x} : \{i \in \mathbb{N} | 1 \leq i \leq n_{phase}\} \rightarrow \{y \in \mathbb{N} | 1 \leq y \leq N_x\}. \tag{28}$$

An algorithm has been developed to scale a signal such that it matches a target signal in both phase and length. The steps are given in Algorithm 1, and the algorithm has been visualized in Figure 12. The algorithm is applied with the original reference position $\mathbf{r}_o$, velocity $\mathbf{v}_o$, acceleration $\mathbf{a}_o$, and learned input $\mathbf{u}_o^*$ of each of the n_R original trajectories. This results in n_R sets of time-phase scaled sequences $\mathbf{r}_s$, $\mathbf{v}_s$, $\mathbf{a}_s$, and $\mathbf{u}_s^*$. Now that all of the original signals have the same length, and are in the same phase for each time step, we can interpolate between the signals.

Given the output of Algorithm 1 the next step is to calculate the interpolation coefficients. Since the true input that would be obtained using ILC is not known, the interpolation coefficients are based on the value of the reference signal and its derivatives. Due to the scaling in time of the time-phase scaling algorithm the physical link between the signals is broken. The scaled velocity is no longer the first derivative of the scaled position, the

Algorithm 1: Time-phase scaling of signals.

Input: An original signal x_o, and the phase index over time of the target signal pi_t

Output: A time-phase scaled signal $x_{o,s}$, that is aligned with the traget signal in terms of phase profile.

Initiate $x_{o,s}$ to be a vector of zeros with the size of the domain of p_t

for $i = 1$ **to** $length(x_{o,s})$ **do**

  $p\_index = p_{i,t}(i)$; // Find index of current phase of the target signal

  $\gamma = \frac{d_o(p\_index)}{d_t(p\_index)}$; // Find the scaling ratio based on durations

  /* Find index j of x_o to be used as $x_{o,s}(i)$ */

  $j_{aux1} = i - s_{i,t}(p\_index)$; // Shift with the starting index in target to align start of the signal with 0

  $j_{aux2} = \gamma \cdot j_{aux1}$; // Scale with ratio γ

  $j = j_{aux2} + s_{i,o}(p\_index)$; // Shift with starting index in original

  $x_{o,s}(i) = x_o(round(j))$; // Ensure integer and use index

return $x_{o,s}$

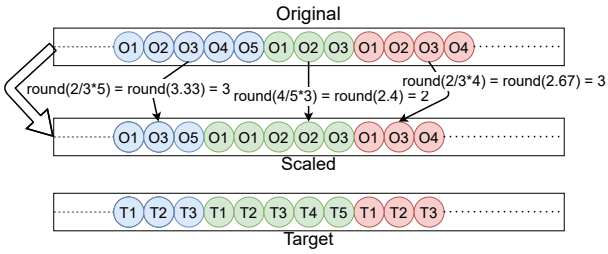

Figure 12: Visualization of the time-phase scaling algorithm.

scaled acceleration is no longer the second derivative, and so forth. For this reason the time-phase scaled values of the original reference derivatives should be taken into consideration when calculating the interpolation coefficients in order to obtain a sensible approximation. We allow for time varying interpolation coefficients, such that we compute the input $\hat{\mathbf{u}}$ at time i as

$$\hat{\mathbf{u}}(i) = \begin{bmatrix} \mathbf{u}^*_{1,s}(i) & \cdots & \mathbf{u}^*_{n,s}(i) \end{bmatrix} \mathbf{k}(i), \tag{29}$$

where

$$\mathbf{k}(i) = \arg\min_{\mathbf{k}(i)} \left\| \begin{bmatrix} \mathbf{r}_t(i) \\ \mathbf{v}_t(i) \\ \mathbf{a}_t(i) \end{bmatrix} - \begin{bmatrix} \mathbf{r}_{1,s}(i) \cdots \mathbf{r}_{n,s}(i) \\ \mathbf{v}_{1,s}(i) \cdots \mathbf{v}_{n,s}(i) \\ \mathbf{a}_{1,s}(i) \cdots \mathbf{a}_{n,s}(i) \end{bmatrix} \mathbf{k}(i) \right\|_2 \tag{30}$$

1) Results of the Interpolation of Time-Phase Scaled Signals in Simulation: An example of the error between the interpolated input sequence and the input sequence obtained through ILC is shown in Figure 13. The resulting tracking error is shown in Figure 14. As can be seen, the input is approximated quite well in most phases of the signal, but the difference between inputs peaks during the jerk phase. These peaks also cause spikes in tracking error, which is corrected for by the feedback controller. Because the time-phase scaling algorithm does not conserve the physical properties of the signal, the time-phase scaled input output trajectories are no longer solutions of the system. This means that the interpolation can at best give a decent approximation, but can not give a theoretically optimal result.

The arithmetic mean of all metrics over the references in T is given in Table III. Interesting to note is that while the average input is only 98.08 % accurate, the error reduction with respect to mass acceleration feedforward is still 99.37 %. This shows that a generalized feedforward controller can result in accurate tracking, even if it does not perfectly replicate the ILC input sequences.

Table III: Mean of performance metrics of the time-phase scaled interpolation.

ζ_u	98.08 %
$\Delta_{u,max}$	10.22 N
ζ	99.37 %
e_{RMS}	1.85×10^{-6} m
e_{max}	11.45×10^{-6} m

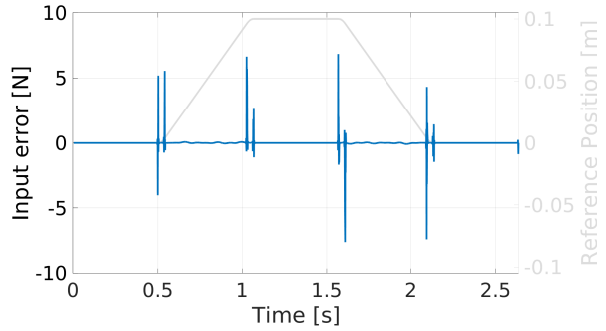

Figure 13: Input error difference ILC and time-phase scaled interpolation.

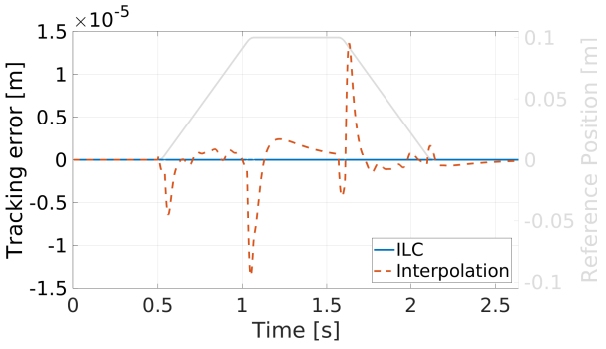

Figure 14: Tracking error comparison ILC and time-phase scaled interpolation.

B. Interpolation of polynomial coefficients

In the interpolation of polynomials approach the aim is still to obtain an input sequence that will give ILC

like performance. Different from the time-phase scaled interpolation approach, this approach tries to conserve the physical properties of the reference signal, and we exploit the fact that all of the seperate phases of the reference can be described using polynomials.

1) Describing a third-order reference trajectory as a piecewise function of polynomials: A third-order reference trajectory can be seen as a piecewise function of third-degree polynomials. Each of these pieces is one of the phases discussed in the previous section. Instead of scaling a signal in time and interpolating for each timestep, we can calculate the interpolation coefficients of a phase based on the coefficients of the polynomial describing the position during the said phase. The input sequence can then be interpolated using the same coefficients.

This does however not solve the problem of the phases being of a different length. To solve this we pose a condition on the original reference trajectories to be used for polynomial interpolation: The duration of each phase of each original reference has to be longer than the longest duration of the corresponding phase in all possible target trajectories. For this reason, the interpolation of polynomials uses the set of original references R_{poly} instead of R.

2) Original references for interpolation of polynomials: The constraint that each phase of each original reference needs to be at least as long as the same phase in the target reference is placed on the original reference set because otherwise there would be no input data at the end of the time series. If a phase has m samples in the target reference, then the first m input samples from each original input sequence are used, as shown in Figure 15. To satisfy this constraint the new reference set R_{poly} is defined, with the parameters as shown in Figure 16. Due to the constraint on the duration $R_{poly} \nsubseteq \mathcal{P}$.

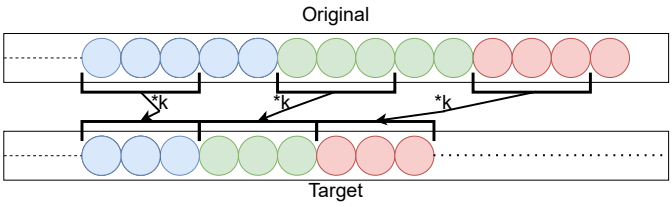

Figure 15: Point selection for polynomial interpolation.

3) Calculation of interpolation coefficients based on polynomial coefficients: Any phase of a third order position profile can be fitted as a third order polynomial.

$$r_i(t) = d_i t^3 + c_i t^2 + b_i t + a_i$$

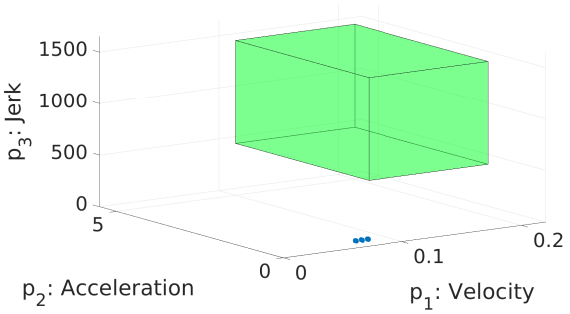

Figure 16: The reference parameters of the 27 references in R_{poly}, shown with space $\mathcal{P}$.

given at least four original reference trajectories which coefficients are linearly independent,

$$\mathbf{r}_{1-n_{R_{poly}}}(t) = \underbrace{\begin{bmatrix} d_1 & c_1 & b_1 & a_1 \\ d_2 & c_2 & b_2 & a_2 \\ \vdots & \vdots & \vdots & \vdots \\ d_{n_{R,poly}} & c_{n_{R,poly}} & b_{n_{R,poly}} & a_{n_{R,poly}} \end{bmatrix}}_{\mathbf{P}} \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix}$$

where $rank(\mathbf{P}) \geq 4$,

any third order target reference can be made through a linear combination of the original references

$$r_t(t) = \underbrace{\begin{bmatrix} d_t & c_t & b_t & a_t \end{bmatrix}}_{\mathbf{p_t}} \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix} \tag{31}$$

$$= \underbrace{\begin{bmatrix} k_1 & k_2 & \cdots & k_{n_{R_{poly}}} \end{bmatrix}}_{\mathbf{k}} \mathbf{r}_{1-n_{R_{poly}}}(t), \tag{32}$$

$$\mathbf{k} = \arg\min_{\mathbf{k}} \mathbf{k}\mathbf{k}^T \quad \text{subject to} \quad \mathbf{p_t} = \mathbf{kP} \tag{33}$$

Using this knowledge a target reference trajectory can be split into the same phases as in section IV-A, and for each of these phases the polynomial coefficients can be calculated. The same can be done for the original references, after which for each phase the interpolation coefficients $\mathbf{k}_i$ can be calculated using formula (33).

Using the calculated interpolation weights $\mathbf{k}_{i,phase}$ for each phase an input sequence $\hat{\mathbf{u}}_{i,phase}$ can be calculated as

$$\hat{\mathbf{u}}_{i,phase} = \begin{bmatrix} \mathbf{u}_1^*[t_{1,i,phase}] \\ \vdots \\ \mathbf{u}_{n_{R,poly}}^*[t_{n_{R,poly},i,phase}] \end{bmatrix} \mathbf{k}_{i,phase}^T, \tag{34}$$

where the used indices are given as

$$t_{j,i,phase} = [s_j(i,phase) \ldots s_j(i,phase) + d_t(i,phase)] \quad \forall j \in [1, n_{R,poly}] \tag{35}$$

and where $\mathbf{u}*_i$ is the input sequence obtained after 20 iterations for the references in R_{poly}, and for each phase

we take as many input values as there are time steps in the target reference $\mathbf{r}_t$ for the current phase, starting from the starting index of the same phase in the corresponding original reference trajectory.

The benefit of this constraint is that there are always samples in the original reference and input sequence to use, and it does preserve the physical properties of the signal. The downsides are that the origin references in R_{poly} are slower than the target references in $\mathcal{P}$, meaning inputs are extrapolated, instead of interpolated. Since the final inputs of a phase are discarded, eventual preactuation for the next phase is no longer active.

4) Results of interpolation of polynomials in simulation: An example of the error between the interpolated input sequence and the input sequence obtained through ILC is shown in Figure 17. The resulting tracking error is shown in Figure 18. The arithmetic mean of all metrics over the 729 references in T is given in Table IV.

The interpolation of polynomials does not result in correct inputs during the switching of phases, resulting in larger input errors when the reference profile switches from phase. This can be seen by the spikes in Figure 17. This is presumably because the tail end of the input sequence of the previous phase is discarded, causing a discrepancy between the internal state of the original signals in R_{poly}, and the actual internal state of the system in the target reference.

Table IV: Mean of performance metrics of the interpolation of polynomials.

ζ_u	89.19 %
$\Delta_{u,max}$	54.04 N
ζ	94.30 %
e_{RMS}	16.46×10^{-6} m
e_{max}	120.92×10^{-6} m

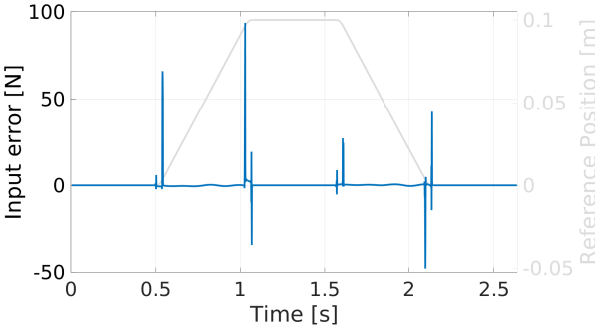

Figure 17: Input error between ILC and the interpolation of polynomials.

V. Feedforward approximation using neural networks

Both interpolation methods take an analytical approach at constructing new input sequences. In this section a different kind of approach is taken, where input sequences obtained through ILC are used to train a neural network approximator, instead of being analytically combined. This approach is simpler, since it does not require system knowledge or any new methods. The downside are that neural networks are not deterministic, there is no one size fits all recipe for creating good approximators, and not implementing system knowledge can lead to worse results.

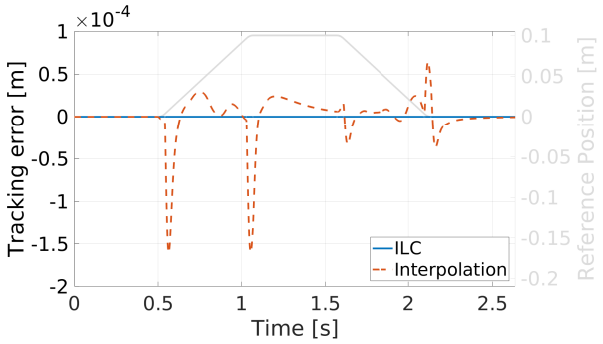

Figure 18: Tracking error comparison ILC and interpolation of polynomials.

A. Designing the neural network

A feedforward neural network with sigmoidal activation functions is known as a universal approximator [12]. It is used to approximate the relation g in

$$u^*(t) = g(r(t), \dot{r}(t), \ddot{r}(t), \dddot{r}(t)). \tag{36}$$

An example of this has been implemented in [15]. This approach has also been implemented in this project to compare the approach to the interpolation methods. The neural network structure used in this project is visualized in Figure 19. It consists of two hidden layers with four neurons with a sigmoidal activation fuction each, and a linear output layer. This design gave the best consistent results for the linear motor model in simulation. Other networks tested consisted of both larger and smaller networks, and also tried linear and ReLU activation functions.

The neural network is trained using the data from the 27 references in R as input, and the corresponding input sequences in U_R as the desired neural network output. divided into 70 % training data, 15 % validation data, and 15 % test data.

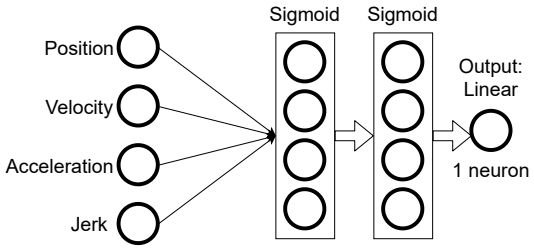

Figure 19: Structure of the neural network approximator.

B. Results

The trained neural network feedforward controller is implemented on the model and tested for all reference

trajectories in T. An example of the error between the interpolated input sequence and the input sequence obtained through ILC is shown in Figure 13. The resulting tracking error is shown in Figure 14. The mean of the four performance metrics for all 729 trajectories in T is given in Table V.

Table V: Mean of performance metrics using the neural network approximator.

ζ_u	98.20 %
$\Delta_{u,max}$	9.35 N
ζ	99.51 %
e_{RMS}	1.42×10^{-6} m
e_{max}	6.61×10^{-6} m

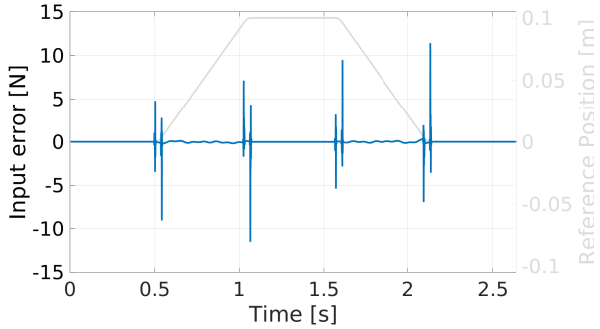

Figure 20: Input error between ILC and NN generated input.

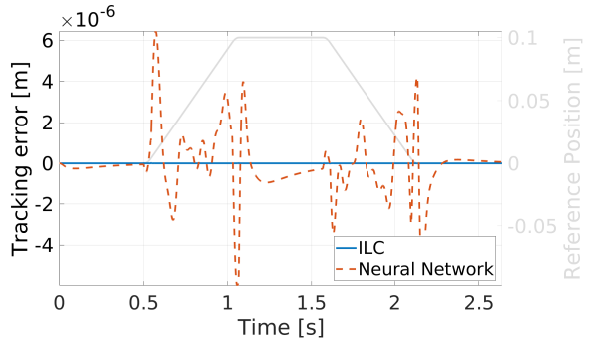

Figure 21: Tracking error comparison ILC and NN generated input.

VI. Comparison

In this section, the different methods are compared. Firstly their results are compared quantitatively, then the methods are compared qualitatively.

A. Quantitative comparison

All 3 methods are trained on a basis of 27 original references. The neural network approach and the time-phase scaled interpolation are trained on reference trajectory set R, and the interpolation of polynomials is based on original reference trajectory set R_{poly}. All 3 methods are tested on the 729 references in T. The average metrics of all three methods discussed are presented in Table VI. The neural network and the interpolation of time-phase scaled signals performed the best over all metrics, with an input approximation accuracy of over 98 % on average, and an average error reduction of over 99 %. Both the methods perform better than the mass acceleration feedforward. The interpolation of polynomials method performs not as well over all metrics but still reduces the error with than 94.30 %.

A few boxplots have been made to visualize how the mean of the performance metrics over all 729 reference trajectories in T is distributed. In these plots the red line shows the median of the data, the box shows the interquartile range, going from the 25^{th} percentile to the 75^{th}, and the whiskers show the minimum and maximum. The whiskers have a maximum length of 1.5× the interquartile range and observations outside the maximum length of the whiskers are shown separately.

Figures 22 and 23 show the input approximation properties of the methods. Here we can see that both the neural network approach and the time-phase scaled interpolation perform very similarly. Both have a very similar median, but the neural network approach has a higher spread, making it less reliable for input approximation. Figures 25, 24, and 26 show the tracking performance of the different methods. We can see here that while the neural network approach had a higher spread in the input approximation, the tracking error of the neural network approach has a smaller spread. Both the neural network approach and the time-phase scaled interpolation have a similar maximum in the error reduction ζ, and a similar minimum in the peak error, but the smaller spread of the neural network approach makes it such that it is slightly better on average. We can also see that both methods are better than the rigid body feedforward for most of the references.

The interpolation of polynomials is not complete and should take into account the internal state of the system when switching states. At the moment it does not, and because of that the performance is not as good as the other methods or the rigid body feedforward.

Table VI: Comparison methods of multiple reference ILC.

Method:	ζ_u	$\Delta_{u,max}$	ζ
Time phase scaled interpolation	98.08 %	10.22 N	99.37 %
Interpolation of polynomials	89.19 %	54.04 N	94.30 %
Neural networks	98.20 %	9.35 N	99.51 %
MA feedforward	-	-	0 %

Method:	e_{RMS}	e_{max}
Time phase scaled interpolation	1.85×10^{-6} m	11.45×10^{-6} m
Interpolation of polynomials	16.46×10^{-6} m	120.92×10^{-6} m
Neural networks	1.42×10^{-6} m	6.61×10^{-6} m
MA feedforward	296.63×10^{-6} m	619.28×10^{-6} m

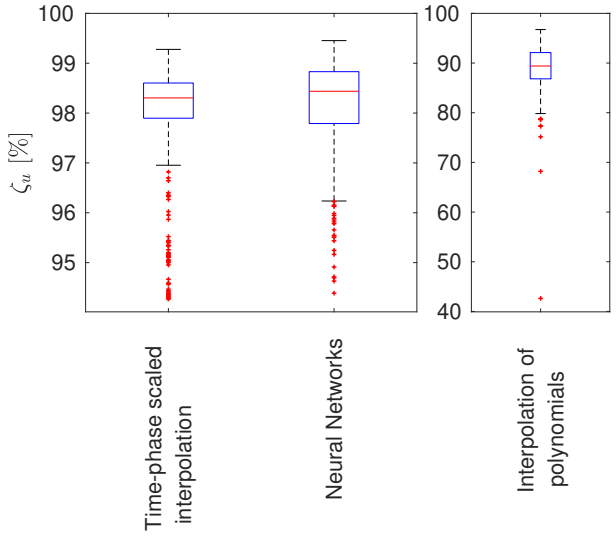

Figure 22: Comparison of the input approximation ζ_u per approach.

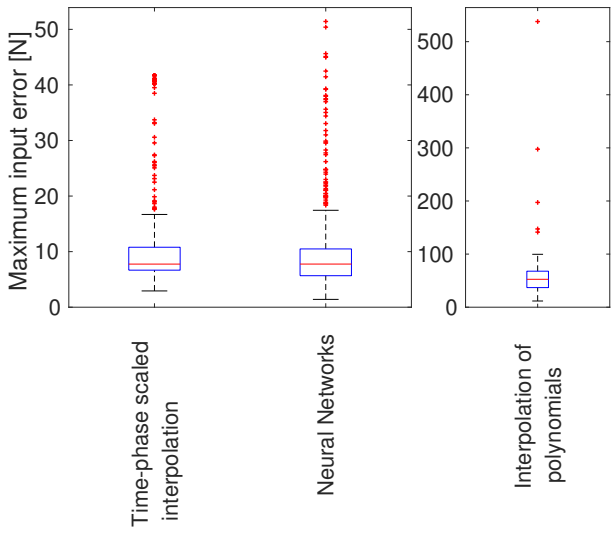

Figure 23: Comparison of the maximum input error $\Delta_{u,max}$ per approach.

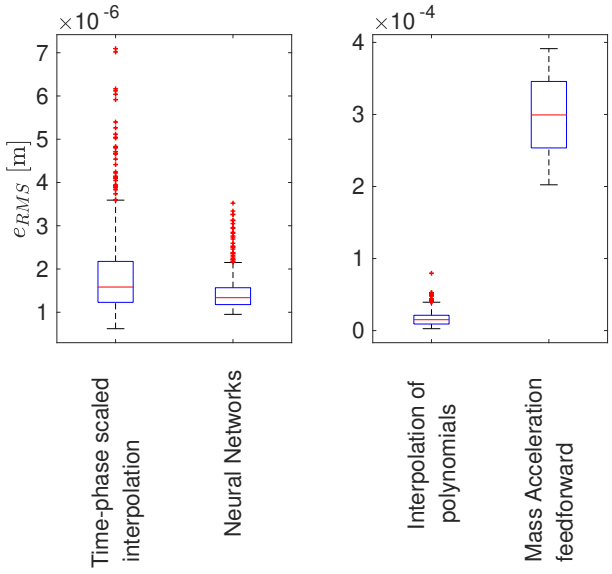

Figure 24: Comparison of the root mean square of the tracking error per approach.

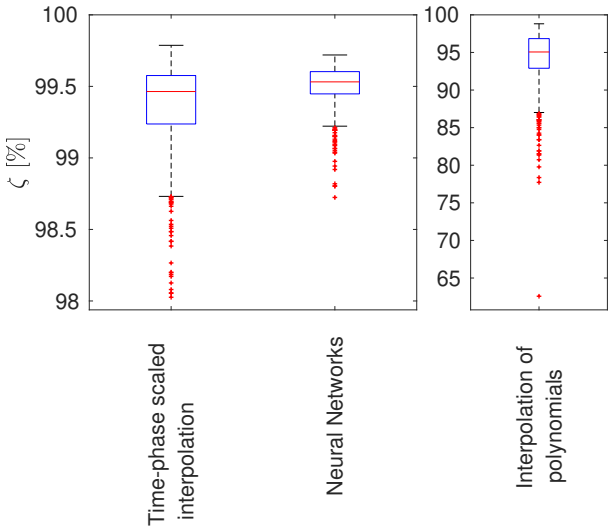

Figure 25: Comparison of the reduction of the tracking error per approach.

B. *Qualitative Comparison*

The time-phase scaled interpolation is not theoretically optimal, since the time-phase scaling algorithm does not conserve the physical properties of the signal. The performance was approximation was accurate on the setup used, but there are no guarantees for other setups. This method is simple in implementation since the steps are always the same regardless of the system. This method also conserves the original references' data. This means that if one of the original references is the target reference, the ILC input sequence can be reconstructed exactly.

The interpolation of polynomials is theoretically more sound since the properties of the signals are conserved. When taking the initial conditions into account correctly we might even be able to prove it to be optimal for linear systems. The interpolation of polynomials has the constraint that each phase of all the original references has to be longer than the corresponding phase in all of the target references. This causes the original references of the interpolation of polynomials to be significantly slower than the target references, forcing extrapolation over interpolation. This should not be an issue for linear systems but will be an issue for nonlinear systems or disturbances. This method is the same for every system, so implementation is consistent. This method does conserve the original references' data.

The neural network approach is almost always an approximation, just like the time-phase scaled interpolation. Only if the true function from the neural network inputs is a linear combination of instances of the activation function it can become optimal, which is not often the case. The neural network approach results in the best performance

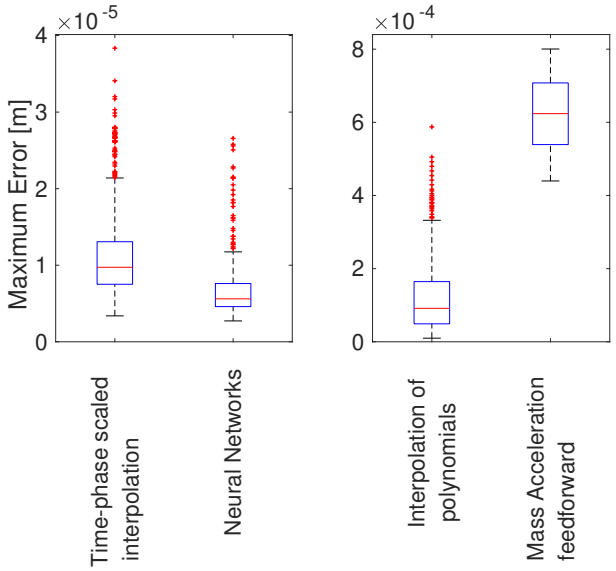

Figure 26: Comparison of the maximum tracking error e_{max} per approach.

in this implementation, but the implementation of the neural network approach is not consistent. Depending on your system the activation function, amount of layers, and amount of neurons per layer that gives a satisfactory result might vary, making the implementation more difficult. This method does not conserve the original references' data. This means that even when one of the original references is the target reference the constructed input sequence is an approximation.

All of the methods discussed in this paper can be calculated offline using the reference trajectory. The two interpolation methods are a bit more memory and computationally intensive, since they have to save all of the original reference trajectories and input sequences in memory, and solve the optimization problems to create a new input sequence. The neural network approach learns a set of weights and biases based on the original references trajectories and input sequences. After learning the neural network approach only has to save the weights and biases and the calculation consists of simple arithmetic and evaluation of the activation functions. This makes the neural network more computationally and memory-efficient than the interpolation methods.

VII. Conclusion

In this thesis, we investigated three approaches to achieve ILC like performance on a range of reference trajectories, without learning an input sequence for all possible reference trajectories. This assumes that we have a set of references with converged ILC on the system in question.

All results have only been tested in simulation on the linear motor model. The findings of this thesis should be confirmed on more models to investigate if the results are consistent.

The time-phase scaled interpolation provided good results, with the performance metrics being relatively close to the neural network approach, while having the benefit of conserving the original reference data and input sequences and having a consistent implementation.

The interpolation of polynomials did not result in a desirable input sequence, since the internal system state was not taken into account correctly. A method should be developed that does take this method into account to see how well it can perform. The method does conserve the original reference data and input sequences.

The neural network approach gave the best results considering error reduction. It also is the least computationally and memory intensive implementation once trained. The exact network and activation function that will give good results will be dependant on the system, so the implementation is not consistent. The original reference data and input sequence are not conserved, thus there will even be an approximation error on trained references.

Future research can proceed in the following directions:

- Take the internal system state into account in the interpolation of polynomials approach, such that the switching of phases does not give as big of an error as it does now.
- Investigate the use of different AI approximators, like for example Gaussian processes, recurrent neural networks, reinforcement learning strategies, etcetera.
- Develop a method to integrate the interpolation methods with artificial intelligence, providing the same relation to data as current interpolation methods, but hopefully improve the performance.

References

[1] C. Rohrig and A. Jochheim. "Identification and compensation of force ripple in linear permanent magnet motors". In: *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*. Vol. 3. 2001, pp. 2161–2166. DOI: 10.1109/ACC.2001.946068.

[2] Anorad Inc. *Linear Motor Reference Manual*. 1999.

[3] M.L.G. Boerlage, M. Steinbuch, P.F. Lambrechts, et al. "Model-based feedforward for motion systems". In: *2003 IEEE International Conference on Control Applications (CCA), Turkey, Istanbul*. Vol. 2. United States: IEEE, 2003, pp. 1158–1163. ISBN: 0-7803-7729-X. DOI: 10.1109/CCA.2003.1223174.

[4] D.A. Bristow, M. Tharayil, and A.G. Alleyne. "A survey of iterative learning control". In: *IEEE Control Systems Magazine* 26.3 (2005), pp. 96–114. ISSN: 10010920. DOI: 10.1109/mcs.2006.1636313.

[5] J. Xu and Y. Tan. "Linear and Nonlinear Iterative Learning Control". In: *Lecture Notes in Control and Information Sciences* 291 (2003). ISSN: 01708643. DOI: 10.1007/3-540-44845-4.

[6] D.J. Hoelzle, A.G. Alleyne, and A.J. Wagoner Johnson. "Basis task approach to iterative learning control with applications to micro-robotic deposition". In: *IEEE Transactions on Control Systems Technology* 19.5 (2011), pp. 1138–1148. ISSN: 10636536. DOI: 10.1109/TCST.2010.2063030.

[7] D.J. Hoelzle and K. Barton. "Flexible iterative learning control using a library based interpolation scheme". In: *Proceedings of the IEEE Conference on Decision and Control* 2 (2012), pp. 3978–3984. ISSN: 01912216. DOI: 10.1109/CDC.2012.6425808.

[8] B. Hencey and A.G. Alleyne. "A robust controller interpolation design technique". In: *IEEE Transactions on Control Systems Technology* 18.1 (2010), pp. 1–10. ISSN: 10636536. DOI: 10.1109/TCST.2008.2009121.

[9] M.C.J. Baggen, M.F. Heertjes, and M.J.G. Molengraft, van de. *Setpoint variation in iterative learning schemes*. English. DCT rapporten. DCT 2006.098. Technische Universiteit Eindhoven, 2006.

[10] J.J.M. Wijdeven, van de and O.H. Bosgra. "Using basis functions in iterative learning control : analysis and design theory". English. In: *International Journal of Control* 83.4 (2010), pp. 661–675. ISSN: 0020-7179. DOI: 10.1080/00207170903334805.

[11] J.J. Bolder, T.A.E. Oomen, and M. Steinbuch. "Exploiting rational basis functions in iterative learning control". English. In: *Proceedings of the 52nd Conference on Decision and Control, 10-13 December 2013, Florence, Italy*. 2013, pp. 7321–7326.

[12] K. Hornik, M. Stinchcombe, and H. White. "Multilayer Feedforward Networks are Universal Approximators". In: *Neural Networks* 2 (1989), pp. 359–366. DOI: 10.1016/b978-0-08-051433-8.50011-2.

[13] K. J. Hunt, D. Sbarbaro, R. Zbikowski, et al. "Neural networks for control systems-A survey". In: *Automatica* 28.6 (1992), pp. 1083–1112. ISSN: 00051098. DOI: 10.1016/0005-1098(92)90053-I.

[14] O. Sørensen. "Additive feedforward control with neural networks". In: *IFAC Proceedings Volumes* 32.2 (1999), pp. 1378–1383. ISSN: 14746670. DOI: 10.1016/s1474-6670(17)56233-3.

[15] S. Bosma, "The generalization of feedforward control for a periodic motion system". MSc thesis. Delft University of Technology, 2019.

[16] K. Patan, M. Patan, and D. Kowalów. "Neural networks in design of iterative learning control for nonlinear systems". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 13402–13407. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.2277.

[17] T.T. Nguyen. "Identification and compensation of parasitic effects in coreless linear motors Identification and Compensation of Parasitic Effects in Coreless Linear Motors". PhD thesis. Eindhoven University of Technology, 2018. ISBN: 9789038645865.

[18] R. Zanasi and R. Morselli. "Third order trajectory generator satisfying velocity, acceleration and jerk constraints". In: *IEEE Conference on Control Applications - Proceedings* 2.1 (2002), pp. 1165–1170. DOI: 10.1109/cca.2002.1038770.

[19] M. Steinbuch and R. van den Molengraft. "Iterative Learning Control of Industrial Motion Systems". In: *IFAC Proceedings Volumes* 33.26 (2000), pp. 899–904. DOI: https://doi.org/10.1016/S1474-6670(17)39259-5. URL: http://www.sciencedirect.com/science/article/pii/S1474667017392595.

[20] M. Tomizuka, T.C. Tsao, and K.K. Chew. "Analysis and synthesis of discrete-time repetitive controllers". In: *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME* 111.3 (1989), pp. 353–358. ISSN: 15289028. DOI: 10.1115/1.3153060.

[21] J.A. Butterworth, L.Y. Pao, and D.Y. Abramovitch. "The effect of nonminimum-phase zero locations on the performance of feedforward model-inverse control techniques in discrete-time systems". In: *Proceedings of the American Control Conference* (2008), pp. 2696–2702. ISSN: 07431619. DOI: 10.1109/ACC.2008.4586900.