arXiv:1709.04407v1 [cs.RO] 13 Sep 2017

An Inversion-Based Learning Approach for Improving Impromptu Trajectory Tracking of Robots with Non-Minimum Phase Dynamics

Siqi Zhou, Mohamed K. Helwa, and Angela P. Schoellig

***Abstract*— This paper presents a learning-based approach for impromptu trajectory tracking of non-minimum phase systems — systems with unstable inverse dynamics. In the control systems literature, inversion-based feedforward approaches are commonly used for improving the trajectory tracking performance; however, these approaches are not directly applicable to non-minimum phase systems due to the inherent instability. In order to resolve the instability issue, they assume that models of the systems are known and have dealt with the non-minimum phase systems by pre-actuation or inverse approximation techniques. In this work, we extend our deep-neural-network-enhanced impromptu trajectory tracking approach to the challenging case of non-minimum phase systems. Through theoretical discussions, simulations, and experiments, we show the stability and effectiveness of our proposed learning approach. In fact, for a known system, our approach performs equally well or better as a typical model-based approach but does not require a prior model of the system. Interestingly, our approach also shows that including more information in training (as is commonly assumed to be useful) does not lead to better performance but may trigger instability issues and impede the effectiveness of the overall approach.**

I. Introduction

High-accuracy trajectory tracking is essential for many robotic and autonomous systems. The concept of using inverse dynamics to enforce high-accuracy or exact tracking is widely used in the control systems literature [1]. However, for many practical problems ranging from aircraft control [2] and flexible robot arm end-effector tracking [3] to hard disk drive track-following [4], the input-output dynamics are non-minimum phase — i.e., the inverse dynamics are inherently unstable. The non-minimum phase nature poses challenges in classical control designs [5] and prohibits the direct application of inversion-based approaches.

In order to take advantage of system inversion for improving the tracking performance, various model-based approaches have been proposed to resolve the instability issue associated with the system inverse of non-minimum phase systems. Two main streams of these approaches are based on *(i)* pre-actuation [6] and *(ii)* inverse approximation [7]. In the pre-actuation approach, first proposed in [6], a bounded input is ensured by pre-loading the system state to a desired initial state designed for the particular desired trajectory. Though exact tracking can be achieved with bounded input signals, the solutions are trajectory-specific and require significant setup time in order to reach the desired initial condition [8].

The authors are with the Dynamic Systems Lab (www.dynsyslab.org), Institute for Aerospace Studies, University of Toronto, Canada. Emails: siqi.zhou@robotics.utias.utoronto.ca, mohamed.helwa@robotics.utias.utoronto.ca, schoellig@utias.utoronto.ca

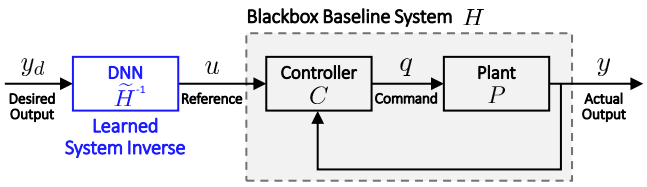

Fig. 1. A simplified block diagram of the proposed DNN-enhanced control architecture for output trajectory tracking. In this architecture, a baseline stabilizing controller is treated as a blackbox, and a DNN module is pre-cascaded to the baseline system to adjust reference signals to improve the tracking performance.

On the other hand, in the inverse approximation approaches, stability of the inverse is ensured by replacing the unstable components of the inverse dynamics with a stable approximation that is capable of achieving precise tracking (see [9], [7] and the references therein). As compared with the pre-actuation approaches, the approximate inversion approaches are more robust against modeling errors and consequent instability issues. Moreover, since the inversion is system-specific, the approximate inversion approaches can be more easily generalized to impromptu tracking tasks (i.e., tracking an arbitrary, feasible trajectory in a one-shot manner). However, it should be noted that due to the model-based nature of both approaches, the effectiveness depends on detailed and sufficiently accurate dynamic models of the systems. This limitation motivates the investigation of learning techniques, which leverage data to improve the performance of model-based approaches.

For *minimum phase systems*, different inverse dynamics learning approaches have been studied. In our previous work [10], [11], a deep-neural-network-based (DNN-based) architecture (Fig. 1) was proposed to enhance the tracking performance of uncertain, minimum phase systems by learning the inverse dynamics, and the effectiveness of the approach was experimentally verified on quadrotor vehicles. This approach led to an average of 43% tracking error reduction on 30 arbitrary, hand-drawn trajectories, as compared with the baseline feedback controller. In addition to our previous work, the potential of utilizing inverse-learning in high-accuracy tracking control has been explored and illustrated with different robotic platforms and learning techniques (e.g., Gaussian processes and support vector regressions); see for instance [12]–[14]. Nevertheless, extending these inverse dynamics learning schemes to non-minimum phase systems is still an open problem.

For *non-minimum phase systems*, a DNN-based feedback error learning approach has been proposed to train an inverse

dynamics model of the plant to improve tracking performance [15], [16]. In this approach, the weights of the DNN are updated adaptively based on the feedback error of a stabilized feedback control loop. As a direct consequence, the training of the DNN requires a plant or a good model of the plant dynamics in place, which may not always be available in practice or desired in the initial training phase. Moreover, similar to the adaptive inverse learning approaches discussed in [11], this approach is more susceptible to instability issues, especially when the DNN is not well-initialized or the hyper-parameters are not properly tuned [17].

In this paper, we present a DNN-based learning approach, based only on input-output data of a non-minimum phase system, that constructs an inverse dynamics model capable of enhancing impromptu tracking while, at the same time, guaranteeing stability of the overall system. In particular, informed by control theory, we will select appropriate inputs and outputs of the inverse-learning module that guarantees stability and performance enhancement. Proper input-output selection of the DNN model is critical for learning the inverse dynamics of non-minimum phase systems — including more or the wrong features may inevitably expose the learning to the unstable dynamic nature, and the consequent numerical issues can deteriorate the performance of the overall system. Apart from demonstrating the effectiveness of our proposed approach, through both theoretical discussions and simulations, we also show how the proposed learning approach relates to a common model-based approximate inversion approach for linear systems [7]. The proposed approach shares the same core concept as the model-based approach. Without requiring a detailed system model, the proposed approach leads to comparable or even better performance, and can be equally applied to nonlinear systems. In addition to theoretical insights and simulations, the efficacy of the proposed inverse-learning approach is experimentally verified on an inverted pendulum on a cart system setup.

II. Problem Formulation

We aim to provide an inversion-based learning approach for enhancing tracking performance of non-minimum phase systems. In particular, the proposed approach should satisfy the following objectives:

- O1. Stability — the overall system, including the learning module, is input-to-output stable [18];
- O2. Training — the learning module relies only on the input-output data rather than a system model;
- O3. Performance and Generalizability — with the pre-cascaded learning module, the root-mean-square (RMS) tracking error is reduced for impromptu tracking tasks compared to the baseline system.

A. Control Architecture

We consider the inversion-based learning architecture shown in Fig. 1. In this architecture, a baseline system is used to stabilize the plant, and a learned system inverse module is pre-cascaded to the baseline system to enhance the tracking performance via modifying the reference signal u. When training, the input-output data (u and y) generated from the baseline system is stored for constructing a training dataset that typically has y and u at selected time steps as the labeled inputs and u at current time step as the labeled output. When applying the trained module, the desired trajectory y_d is given to the learned inverse model as input (in place of y) to compute a reference u that is sent to the baseline system. This setup is similar to our recent work on minimum phase systems [10], [11].

The architecture in consideration is different from typical inversion-based feedforward architectures where the inverse of the open-loop plant P is used and the output signal from the inverse is directly applied to the plant [12], [16]. By learning the inverse of a stabilized baseline system, the proposed control architecture allows the performance enhancement problem to be decoupled from the plant stabilization problem, which simplifies the design, analysis, and practical implementation [11].

B. System Representations

We first motivate our proposed approach from the analysis of linear time-invariant (LTI), single-input-single-output (SISO) systems and then extend our discussion to nonlinear SISO systems. Effectiveness of the proposed approach for nonlinear systems is illustrated with both simulations and experiments. For linear systems, we represent the baseline feedback control system by the transfer function

$$H(z) = \frac{Y(z)}{U(z)} = \frac{N(z)}{D(z)} = \frac{1 + \sum_{i=1}^{n-r} \alpha_i z^i}{\sum_{i=0}^{n} \beta_i z^i} \tag{1}$$

where $U(z)$ and $Y(z)$ are the z-transforms of the input and output of the system, $N(z)$ and $D(z)$ denote the numerator and denominator polynomials, n is the order of the system, r is the relative degree of the system, and $\alpha_i, \beta_i \in \mathbb{R}$ are scalar constants. For nonlinear systems, we consider the control affine nonlinear system:

$$\begin{aligned} x(k+1) &= f(x(k)) + g(x(k))\, u(k), \\ y(k) &= h(x(k)), \end{aligned} \tag{2}$$

where $k \in \mathbb{Z}_{\geq 0}$ is the discrete time index, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}$ is the input, $y \in \mathbb{R}$ is the output, and $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are nonlinear smooth functions (i.e., functions whose all orders of differentiation exist and are continuous).

C. Assumptions

In deriving a solution for our problem, we make the following assumptions:

- A1. The underlying plant is stabilizable and the baseline system is stable;
- A2. At any time instant k, the current and future values of the desired trajectory are known up to time $k+n$, where n is the order of the baseline system;
- A3. Feedforward neural networks (FNN) and standard back-propagation algorithms are used for designing and training the inverse dynamics module.

Note that, A1 through A3 are reasonable assumptions in practice. For A2, a preview of n time steps of the desired

trajectory is typically not hard to achieve and this assumption does not prevent potential combinations with on-line trajectory generation and adaptation algorithms. Moreover, for A3, even though we use FNNs as the learning tool in this paper, the proposed approach can be easily adapted to other nonlinear regression learning techniques.

III. Non-Minimum Phase System Inverse-Learning

As introduced in Section I, for non-minimum phase systems, one set of solutions to resolve the instability issue in inversion-based approaches is to utilize stable inverse approximations. In this section, we adapt this concept to unknown, possibly nonlinear baseline systems using a DNN-based control architecture (Fig. 1). The challenge lies in the fact that the DNN training is built upon the input-output data of the baseline system, which essentially captures the unstable inverse dynamics nature in a discrete form.

A. Running Example Setup

In Subsections III-B through III-C, we will motivate and illustrate our proposed approach via a practical example of a four-wheeled car model. As shown in [5], the dynamics of a conventional car can be represented by a nonlinear bicycle model; by linearizing the system at zero steering and heading angles, and assuming a constant non-zero velocity v, the response from the steering angle of the front wheel ϕ to the lateral position of the center of the front wheels y can be represented by the following transfer function

$$P(s) = \frac{Y(s)}{\Phi(s)} = \frac{v(s+\frac{v}{l})}{s^2}, \tag{3}$$

where l is the length between the centers of the front and back wheels, and $Y(s)$ and $\Phi(s)$ are the Laplace transforms of $y(t)$ and $\phi(t)$, respectively. This linearized system (plant P in Fig. 1) has a zero at $s = -\frac{v}{l}$, which is minimum phase for $v > 0$ (forward driving) and non-minimum phase for $v < 0$ (backward driving). With chosen speed $|v| = 15$ m/s and length $l = 4$ m, standard pole placement can be used to find a baseline PD controller (controller C in Fig. 1) for computing the steering commands to achieve lateral position tracking. By applying zero-order-hold discretization with a sampling time of 0.15 s, the closed-loop dynamics from the reference lateral position u to the actual lateral position y can be expressed by the discrete-time transfer functions

$$H_{\text{forward}}(z) = \frac{Y(z)}{U(z)} = \frac{0.08334z - 0.04724}{z^2 - 1.602z + 0.6376} \tag{4}$$

for the forward motion and

$$H_{\text{backward}}(z) = \frac{Y(z)}{U(z)} = \frac{-0.04454z + 0.08064}{z^2 - 1.602z + 0.6376} \tag{5}$$

for the backward motion, where $U(z)$ and $Y(z)$ denote the z-transforms of the reference and actual lateral positions respectively[1]. These systems both have two poles at $z = 0.8607$ and $z = 0.7408$; yet, H_{forward} has a minimum phase zero at $z = 0.5668$, while H_{backward} has a non-minimum phase zero at $z = 1.8106$.

[1]Note that, when the reference u is a constant, the baseline system for backward-driving essentially tries to parallel part the car, which is a well-known non-minimum phase problem — the steering angle ϕ is commanded such that y first swings away from the sidewalk before moving towards it.

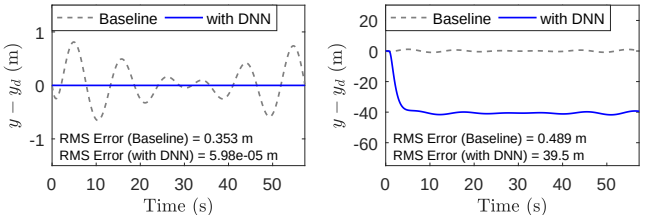

Fig. 2. Tracking error $y(k) - y_d(k)$ of the minimum phase forward-driving system (*left*) and the non-minimum phase backward-driving system (*right*) for a test trajectory $y_d(t) = \sin\left(\frac{4\pi}{23}t\right) + \cos\left(\frac{4\pi}{21}t\right) - 1$.

B. Background on Exact Inverse Learning

Given the control architecture in Fig. 1, in [11], it is shown that for a minimum phase system with a well-defined relative degree[2] r, exact tracking (i.e., $y(k+r) = y_d(k+r)$) can be achieved in theory by training the DNN to model the *exact inverse dynamics* of the baseline system. Following [11], for learning the exact inverse of system (2), the proper selection of inputs $\mathcal{I}$ and outputs $\mathcal{O}$ of the DNN module are $\mathcal{I} = \{x(k), y_d(k+r)\}$ and $\mathcal{O} = \{u(k)\}$. For LTI systems, based on the representation (1), the inputs of the DNN module can be alternatively selected as

$$\mathcal{I} = \{y_d(k-n+r : k+r), u(k-n+r : k-1)\}, \tag{6}$$

where consecutive time indexes are abbreviated with ':' [11].

When applying the above results to train the DNN module from input-output data, only basic system properties (i.e., n and r) are needed. In practice, a system's order n can be determined from basic physics laws and the relative degree r can be determined from simple experiments such as step responses. Although the *exact inverse learning* approach can be conveniently implemented in practice to enhance impromptu tracking [10], its effectiveness is restricted to *minimum phase systems* [11].

To illustrate the issue of directly applying the exact inverse learning approach to *non-minimum phase systems*, FNNs[3] with inputs selected as in (6) are applied to system (4) and (5). From Fig. 2, it can be seen that the exact inverse learning approach is effective only for the minimum phase, forward-driving system (4). For the backward-driving system (5), which is non-minimum phase, the learned inverse model suffers from numerical issues due to the inherent instability of the system inverse and deteriorates the baseline system performance. Thus, *direct application of the exact inverse learning approach to non-minimum phase systems is ineffective and leads to adverse effects.* In the following

[2]See [11] and the references therein for formal definitions of relative degree. The relative degree of a discrete-time system can be intuitively understood as the inherent time delay of the system. Experimentally, it is the number of time steps between the time at which an input u is applied and the time at which the output y first reacts.

[3]The FNNs, having 2 hidden layers of 5 hyperbolic tangent neurons, are trained with Matlab's Neural Network Toolbox. The training dataset is generated from the baseline system's responses on 42 different sinusoidal trajectories covering the potential operating space.

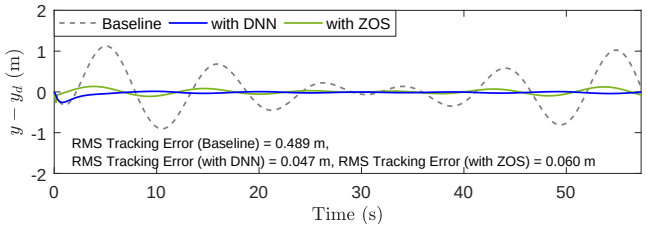

Fig. 3. Tracking error comparison of of the non-minimum phase backward-driving system (5), the system enhanced with the ZOS approximate inverse, and that enhanced by the DNN module trained based on (10) for the test trajectory $y_d(t) = \sin\left(\frac{4\pi}{23}t\right) + \cos\left(\frac{4\pi}{21}t\right) - 1$.

subsections, we adapt the DNN-enhanced architecture to non-minimum phase systems.

C. The Proposed Approach: DNN Input Modification

We propose a learning approach that achieves stability (O1) and performance enhancement (O3) through modifying the DNN input selection. We first consider the linear baseline system (1), for which the exact inverse is

$$H^{-1}(z) = \frac{U(z)}{Y_d(z)} = \frac{D(z)}{N(z)} = \frac{\sum_{i=0}^{n} \beta_i z^i}{1 + \sum_{i=1}^{n-r} \alpha_i z^i}, \tag{7}$$

where $Y_d(z)$ is the z-transform of the desired output $y_d(k)$. For non-minimum phase systems, at least one root of the denominator $N(z)$ is outside of the unit circle, which is the source of instability that prevents the direct application of the inverse learning scheme in (6) from being effective (see the right panel of Fig. 2). If the input of the DNN module is selected such that the unstable dynamics associated with $N(z)$ cannot be learned, then the instability-induced numerical issues would not arise. By applying the inverse z-transforms to (7), it can be shown that,

$$u(k) = \sum_{i=0}^{n} \beta_i y_d(k+i) - \sum_{i=1}^{n-r} \alpha_i u(k+i) \tag{8}$$

or

$$u(k) = F(\underbrace{y_d(k:k{+}n)}_{\text{from } D(z)}, \underbrace{u(k{+}1:k{+}n{-}r)}_{\text{from } N(z)}), \tag{9}$$

where $F(\cdot)$ denotes a generic multi-variable function. From (9), it can be seen that the unstable dynamics associated with $N(z)$ are reflected in the dependencies of $u(k)$ through the sequence of reference signals $u(k{+}1:k{+}n{-}r)$.

Proposed Input-Output Selection. Based on (9), we propose the following DNN input-output selection:

$$\mathcal{I} = \{y_d(k:k{+}n)\} \text{ and } \mathcal{O} = \{u(k)\}, \tag{10}$$

where the sequence of u is removed from the input $\mathcal{I}$ to prevent the DNN module from learning the unstable dynamics associated with $N(z)$.

The efficacy of the DNN input-output selection based on (10) is illustrated with the running example (see Fig. 3). By comparing the DNN-enhanced response (blue) with the baseline response (gray), it can be seen that the proposed modification of the DNN input selection does lead to improved tracking performance. From a further comparison with Fig. 2, it is interesting to see that, for the non-minimum phase system, *the DNN trained with less inputs leads to a better performance*. It should be noted that, in contrast to typical DNN applications such as image classification, for control applications, the training objective (e.g., minimizing regression error) and performance objective (e.g., minimizing tracking error) may not coincide. Consequently, DNN training algorithms may not effectively phase out unnecessary input dimensions to achieve a good performance.

In order to further examine the effectiveness of the proposed approach, the proposed approach is also compared against a typical model-based inverse approximation for linear systems – the zero-order series (ZOS) approximation [7]. In the ZOS approximation approach, the transfer function polynomials associated with the unstable zero dynamics are approximated by zero-order Taylor series [7]. In particular, the generic transfer function representation in (1) can be expressed as $H(z) = \frac{N_s(z)N_u(z)}{D(z)}$, where $N_s(z)$ and $N_u(z)$ correspond to the numerator polynomial with stable and unstable zeros respectively; the ZOS approximate inverse is

$$\widetilde{H}^{-1}(z) = \frac{D(z)}{N_u(z)|_{z=1} N_s(z)}. \tag{11}$$

Given the backward-driving system model (5), the ZOS approximate inverse of the system is derived to be

$$\widetilde{H}^{-1}_{\text{backward,ZOS}}(z) = \frac{z^2 - 1.602z + 0.6376}{0.0361}. \tag{12}$$

The response of the ZOS-enhanced system, which is the cascade of $\widetilde{H}^{-1}_{\text{backward,ZOS}}(z)H(z)$, is presented as the green curve in Fig. 3. From this example, it can be seen that as compared to the ZOS approximation approach (green), *the proposed DNN-based learning approach (blue) is capable of achieving a comparable or even better performance without relying on detailed dynamic models of the baseline system*. In addition to the input-output data of the baseline system, the only information required to train the network in the proposed approach is the order of the baseline system.

D. Stability of the Proposed Approach

The proposed approach is derived from the transfer function formulation to guarantee stability for LTI systems; for nonlinear systems, stability can be proved from properties of the regression model. With the assumptions A1, A3, and the additional mild assumptions on the learning module,

A3a. the FNN is properly trained (i.e., has finite weights and bias), and

A3b. the FNN activation functions $\sigma(\cdot)$ are continuous,

we prove the following lemma:

Lemma 1. Stability. For the inversion-based learning control architecture in Fig. 1 and nonlinear system (2), under the assumptions A1 and A3, the learning module input-output selection in (10) ensures that the overall control system (from y_d to y) is input-to-output stable.

Proof. From (10), the learning module approximates a mapping from $\mathcal{I} = \{y_d(k:k{+}n)\}$ to $\mathcal{O} = \{u(k)\}$. For a

typical L-layer FNN with $n+1$ inputs and 1 output, by denoting $\zeta_0(k) = \begin{bmatrix} y_d(k) & y_d(k+1) & \cdots & y_d(k+n) \end{bmatrix}^\intercal$ as the network input at time k, the output of a neuron i in a hidden layer l, denoted by $\zeta_{l,i}(k)$, can be expressed as

$$\zeta_{l,i}(k) = \sigma\left(\sum_{j=1}^{N_{l-1}} w_{l,ij}\zeta_{l-1,j}(k) + b_{l,i}\right), \qquad (13)$$

where $\sigma(\cdot)$ is the activation function, $l \in \mathbb{N}$, $1 \le l \le L-1$, is the layer index, $N_l \in \mathbb{N}$ is the number of neurons in layer l, $\zeta_l \in \mathbb{R}^{N_l}$ is the output of the layer l, $w_l \in \mathbb{R}^{N_l \times N_{l-1}}$ and $b_l \in \mathbb{R}^{N_l}$ are the weights and bias associated with layer l, $\zeta_{l,i}$ and $\zeta_{l-1,j}$ are respectively the i^{th} element of the vector ζ_l and the j^{th} element of the vector ζ_{l-1}, $w_{l,ij}$ is the i^{th} row and j^{th} column element of the matrix w_l, and $b_{l,i}$ is i^{th} element of the vector b_l. The output of the network $\widetilde{F}(\zeta_0(k))$ is

$$\widetilde{F}(\zeta_0(k)) = \sum_{j=1}^{N_{L-1}} w_{L,1j}\zeta_{L-1,j}(k) + b_{L,1}. \qquad (14)$$

By assumptions A3a and A3b, the network parameters (w, b) are bounded and σ is continuous; hence, the output of each neuron i in layer l (i.e., $\zeta_{l,i}$) is continuous in ζ_0. Moreover, since $\widetilde{F}$ is a composition of $\zeta_{l,i}$, $\widetilde{F}$ is also continuous in ζ_0. Since every continuous function from a compact space into a metric space is bounded, the network output $u(k)$ is bounded for bounded input $\zeta_0(k)$. Furthermore, by assumption A1, the baseline system is input-to-output stable; hence, for an arbitrary bounded desired trajectory y_d, the output $u(k)$ of the FNN is bounded, and the overall system from y_d to y is input-to-output stable. □

E. Insights on Performance Enhancement

Given that the stability (O1) is achieved through the input selection of the learning module in (10), in this subsection, we address the performance enhancement objective (O3). In particular, for system (1), we show that the learning module trained with the proposed input-output selection in (10) is able to approximate an inverse mapping that enhances the baseline system performance.

Insight 1. Approximate Inverse Learning. For system (1), given sufficiently high sampling rate, the input selection in (10) enables the FNN to learn an approximate inverse, where the sequence of reference components in the input of the exact inverse map is approximated by $u(k)$.

In order to clarify the insight above, we first present a toy example. Consider a linear function with input $\boldsymbol{\xi} = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_m \end{bmatrix}^\intercal \in \mathbb{R}^m$ and output $\upsilon \in \mathbb{R}$: $\upsilon = F_1(\boldsymbol{\xi})$. If a particular input ξ_p is correlated to the output υ by the linear function $\upsilon = F_2(\xi_p)$ and $\frac{\partial F_1}{\partial \xi_p} \neq \frac{dF_2}{d\xi_p}$, then υ can be re-expressed as a linear function of the remaining components of the vector $\boldsymbol{\xi}$: $\upsilon = F_3(\widetilde{\boldsymbol{\xi}})$, where $\widetilde{\boldsymbol{\xi}} := \begin{bmatrix} \xi_1 & \dots & \xi_{p-1} & \xi_{p+1} & \dots & \xi_m \end{bmatrix}^\intercal$. This implies that a regression model for the output υ can be found with either $\boldsymbol{\xi}$ or $\widetilde{\boldsymbol{\xi}}$ as the input. This simple discussion can be generalized to the case when the removal of the dimension ξ_p *does not* lead to a one-to-many map from $\widetilde{\boldsymbol{\xi}}$ to υ; a regression model can be constructed in a lower-dimensional input space to uniquely determine the output υ for a given $\widetilde{\boldsymbol{\xi}}$. An illustration is shown in Fig. 4. When a component of the input vector is related to the output by the function F_2, the data points generated by the map F_1 are restricted to the intersection of the manifolds defined by F_1 and F_2. When removing ξ_p from the input of the dataset, the data points are projected onto a lower-dimensional space that is orthogonal to ξ_p.

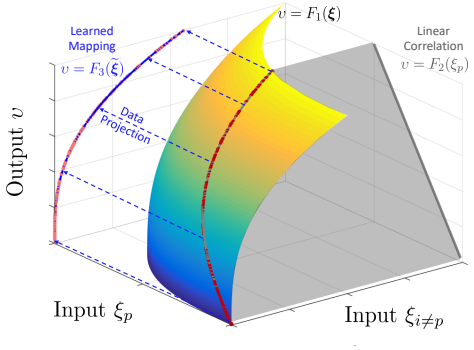

Fig. 4. Illustration of data projection in the approximate inverse learning.

On the training side, since an arbitrary smooth trajectory can be expressed as a superposition of sinusoidal functions, without loss of generality, we consider a single sinusoidal training trajectory of the form $u(t) = A\sin(\frac{2\pi}{T}t) + b$, where t denotes the continuous time. For either system (1) or (2), it can be shown using Taylor series expansion of $u(t)$ that at time step k, future references $u(k+p)$ for $p = 1, ..., n-r$ can be related to the current reference $u(k)$ by

$$u(k+p) = u(k) + \sum_{i=1}^{\infty}\left(\frac{2\pi p\Delta t}{T}\right)^i c_i(k), \qquad (15)$$

where Δt denotes the sampling time and $|c_i(k)| \le \frac{A}{i!}$. Given that p is typically a small positive number bounded by $n-r$, if Δt is sufficiently small as compared to the period of the training trajectory T, then from (15), at a particular time step k, the future reference $u(k+p)$ and $u(k)$ are approximately correlated by the identity function.

Given this approximate correlation and by the result above, it can be seen that though dependent reference components are removed from the FNN input based on the selection in Eqn. (10), the FNN can still learn a regression model to output a reference u that best matches that in the training dataset. Hence, the FNN acts as an approximate inverse from output y to input u that reduces the error between y_d and y. From (15), the error involved in consecutive reference signal approximations and the inherent regression error in the learned inverse model is smaller for smaller Δt (i.e., higher sampling frequency).

For nonlinear systems, to achieve exact tracking, the learning module should model the output equation of the inverse dynamics, and consequently, $u(k)$ should be a nonlinear function of $x(k)$ and $y_d(k+r)$; however, for non-minimum phase systems, the internal instability of $x(k)$ can cause numerical issues [11]. One trivial solution is to remove the state

$x(k)$ from the DNN input and use $\mathcal{I} = \{y_d(k+r)\}$. Instead, we suggest to use the same proposed input selection in (10). A rough conjecture on this selection is as follows. Since smooth nonlinear systems can be approximated by piecewise affine/linear systems with arbitrary accuracy [19], one can always represent the considered, smooth nonlinear system as an aggregation of local, n-dimensional, affine/linear models defined on local regions of a cover/partition of the nonlinear system state space. Since all models have order n, by following the derivation in Section III-C for each local model, one can reach the same input selection as in (10) for each local model. Thus, it is reasonable to select the inputs for the DNN as in (10) for nonlinear systems. The effectiveness of the proposed input-output selection for nonlinear systems will be verified through both (nonlinear system) simulations and experiments (see Sections IV and V).

F. Connection with the ZOS Approach

From Section III-E, the proposed input selection in (10) is linked to the input selection based on the exact inverse in (6) through the approximations of the sequence of reference signals. In this subsection, we will show a connection between the proposed approach and the model-based ZOS approach in the literature.

Insight 2. Connection with ZOS. The approximation of the sequence of reference signals with the current reference $u(k)$ is equivalent to approximating the numerator of the transfer function $N(z)$ in (1) with $N(z)|_{z=1}$. Through the modified input selection in (10), the proposed learning approach achieves stability (O1) and performance enhancement (O3) in a similar manner to the model-based ZOS approach in (11).

For the exact inverse in (7), the corresponding representation in the time-domain is shown in (8). When the future references $u(k+i)$ for $i = 1, ..., n-r$ are approximated by $u(k)$, then

$$\sum_{i=0}^{n} \beta_i y(k+i) \approx \left(1 + \sum_{i=1}^{n-r} \alpha_i\right) u(k). \tag{16}$$

By applying the z-transform to (16), we obtain:

$$H^{-1}(z) \approx \frac{\sum_{i=0}^{n} \beta_i z^i}{1 + \sum_{i=1}^{n-r} \alpha_i} = \frac{D(z)}{N(z)|_{z=1}}. \tag{17}$$

By comparing this expression with the approximation applied in the ZOS approach in (11), it can be seen that they both achieve stability through the approximation of unstable zero dynamics at $z = 1$, and compensate for the delays introduced by the dynamics associated with the poles (i.e., $D(z)$) to improve the tracking performance.

It should be noted that the generalizability of the FNN to new trajectories depends on the invariance of the phase and magnitude errors of the transfer function $\frac{Y(z)}{Y_d(z)} = \frac{N(z)}{N(z)|_{z=1}}$ with respect to the frequency of the desired trajectory; it can be shown that the generalizability is better if the location of the zeros, the roots of $N(z)$, are further away from $z = 1$.

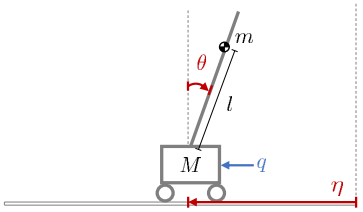

Fig. 5. Schematic diagram of the pendulum-cart system.

IV. Simulation Results

An inverted pendulum on a cart system (referred to as the pendulum-cart system) is a benchmark example used for exploring control techniques for non-minimum phase systems. In addition to being nonlinear, due to the presence of the inverted pendulum, the system has unstable forward and inverse dynamics, which make the tracking control of the cart position a non-trivial problem. In this section, we use this example to examine the efficacy of the proposed approach for enhancing the tracking performance of nonlinear systems.

A. Dynamic Model and Control Objective

As illustrated in Fig. 5, the pendulum-cart system has two degrees of freedom – the cart linear position η and the pendulum angular position θ. By applying the Lagrangian's equations, a dynamics model of the pendulum-cart system can be obtained [20]:

$$\begin{aligned} \ddot{\eta} &= \frac{q + mg\sin\theta\cos\theta - ml\dot{\theta}^2\sin\theta}{M + m\sin^2\theta} \\ \ddot{\theta} &= \frac{q\cos\theta + (M+m)g\sin\theta - ml\dot{\theta}^2\sin\theta\cos\theta}{l\left(M + m\sin^2\theta\right)}, \end{aligned} \tag{18}$$

where M and m are the masses of the cart and the pendulum, l is the effective length of the pendulum relative to the pivot point, and q is the force applied on the cart. By defining the state of the system as $x = \begin{bmatrix}\eta & \dot{\eta} & \theta & \dot{\theta}\end{bmatrix}^\intercal$, its input as the force q, and its output as the full state $y = x$, the nonlinear state-space representation of the pendulum-cart system can be written in the control affine form:

$$\begin{aligned} \dot{x}(t) &= f_1(x_2(t), x_3(t), x_4(t)) + g_1(x_3(t))\, q(t) \\ y(t) &= x(t), \end{aligned} \tag{19}$$

where $x_2(t) = \dot{\eta}(t)$, $x_3(t) = \theta(t)$, and $x_4(t) = \dot{\theta}(t)$. The control objective is to compute a control input q such that the cart tracks a desired trajectory $\eta_d(t)$ while the pendulum is balanced at the upright position. The desired output is $y_d(t) = \begin{bmatrix}\eta_d(t) & \dot{\eta}_d(t) & 0 & 0\end{bmatrix}^\intercal$.

B. Baseline System Controller

Through linearizing the system (19) at the equilibrium point ($\eta = \eta_d$, $\dot{\eta} = 0$, $\theta = 0$, $\dot{\theta} = 0$, and $q = 0$) and using the pole placement technique, a feedback control law can be found to stabilize the nonlinear pendulum-cart system (19) with the pendulum held at the upright position: $q(t) = K_1(u(t) - y(t))$, where $K_1 \in \mathbb{R}^4$ is the controller gains and $u = \begin{bmatrix}\eta_r & \dot{\eta}_r & 0 & 0\end{bmatrix} \in \mathbb{R}^4$ is the reference sent to the baseline system (Fig. 1). In accordance with the experimental setup in Section V, for the

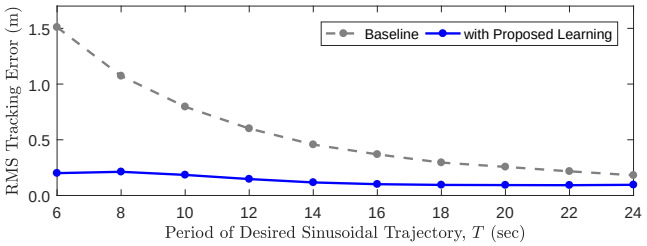

Fig. 6. The RMS tracking error of the baseline and the learning-enhanced system for a set of desired trajectories of the form $\eta_d(t) = \frac{5}{2}\sin\left(\frac{2\pi}{T}t\right)$, where the periods T are chosen to be different from that used for training. A visualization of the result for the case $T = 12$ s can be found at this link: http://tiny.cc/fq0mny.

simulations, we choose $M = 1.07$ kg, $m = 0.23$ kg, and $l = 0.33$ m. By placing the closed-loop poles at $\{-1, -2, -3, -4\}$, the controller gains are set to be $K_1 = \begin{bmatrix}-0.8678 & -1.808 & 25.46 & 4.140\end{bmatrix}$.

C. Learning Module

A learning module, pre-cascaded to the baseline system as in Fig. 1, is designed based on (10) to enhance the performance of the cart position tracking. Given the desired trajectory η_d (a component of y_d), at a time instance k, the learning module computes an adjusted reference signal η_r (a component of u) to be sent to the baseline system. The $\dot{\eta}_r$ component in u is generated from the η_r trajectory. A FNN with 2 hidden layers of 5 hyperbolic tangent neurons is used for learning the approximate inverse of the baseline system. Assuming that the baseline system succeeds to stabilize the pendulum at the upright position, then from (18), the dynamics associated with η may be approximated by a second order system; by (10), the input and output of the learning module are selected to be $\mathcal{I} = \{\eta_d(k:k{+}2)\}$ and $\mathcal{O} = \{\eta_r(k)\}$. The module is trained on 30 sinusoidal trajectories with different combinations of amplitudes $\{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ m and periods $\{5, 10, 15, 20, 25\}$ s. The training algorithm and validation process are similar to that described in [11]. The learning module is executed at sampling intervals of 0.015 s.

D. Results

The tracking performance of the baseline system and the learning-enhanced system are compared in Fig. 6 for a set of sinusoidal trajectories that are not used for training. As can be seen from Fig. 6, though the baseline system is capable of stabilizing the pendulum-cart system, the tracking error increases with the decreasing period of desired trajectories. In contrast, when the learning module is added to the baseline system, the tracking error is approximately maintained at a smaller constant value over the range of trajectory periods covered by the training dataset, which shows the generalizing capabilities of the learning approach. The reduction of RMS error achieved by the learning module ranges from 47% to 87% for the set of testing trajectories.

Fig. 7 shows the adverse impact when a single past reference is included in the proposed input selection of the learning module; i.e, when $\mathcal{I} = \{\eta_d(k:k{+}2), u(k{-}1)\}$. It can be seen that when the additional information is included,

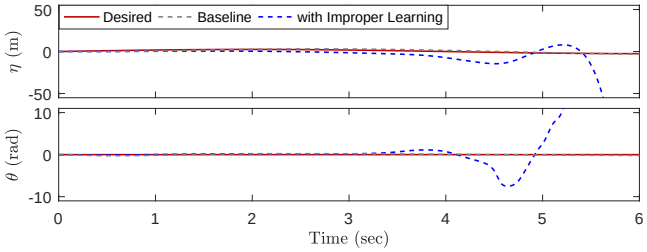

Fig. 7. Illustration of the adverse effect caused by the inclusion of an additional reference component in the input $\mathcal{I}$ of the learning module.

the pendulum-cart system quickly becomes unstable. Thus, for non-minimum phase systems, the selection of inputs in model learning is essential, and the inclusion of unnecessary inputs can prevent not only the learning approach but also the baseline system from being functional.

V. Experimental Results

The effectiveness of the proposed approach is demonstrated on a Quanser pendulum-cart setup [21].

A. Experiment Setup

The experiment setup is similar to that of the simulation (see Section IV-A), except that the input force q is replaced by the input voltage v to a DC motor actuating the cart. By using a simple voltage-to-force model $q(t) = -7.74\dot{\eta}(t) + 1.73v(t)$ [21], system (19) can be re-expressed as

$$\begin{aligned}\dot{x}(t) &= f_2(x_2(t), x_3(t), x_4(t)) + g_2(x_3(t))\, v(t)\\ y(t) &= x(t),\end{aligned} \tag{20}$$

where the measured states are $\eta(t)$ and $\theta(t)$, and $x(t)$ is estimated through the design of a full-state observer. A controller $v(t) = K_2(u(t) - y(t))$ with $K_2 = \begin{bmatrix}-105.6 & -55.04 & 130.7 & 23.67\end{bmatrix}$ is designed by placing closed-loop poles at $\{-7, -8, -9, -10\}$ and is run at 1 kHz.

We compare the proposed learning approach with the baseline system and the model-based ZOS approach. In the experiments, the learning module is run at 70 Hz [10]; the design and training procedure for the inverse-learning module are similar to that of the simulations (see Section IV-C). The training dataset is constructed from 18 sinusoidal trajectories with combinations of amplitudes $\{0.04, 0.06, 0.08\}$ m and periods $\{5, 6, 7, 8, 9, 10\}$ s. On the other hand, the ZOS approach is implemented based on the linearized state-space model of system (20) that is used for designing the baseline system. From the linearized system, a discrete-time transfer function from the reference η_r to the output η can be determined. By applying (11), the ZOS approximate inverse of the response from η_r to η is obtained:

$$\widetilde{H}^{-1}_{\text{ZOS}}(z) = \frac{z^4 - 3.5217z^3 + 4.6504z^2 - 2.7290z + 0.6005}{0.00137z^2 - 0.0001066z - 0.001066}. \tag{21}$$

For the experimental comparison, the ZOS approximate inverse in (21) replaces the learning module in Fig. 1, and generates reference signals η_r based on η_d at 70 Hz.

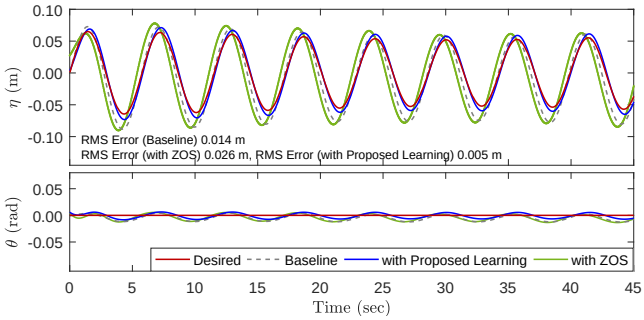

Fig. 8. The cart position η and the pendulum angle θ of the baseline, the ZOS, and the proposed learning-based systems on a test trajectory $\eta_d(t) = \frac{117}{2000}\sin(\frac{2\pi}{5}t) + \frac{13}{2000}\sin(\frac{4\pi}{11}t)$.

B. Results

Fig. 8 shows the comparison of the tracking performance of the baseline, the ZOS, and the proposed learning-based systems. The stability objective is achieved by all three systems, and the pendulum position is kept approximately at the upright position ($\theta = 0$ rad). By comparing the cart position $\eta(t)$ of the baseline (gray) and the proposed learning-based system (blue), the addition of the learning module in the proposed approach effectively compensates for the phase and the magnitude errors in the baseline system response. The learning module reduces the cart tracking RMS error from approximately 0.014 m to approximately 0.005 m, which corresponds to a 60% reduction.

In contrast, by comparing the $\eta(t)$ of the ZOS approach (green) with the baseline response (gray), the addition of the approximate inversion led to worse tracking performance. Though the linearized state-space model is sufficiently accurate for deriving a baseline controller that stabilizes the pendulum-cart system, the application of the model-based system inversion approach requires a much more detailed and accurate system model. In comparison, despite the nonlinearities in the system dynamics, the proposed learning-based approach can be directly and effectively implemented based on the input-output data.

VI. Conclusions

In this paper, an inversion-based learning approach is presented to enhance the impromptu tracking performance of non-minimum phase systems. In particular, the learning module approximates the inverse dynamics of a stabilized feedback control loop, and stability of the approximate inverse learning is ensured through learning-module input selection. As demonstrated in simulations and experiments of nonlinear systems, requiring only input-output data of the baseline system, the proposed approach leads to better performance as compared with that of the ZOS approximate inverse, one of the typical model-based approaches in the literature. This is relevant for robotics as many robotics control systems are non-minimum phase.

References

[1] G. M. Clayton, S. Tien, K. K. Leang, Q. Zou, and S. Devasia, "A review of feedforward control approaches in nanopositioning for high-speed spm," *Journal of Dynamic Systems, Measurement, and Control*, vol. 131(6), pp. (061 101) 1–19, 2009.

[2] S. A. Al-Hiddabi and N. H. McClamroch, "Tracking and maneuver regulation control for nonlinear nonminimum phase systems: Application to flight control," *IEEE Trans. on Control Systems Technology*, vol. 10(6), pp. 780–792, 2002.

[3] A. De Luca, P. Lucibello *et al.*, "Inversion techniques for trajectory control of flexible robot arms," *Journal of Field Robotics*, vol. 6(4), pp. 325–344, 1989.

[4] J. Levin, N. O. Perez-Arancibia, P. A. Ioannou, and T. Tsao, "A neural-networks-based adaptive disturbance rejection method and its application to the control of hard disk drives," *IEEE Trans. on Magnetics*, vol. 45(5), pp. 2140–2150, 2009.

[5] J. B. Hoagg and D. S. Bernstein, "Nonminimum-phase zeros – much to do about nothing – classical control revisited Part II," *IEEE Control Systems*, vol. 27(3), pp. 45–57, 2007.

[6] S. Devasia, D. Chen, and B. Paden, "Nonlinear inversion-based output tracking," *IEEE Trans. on Automatic Control*, vol. 41(7), pp. 930–942, 1996.

[7] B. P. Rigney, L. Y. Pao, and D. A. Lawrence, "Nonminimum phase dynamic inversion for settle time applications," *IEEE Trans. on Control Systems Technology*, vol. 17(5), pp. 989–1005, 2009.

[8] Y. Zhang, Q. Zhu, and R. Xiong, "Pre-action and stable inversion based precise tracking for non-minimum phase system," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2016, pp. 5682–5687.

[9] L. Benvenuti, M. D. Di Benedetto, and J. W. Grizzle, "Approximate output tracking for nonlinear non-minimum phase systems with an application to flight control," *Intl. Journal of Robust and Nonlinear Control*, vol. 4, no. 3, pp. 397–414, 1994.

[10] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 5183–5189.

[11] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2017, Accepted. Available at arXiv: 1705.10932.

[12] A. S. Polydoros, L. Nalpantidis, and V. Krüger, "Real-time deep learning of robotic manipulator inverse dynamics," in *Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 3442–3448.

[13] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf, "Learning inverse dynamics: a comparison," in *European Symposium on Artificial Neural Networks*, no. EPFL-CONF-175477, 2008.

[14] C. Williams, S. Klanke, S. Vijayakumar, and K. M. Chai, "Multi-task gaussian process learning of robot inverse dynamics," in *Advances in Neural Information Processing Systems*, 2009, pp. 265–272.

[15] S. Jung and S. S. Kim, "Control experiment of a wheel-driven mobile inverted pendulum using neural network," *IEEE Trans. on Control Systems Technology*, vol. 16(2), pp. 297–303, 2008.

[16] A. de Almeida Neto, W. R. Neto, L. C. S. Góes, and C. Nascimento, "Feedback-error-learning for controlling a flexible link," in *Proc. of the IEEE Brazilian Symposium on Neural Networks*, 2000, pp. 273–278.

[17] F.-C. Chen and H. K. Khalil, "Adaptive control of a class of nonlinear discrete-time systems using neural networks," *IEEE Trans. on Automatic Control*, vol. 40(5), pp. 791–801, 1995.

[18] E. D. Sontag and Y. Wang, "Notions of input to output stability," *Systems & Control Letters*, vol. 38(4), pp. 235–248, 1999.

[19] M. K. Helwa and P. E. Caines, "Epsilon controllability of nonlinear systems on polytopes," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2015, pp. 252–257.

[20] A. M. Bloch, N. E. Leonard, and J. E. Marsden, "Controlled Lagrangians and the stabilization of mechanical systems I: The first matching theorem," *IEEE Trans. on Automatic Control*, vol. 45(12), pp. 2253–2270, 2000.

[21] Quanser Consulting Inc., "IP02 self-erecting inverted pendulum user's guide," 1996, Available at: http://www.mecatronica.eesc.usp.br/wiki/upload/1/11/Manual_SelfErecting.pdf.