Inverse Models for Trajectory Control Aided by Data, Machine Learning Models, and GPUs

Liangwu Yan

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2024

Reading Committee:

Santosh Devasia, Chair

Joseph L. Garbini

Xu Chen

Program Authorized to Offer Degree:
Department of Mechanical Engineering

University of Washington

Abstract

Inverse Models for Trajectory Control Aided by Data, Machine Learning Models, and GPUs

Liangwu Yan

Chair of the Supervisory Committee:
Santosh Devasia
Department of Mechanical Engineering

This dissertation investigates how the easy access to large amount of data and cheap computation power will benefit the usage of the inverse models for trajectory control. In general, model-based inversion methods have been used to achieve high precision trajectory tracking in the past, and iterative methods with inverse models tend to achieve some of the highest precision possible for output tracking. However, it is challenging to get plant models for many practical systems such as robots with complex environmental interactions. In this context, this thesis explores three scenarios using both theory and experiments: (1) the use of data-based inverse models for improving precision in iterative control, (2) identifying the type of observables needed to develop machine-learning-based (neural-net-based) inverse models to enable precise tracking, and (3) the use of inverse-models in improving performance of model predictive path integral control (MPPI) to regulate the trajectory, relying on parallel computation powered by graphic processing units (GPUs).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would never forget the graduate school journey accompanied by my advisor Professor Santosh Devasia for his guidance and support all the way. I remember that Professor Devasia comforted me when I received my first reject & resubmit. I was told by Professor Devasia to be kind when I started my first paper review for journals. I learnt a lot from presenting in the lab meetings, which helped hone my slides and presentation skills. I was supported to join academic conferences and do internship to explore outside the lab as well. Overall, it's a challenging yet memorable journey that prepares me for the future, and it cannot be made without the guidance and support from my advisor Professor Santosh Devasia.

I would like to thank the rest of my committee, Professor Xu Chen, Professor Karen Leung, and Professor Joseph Garbini for their valuable advice and support.

I wish to thank the collaborating authors for their help in my first journal publication. I wish to thank Professor Parker Owan for building the MASCOT robot that I used in my first project and the helpful research discussions.

I would like to thank Mr. Michael Xu for encouraging me to study abroad when I was an undergraduate at Shanghai Jiao Tong.

I wish to thank the team at the Precision Controls Lab for their helpful discussions, encouragement, experiment setup and Robot Operating System (ROS) knowledge.

I would also like to acknowledge the National Science Foundation for supporting me through a major part of my graduate school through the NSF Grant CMMI 1824660.

Lastly, I want to thank my parents for their consistent and persistent love.

DEDICATION

to my parents, Youguo Yan & Jianghua Shi.

Chapter 1

INTRODUCTION

1.1 Research goal

A lot of areas are being boosted by data-based methods, e.g., Alex-net for computer vision [5], Chat-GPT for language models developed by Open AI [6], machine-learning-based recommender systems [7], data-based modeling and learning-based control in dynamic systems [8, 9, 10] (the focused area of this thesis), etc. Diverse machine learning models are developed to handle the tasks in different scenarios, e.g., regression model [11], neural-net [12], clustering model [13], etc. What's more, GPUs can offer speed-up in dynamic systems' simulations [14, 15] and machine learning models' training and prediction [16, 17]. This motivates the current thesis to use data-based methods to improve performance of trajectory tracking and control.

It is noted that model-based inversion methods have been used to achieve high precision trajectory tracking in the past, and iterative methods with inverse models tend to achieve some of the highest precision possible for output tracking. However, it is challenging to get plant models, e.g. of series elastic actuators (SEA) robots when interacting with environments, see Fig. 1.1.

In this context, this thesis explores three scenarios: (**SC1**) the use of data-based inverse models for improving precision in iterative control, (**SC2**) identifying the type of information needed to develop data-based inverse models, and (**SC3**) the potential use of inverse-models in improving performance of model predictive path integral control (MPPI) to regulate the trajectory in uncertain and complex environments, relying on parallel computation powered by graphic processing units (GPUs). These scenarios and the associated main contributions are described below.

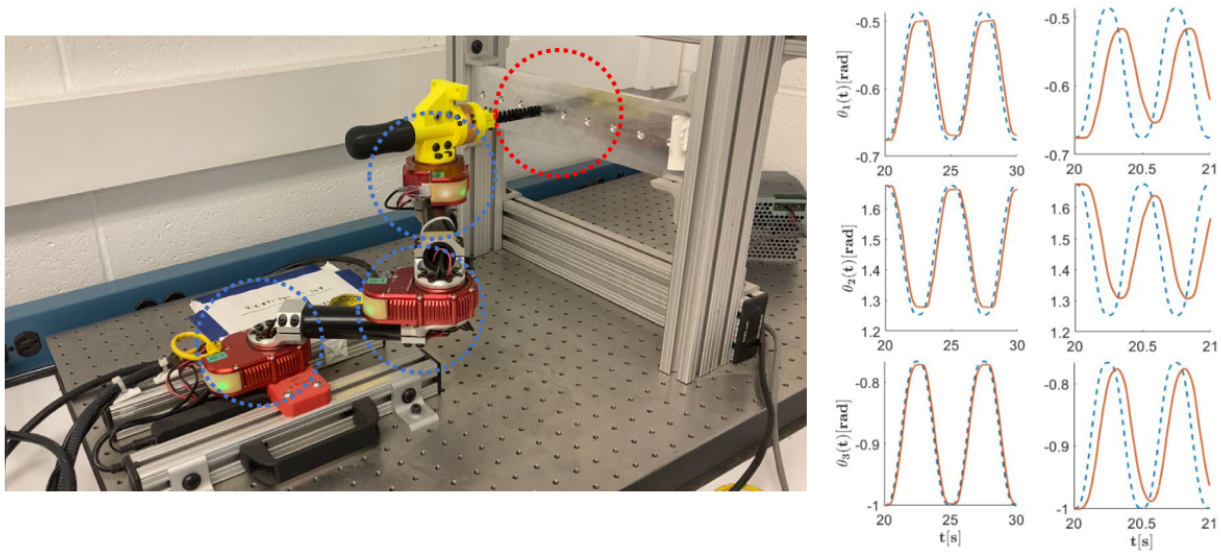

Figure 1.1: (Left: SEA-driven robot for pilot-hole cleaning) Challenge in modeling SEA-coupled dynamics for each joint (circled in blue) and the interaction dynamics for hole-cleaning motions (circled in red) that happen between the flexible brush and the rigid plate. (Middle: (slow) 5 s per cycle tracking references & Right: (fast) 0.5 s per cycle tracking references with plots of the actual joint angles (solid orange lines) and the tracking references (dashed blue lines).) Tracking errors for three joints are large if tracking references are fast.

1.2 SC1: Can precise tracking be achieved in the context of unknown interaction dynamics (challenging to model from First Principles)?

Lightweight, relatively-small, SEA robots are well suited for automating manufacturing tasks in confined spaces such as an aircraft wing or tail, e.g., for cleaning pilot holes (used for fixtures) during aircraft assembly. Often these are hard to reach spaces, and workers need to crawl through tight spaces and manual operations can be ergonomically challenging. The robot needs to be lightweight (around 20 pounds or less) for easy placement, often by hand. Moreover, direct estimates of the joint forces found by measuring the deformation of the elastic element in SEAs [18, 19], along with low-impedance control can limit the forces applied by the robot when maneuvering under uncertainty in confined spaces, which can help avoid po-

tential damage to the workpiece and costly repairs. Nevertheless, this increased control over forces during maneuvering in uncertain environments comes at the cost of lower positioning precision of lightweight robots [20], even at the final operating points where the environmental uncertainty might be lower. However, precision positioning, around an operating point is desirable since it can enable the use of lightweight SEA robots in manufacturing tasks in hard-to-reach, confined spaces. Such precision in conjunction with compliance (which avoids damage) can be especially beneficial for contact-type applications such as cleaning of holes. The problem in achieving precision is that the flexural systems in lightweight SEA robots result in non-minimum phase dynamics, and high gains (for improved performance) can lead to instability. Therefore, there are limits to the achievable precision with feedback [21]. In general, model-based methods, such as inversion to find feedforward inputs to augment the feedback, can be used to improve performance of robots, e.g., [22, 23, 24, 25]. A challenge to accurate modeling is the difficulty in capturing nonlinearities and contact-related effects in SEA robots, e.g., [19, 26]. Therefore, tracking errors with such model-inversion-based approaches can be large if there are significant modeling errors. This motivates the first main contribution (**MC1**) for improving precision-positioning of SEA robots at different operating points (around which the dynamics can be linearized), even in the presence of modeling uncertainties. While **MC1** is shown to be well-suited to improve precision around an operating point, it is not suited for applications where large robot motions are required. In such cases, the nonlinearity can be significant and alternative approaches such as Lyapunov-based iterative methods [27] or iterative methods for model-based impedance control [28] could be considered. Moreover, when the signals are periodic and band-limited, the frequency-domain regression method developed in **MC1** could be extended to the nonlinear case using generalized frequency-response functions as in [29].

The goal in **SC1** is to improve the positioning precision of lightweight robots with series elastic actuators (SEAs). In particular, this part of the thesis seeks to improve the precision of SEA robots around specified operating points, through a multi-input multi-output (MIMO), iterative learning control (ILC) approach. **MC1** are to (i) introduce an input-weighted

complex kernel to estimate local MIMO models using complex Gaussian process regression (c-GPR) (ii) develop Geršgorin-theorem-based conditions on the iteration gains for ensuring ILC convergence to precision within noise-related limits, even with errors in the estimated model; and (iii) demonstrate precision positioning with an experimental SEA robot. Additionally, comparative experimental results, with and without ILC, show around 90% improvement in the positioning precision (close to the repeatability limit of the robot) and a 10-times increase in the SEA robot's operating speed with the use of the MIMO ILC in **MC1**.

1.3 SC2: Can precise tracking be achieved with hidden-state dependency?

The frequency method in **MC1** requires future tracking references that may not be always accessible. With increasing ease of collecting data and low-cost storage, there is increasing interest in using data-enabled methods developing models for prediction and control, [30, 31, 32, 33, 34, 35, 36]. Such models can be optimized to best fit the data and methods are also available to estimate the error bounds on the predictions, e.g., using predicted time derivatives of the observables [31]. However, the conditions under which such data-enabled models can achieve sufficient precision remain unclear. A major challenge is that the model (i.e., the relationship between the input and the measurable outputs) can be dependent on the system's internal states, which are hidden in the sense that they are not directly measured, nor inferred using standard observer designs since they require prior knowledge of the system dynamics.

The second main contribution (**MC2.1**,**MC2.2**) is to show that, irrespective of the selected model, removing the hidden-state dependence and achieving a desired precision of inverse operators require (i) a sufficiently-long past history of the output and (ii) sufficiently-precise estimates of the output's instantaneous time derivatives that are necessary and sufficient for linear systems, and under some conditions, for nonlinear systems. This insight, about the required observables (output history and derivative) for removing the hidden-state dependence and achieving precision, is used to develop a data-enabled algorithm to learn the inverse operator for multi-input multi-output square systems. **MC2.1** developed the anal-

ysis for single-input-single-output (SISO) linear systems and **MC2.2** extends the results to multi-input-multi-output (MIMO) square systems.

Simulation examples are used to illustrate that neural nets (with universal approximation property) can learn the inverse operator with sufficient precision only if the required observables, identified in this work, are included in training.

1.4 SC3: Can safe and optimal trajectory regulation be achieved with dynamic obstacles in real-time?

The difference in Scenario 3 lies in that the tracking references make less sense when there are dynamic obstacles in the environment, since a good reference may be a bad one in the future time. Therefore, the selection of the trajectory becomes important. For example, the success of the model predictive path integral control (MPPI) approach depends on the appropriate selection of the input distribution used for sampling. However, it can be challenging to select inputs that satisfy output constraints in dynamic environments.

With the goal of increasing rollouts that satisfy output constraints, this part of the thesis proposes the sampling of trajectories directly from the output space. Then, the proposed output-sampling-based MPPI (o-MPPI) approach uses the inverse model $G^{-1}(\cdot)$ to map outputs to inputs rather than the traditional approach in MPPI of using the input-to-output forward model $f(\cdot)$. The inputs obtained using the inverse model are then weighted to obtain the optimized control sequence, as in standard MPPI. The supporting video for the paper can be found at https://youtu.be/snhlZj3l5CE.

The third main contributions (**MC3**) are the following.

1. It proposes an output-sampling-based MPPI (o-MPPI) using inverse models, which improves the ability to select rollouts that satisfy typical output constraints. An advantage of output sampling is that it can leverage well-established trajectory planning algorithms in robotics to select output samples and then use inverse models (physics-based or data-based) to find the associated inputs. This is different from selecting inputs and then finding the associated outputs using forward models.

2. Comparative simulations and experiments of autonomous control of bots (TurtleBot3 Burger) in a dynamic autonomous driving around a track, are used to show that the proposed o-MPPI requires substantially (20-times) less number of rollouts and (4-times) smaller prediction horizon when compared with the standard MPPI to achieve a similar success rates.

1.5 Research timeline and thesis structure

The summary of the research timeline, the main problem in different scenarios, and contributions are shown in Fig. 1.2 below. The subsequent chapters begin with reviews of the major challenges in different **SC**s in Chapter 2. Chapter3 introduces **MC1** that handles the challenge of the trajectory tracking with unknown interaction dynamics in **SC1**, including the derivations of the proposed new input-weighted Gaussian process regression, the upper bounds on iteration gain design for MIMO system, and the comparative experimental results. Chapter 4 presents **MC2.1** that identifies the type of output data needed to remove hidden-state dependence, with neural-net-based modeling for a spring-mass-damper system that has hidden-state dependence. Chapter 5 describes **MC2.2** that extends the inverse operator in **MC2.1** to multi-input-multi-output nonlinear systems. Chapter 6 shows the output-based Model Predictive Path Integral control (o-MPPI) developed in **MC3**. At last, Chapter 7 discusses the potential directions in the future about the usage of the inverse operator modeling for multi-input-multi-output system in model predictive control (MPC).

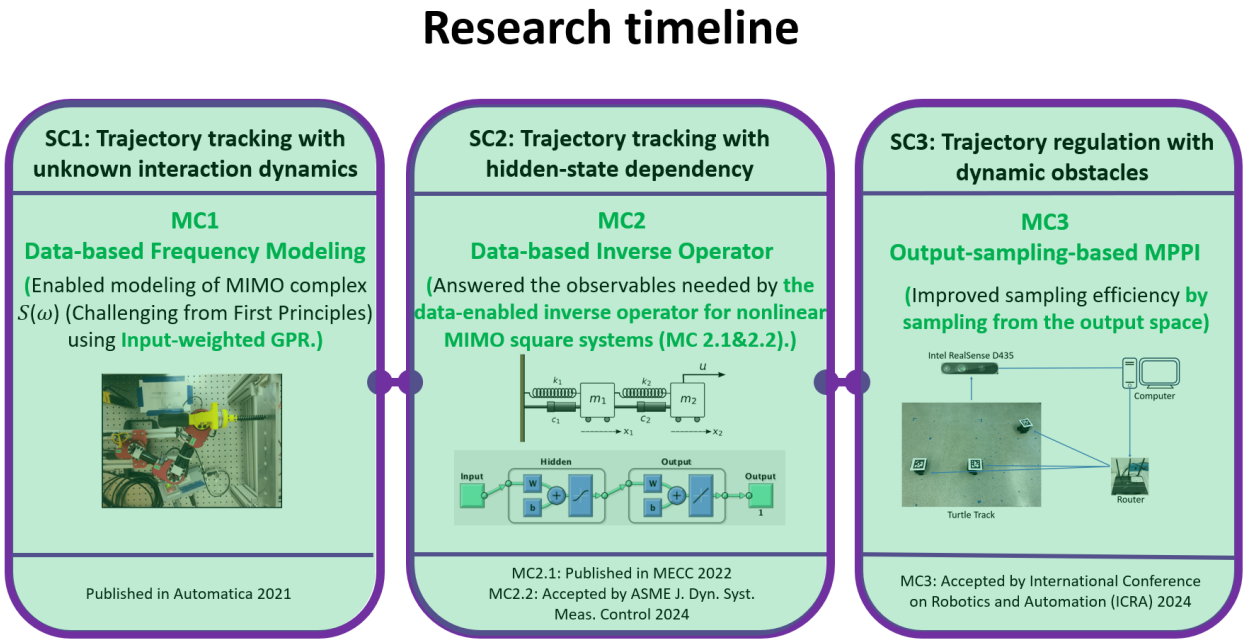

Figure 1.2: Research timeline, publications [1, 2, 3, 4].

Chapter 2

BACKGROUND

This chapter provides backgrounds and the main challenges in each Scenario (**SC1**, **SC2** and **SC3**) in this thesis. Sections are taken from the published & submitted articles and the submitted.

2.1 Data-based frequency modeling for trajectory tracking with unknown interaction dynamics (SC1)

Lightweight SEA robots, along with low-impedance control, can maneuver without causing damage in uncertain, confined spaces such as inside an aircraft wing during aircraft assembly. Nevertheless, substantial modeling uncertainties in SEA robots reduce the precision achieved by model-based approaches such as inversion-based feedforward. Iterative learning methods can enable precision control even in the presence of modeling uncertainties, especially when the task can be learned ahead of time or can be repeated [37]. Moreover, previously learned precision sub-tasks could be recombined to form new tasks [38]. When the model uncertainty is low, model inversion can be used to improve the performance of robots with elastic joints, e.g., [24] and can be applied to correct for nonminimum-phase flexural dynamics of lightweight robots [39]. Therefore, in **SC1**, a model-inversion-based ILC [40, 41, 42] is used to account for modeling uncertainties with the lightweight SEA robot. However, the ILC iterations could potentially diverge if the modeling uncertainty is large, e.g., [43]. This motivates approaches developed to reduce the model uncertainty to improve convergence [44], by using observed input-output data ($I_O(\omega), O_O(\omega)$ at frequency ω) of the form $I_O(\omega)/O_O(\omega)$ for single-input single-output (SISO) systems, rather than using the inverse of a known model. Even with the direct use of input-output data, the effective

model uncertainty $\Delta(\omega)$ can be large still if the signal-to-noise ratio is small, e.g., when the desired output $O_O(\omega)$ at some frequency ω is small [45]. An approach is to not iterate at those specific frequencies where the desired output is small, e.g., [44, 46]. Another approach is to inject additional input at such frequencies (where the desired output is small) to ensure persistence of excitation when estimating the model from data, which enables the learned model to be portable and applicable to track new trajectories, e.g., [47]. In either case, provided the uncertainty is sufficiently small, the ILC converges to the exact-tracking input for SISO systems, even in the presence of modeling uncertainties. Therefore, it is expected that the model-inversion-based ILC approach could improve the precision for the MIMO case, in the presence of small uncertainties, e.g., with the use of a linearized model of the SEA robot obtained at an operating point.

While convergence conditions have been well established for the SISO inversion-based ILC [41], extension to the MIMO inversion-based ILC remains challenging. For sufficiently-diagonally-dominant systems G, convergence can be established with the use of a diagonalized inverse model in the ILC as shown in [48]. Recent efforts [49, 50] have considered ILC with the full MIMO inverse [40]. For example, the ILC gain $\rho(\omega)$ could be optimized to minimize the tracking error with the constraint that the iterations converge, e.g., by using a Q-filter when computing the ILC input [50]. The Q-filter approach trades off between robustness and performance; if the anticipated uncertainty is small, then the tracking error is small, but not zero, when the Q-filter is nontrivial, i.e., $Q(\omega) \neq 1$. In contrast, **MC1** develops conditions on the ILC gain, using the Geršgorin theorem as in [49], for ensuring convergence to the desired output, even under some modeling uncertainty.

A challenge with inversion-based ILC is that the actual system dynamics $S(\omega)$ and the uncertainty $\Delta(\omega)$ are unknown. Therefore, it is difficult to ensure that the convergence condition on the model uncertainty is met. One approach is to use a relatively large number of repetitive experiments for identifying both the model and the uncertainty prior to applying the ILC [41, 50]. An alternative approach is to use kernel-based regression approaches to estimate the model and its uncertainty for the SISO case [47] from data obtained from a

relatively-sparse number (one or two) of the ILC iterations. This use of ILC data allows the data-based model to capture the local, linear model near the work area that includes potential contact-dependent effects, which can be challenging to model from first principles. Note that kernel-based Gaussian process regression (GPR, popularly also referred to as machine learning) is well suited to estimation of the general functions (such as the frequency response function) as well as their uncertainty from noisy experimental data [51]. Though complex kernels have been used in the past for GPR [52], they cannot be used to estimate models for multi-input systems. This motivates the development, in **MC1**, of an input-weighted complex kernel to estimate the complex-valued system MIMO model $\hat{S}(\omega)$ as well as its uncertainty $\Delta(\omega)$. Since it is designed in the Fourier domain, it captures the noncausality needed for many system inversion approaches, e.g., as investigated in [53], and can be used with SISO frequency-domain kernels which ensure BIBO stability of the resulting models, e.g., as in [54]. Such kernel-based methods tend to be robust than typical model-identification methods in infinite-dimensional spaces, e.g., as discussed in [55].

MC1 extends preliminary results in [49] by (i) proving the conditions for convergence, (ii) using augmented inputs in the ILC to ensure sufficiently large signal-to-noise ratio in the frequency range of interest, and (iii) applying and evaluating the approach for precision control of a lightweight SEA robot. Comparative experimental results, with and without ILC, are presented that show around 90% improvement in the positioning precision with the use of the MIMO ILC (close to the repeatability limit of the robot) and a 10-times increase in the SEA robot's operating speed.

2.2 Data-based inverse operator for trajectory tracking with hidden-state dependence (SC2)

The advent of easy access to large amount of data has sparked interest in directly developing the relationships between input and output of dynamic systems. A challenge is that in addition to the applied input and the measured output, the dynamics can also depend on hidden states that are not directly measured.

Several approaches are available to address the lack of direct access to the hidden states. One approach is to represent the dynamics through Markov models with a predefined number of hidden states, and then minimize the model prediction error [56, 57, 58]. A difficulty is that the optimal selection of the number of hidden states can be computationally expensive, and there is no guarantee that the resulting models will achieve the desired precision. A second class of approaches to handle the lack of direct access to the hidden states is to model the system dynamics (flow) in a lifted observable space (with generalized functions of the observables) using Koopman operator theory [8, 59]. Recent techniques include sparse identification of nonlinear dynamical systems (SINDy) [9] and linearization Dynamic Mode Decomposition(DMD) [60]. Nevertheless, with a finite number of states, there is uncertainty about how to select a sufficient set of generalized observable functions to achieve a specified level of prediction precision. A third class of approaches is to use time history of the input and output data to find forward models, e.g., with (i) transfer function models in the frequency domain [47, 1]; (ii) autoregressive models with extra input (ARX) [61] as well as nonlinear ARX (NARX) [36, 62]; (iii) time-delayed information in the Koopman operator framework [63]; and fitting a relation between the time-delayed output data and the inverse input [53, 64, 65]. Again, determining the type of data needed to capture the input-output relationship (with high precision) when models are not available a-priori remains uncertain. When precision of the inverse is not sufficient, it can be improved using iterative techniques, with the inverse of the plant considered as the learning operator, [66, 67, 68, 69]. Nevertheless, increasing the precision of the inverse model can improve ILC convergence.

The goal in **SC2** is to identify the type of output data needed to develop inverse (output-to-input) operators, with a desired level of precision. Rather than the two step processes of first learning forward models and second using model-predictive control (MPC) to optimally select the control input, the proposed approach seeks to solve the inverse problem of directly finding the input for a given output, e.g., similar to [70, 71]. In particular, the relative degree of the system is used to identify the number of time derivatives that need to be added to input-output data to facilitate precision data-enabled learning of the inverse operator.

Previous works on inversion of system dynamics, using known models of the system, have shown that the impact of neglecting the boundary conditions of the internal states can be made arbitrarily small [72, 73] by choosing a sufficiently large time history of the desired output and its derivatives. Additionally, errors in the output tracking due to model uncertainty can be corrected using iterative methods (in the frequency domain) as shown in [1, 47]. This frequency-domain approach maps the desired output over the entire time (both past and future) into the precision tracking input – also over the entire time. The results at the end of such iterations, in the time domain lead to an output-tracking input. The resulting input-output pair, at the end of the iterations, can be used as training data to train a neural-net model for learning the inverse operator [65] although the iterations needed to find the input-output pairs add to the time and effort needed for training. Alternatively, Gaussian-process-based inverse operators can be developed between the past and preview of the output and the output tracking input [74]. These prior works indicate that the inverse operator can be learned from data, without the need to explicitly capture the hidden-state dynamics.

This motivates the proposed data-enabled algorithm in **MC2.1** & **MC2.2** to learn the inverse operator directly from input-output data (without the need to explicitly capture the hidden state dynamics) by using time-delayed observations of the output, along with the output's time derivatives.

2.3 Inverse-model-based model predictive control for trajectory regulation with moving obstacles (SC3)

Sampling-based approaches such as model predictive path integral control (MPPI) [75] have become popular methods to solve optimization problems due to fast computations possible with graphic processing units and parallelized computing. In such methods, models are used to predict the cost for a series of input rollouts, and the final input selection is a cost-weighted average of the rollouts. An advantage of the sample-based approach is that it is derivative-free, does not require approximation of the system dynamics and cost functions, and allows

for non-differentiable cost functions [76].

Successful use of the MPPI algorithm requires proper selections of the mean and the covariance of the input samples. If the selected mean leads to a rollout that enters inside an infeasible region (this could be caused by the poor initialization, system disturbance, or sudden environment changes), then most of the sampled rollouts can result in failure [77]. Although a larger covariance can be used to alleviate failures by increasing exploration [78], it also often requires a larger number of samples and increased computation load, and can lead to undesirable input chattering [75, 79]. Therefore, there is substantial ongoing interest in methods to appropriately select the input samples to improve MPPI performance. However, in general, it is challenging to appropriately select a desirable input mean to meet constraints in the output space, especially in dynamic environments. In general, even when the reference input can be selected as desired outputs to a closed-loop system that satisfies constraints, it does not ensure that the output achieves precision tracking of the desired trajectories.

2.3.1 MPPI with adjusted trajectory distribution

Several efforts are made to improve the robustness and sample efficiency of standard MPPI by appropriately adjusting the trajectory distribution using different sampling strategies [77, 80, 81, 82]. For example, warm-starts are used [75] to improve MPPI-variants. In [80], covariances are adapted to accommodate different actions for better exploration. In [81], the samples are drawn from a normal and log-normal (NLN) distribution instead of the Gaussian distribution, which can yield improved performance in cluttered environments. In [82], the diagonal covariance matrix Σ_ϵ is changed into a nondiagonal covariance matrix and updated through the adaptive importance sampling procedure. Recently, constraints were added to the terminal state of the prediction horizon in [77] to adjust the input distribution. In all these works sampling is in the input space followed by the use of forward models to determine the outputs. In contrast, **MC3** proposes sampling in the output space (since constraints might be more easily satisfied in the output space) and then uses an inverse model to find the corresponding inputs. Once the input-output pairs are found, the proposed output-

sampling-based MPPI (o-MPPI) in **MC3** is similar to standard MPPI and can use the same cost functions and weighting strategies.

2.3.2 MPPI with output-space-informed mean

The output space has been leveraged in the past to improve the input-sampling efficiency of MPPI. Since a better mean for the input distribution can also help improve MPPI, an appropriately selected output can be used to inform the mean selection used in standard MPPI. For example, in [78], the fast rapid-exploring-tree (RRT^*) algorithm is used to provide guidance about the mean value of the input for improving sampling efficiency in the input space. In [83], trained Conditional Variational Autoencoders that take into account the contextual information are used to better inform the mean values by taking into the control uncertainties. Thus, the above methods utilize output-space information to guide the selection of the mean value for the input sampling. The proposed o-MPPI approach in **MC3** extends this idea and fully samples in the output space and then uses inverse models to find the corresponding input. It is noted that sampling of the output space has been used in the path-planning community to smooth and potentially optimize feasible trajectories, e.g., [84, 85]. Here, these planned trajectories are inputs to the closed-loop system, and forward models (of the closed-loop system) are used to predict the system response and for optimizing the selected input (final output trajectory) to the closed-loop system. The proposed o-MPPI in **MC3** can be used with any trajectory planning algorithm such as the fast rapid-exploring-tree (RRT^*) [78] - the main difference is that inverse models are used in o-MPPI to find inputs that track the selected output trajectories.

2.3.3 Inverse models

The proposed o-MPPI in **MC3** leverages the strong history in inverse dynamics to find maps from outputs to inputs. For example, physics-based models can be used to analytically find the inverse input [86] in robotics applications. When the physics-based models are not sufficiently precise, the Gaussian process can be used to learn the discrepancy [87]. The

entire inverse model can be approximated with deep learning models [88] or Gaussian process models [89]. Additionally, data-enabled models can be used to learn the inverse output-to-input map from input-output data [2]. Thus, the proposed o-MPPI in **MC3** can use a wide range of inverse modeling tools to find the inputs associated with the sampled output trajectories.

Chapter 3

MIMO ILC FOR PRECISION SEA ROBOTS USING INPUT-WEIGHTED COMPLEX-KERNEL REGRESSION (MC1)

This chapter proposes an approach to MIMO ILC for Precision SEA Robots using Input-Weighted Complex-Kernel Regression (**MC1**) and is based on a work published in Automatica [1]. This work improves the positioning precision of lightweight robots with series elastic actuators (SEAs). Lightweight SEA robots, along with low-impedance control, can maneuver without causing damage in uncertain, confined spaces such as inside an aircraft wing during aircraft assembly. Nevertheless, substantial modeling uncertainties in SEA robots reduce the precision achieved by model-based approaches such as inversion-based feedforward. Therefore, this chapter improves the precision of SEA robots around specified operating points, through a multi-input multi-output (MIMO), iterative learning control (ILC) approach. The main contributions of this chapter are to (i) introduce an input-weighted complex kernel to estimate local MIMO models using complex Gaussian process regression (c-GPR); (ii) develop Geršgorin-theorem-based conditions on the iteration gains for ensuring ILC convergence to precision within noise-related limits, even with errors in the estimated model; and (iii) demonstrate precision positioning with an experimental SEA robot. Comparative experimental results, with and without ILC, show around 90% improvement in the positioning precision (close to the repeatability limit of the robot) and a 10-times increase in the SEA robot's operating speed with the use of the MIMO ILC.

3.1 Problem formulation

3.1.1 Inversion-based iterative-learning control (ILC)

Given a linear time-invariant (LTI) system with transfer function (or frequency response function) $S(\omega)$

$$O(\omega) = S(\omega)I(\omega), \tag{3.1}$$

with output $O(\omega)$ and input $I(\omega)$ represented in the Fourier domain at frequency ω, the goal is to find an input $I_d(\omega)$ that yields exact tracking of the desired output $O_d(\omega)$, i.e.,

$$O_d(\omega) = S(\omega)I_d(\omega), \tag{3.2}$$

where the system $S(\omega)$ is stable and the number of outputs m is not more than the number of inputs n, i.e., $m \leq n$.

Assumption 1. *The LTI system $S(\omega) \in \mathbb{C}^{m\times n}$ has full row rank on the imaginary axis.*

Remark 1. *The rank condition implies that there are no transmission zeros on the imaginary axis, which guarantees the existence of a finite input $I_d(\omega)$ that achieves exact tracking of the desired output $O_d(\omega)$ and ensures robustness of the inverse [90]. However, conditions have been studied recently on when such zeros are acceptable for inversion [91].*

The iterative approach generates a new input I_{k+1} for use at iteration step $k+1$ based on the tracking error

$$E_k(\omega) = O_d(\omega) - O_k(\omega) \tag{3.3}$$

from the prior iteration step k, which is similar to prior use for the square system case, e.g., in [41, 40, 49]

$$I_{k+1}(\omega) = I_k(\omega) + \hat{S}^{\dagger}(\omega)\rho(\omega)(O_d(\omega) - O_k(\omega)) \tag{3.4}$$

where $\hat{S}^{\dagger}(\omega) = \mathbf{W}^{-1}(\omega)\hat{S}^{*}(\omega)(\hat{S}(\omega)\mathbf{W}^{-1}(\omega)\hat{S}^{*}(\omega))^{-1}$ is the pseudo-inverse of the estimated model $\hat{S}(\omega)$ of the unknown system $S(\omega)$ with $\hat{S}(\omega)\hat{S}^{\dagger}(\omega) = \mathbf{1}$ and the identity matrix is denoted by $\mathbf{1}$. The superscript $*$ denotes the complex conjugate, $\mathbf{W}(\omega) \in \mathbb{R}^{n\times n}$ is an

invertible diagonal matrix, with $\mathbf{W}_{i,i}(\omega) = w_i(\omega) > 0$, and $\rho(\omega) \in \mathbb{R}^{m\times m}$ is a diagonal matrix with nonnegative elements representing the iteration gain.

Remark 2. *For square systems with the same number of inputs n and outputs m, i.e., $n = m$, the pseudo-inverse $\hat{S}^{\dagger}(\omega)$ becomes the exact inverse $\hat{S}^{-1}(\omega)$. For the actuator redundant case, i.e., $n > m$, $I(\omega) = \hat{S}^{\dagger}(\omega)O(\omega)$ solves the input-output equation $O(\omega) = \hat{S}(\omega)I(\omega)$ while minimizing the frequency-weighted input cost $\|I(\omega)\|^2_{\boldsymbol{W}(\omega)} = I^*(\omega)\,\boldsymbol{W}(\omega)I(\omega)$.*

Multiplying $S(\omega)$ on both sides of Eq (3.4) results in

$$O_{k+1}(\omega) = O_k(\omega) + S(\omega)\hat{S}^{\dagger}(\omega)\rho(\omega)(O_d(\omega) - O_k(\omega)).$$

Subtracting the desired output $O_d(\omega)$ from both sides yields a relation between error defined in Eq. (3.3) at consecutive iteration steps, as

$$E_{k+1}(\omega) = [\mathbf{1} - \rho(\omega) - \Delta(\omega)\rho(\omega)]E_k(\omega) = G(\omega)E_k(\omega) \tag{3.5}$$

where the unknown model uncertainty $\Delta(\omega) \in \mathbb{C}^{m\times m}$

$$\Delta(\omega) = S(\omega)\hat{S}^{\dagger}(\omega) - \mathbf{1}. \tag{3.6}$$

In the following, the initial iteration input I_0 is selected as the desired output, $I_0(\omega) = O_d(\omega)$, which is assumed to be sufficiently smooth and bounded over the frequency range.

Remark 3. *The model uncertainty $\Delta(\omega)$ in Eq. (3.6) quantifies the error in the estimated model $\hat{S}$, or equivalently the error in computing the pseudo-inverse $\hat{S}^{\dagger}$.*

Remark 4. *From Assumption 1, the estimated model $\hat{S}(\omega)$ has full row rank on the imaginary axis for sufficiently small model estimation error $\delta(\omega) = S(\omega) - \hat{S}(\omega)$.*

3.1.2 Problem statement

The iterations result in a contraction if the model uncertainty $\Delta(\omega)$ is sufficiently small. From Eq. (3.5), the tracking error $E_{k+1} = [G(\omega)]^k E_1(\omega)$, and therefore, with increasing iterations,

the tracking error $E(\omega)$ tends to zero in the 2-norm, if and only if $\lim_{k\to\infty}[G(\omega)]^k = \mathbf{0}$, which in turn occurs if and only if the spectral radius $\sigma_G(\omega)$ of the contraction gain $G(\omega)$ is less than one as in Eq. (3.9), e.g., see Theorem 5.6.12 in [92]. Therefore, as shown in the previous works e.g., [41, 40, 49], the MIMO ILC in Eq. (3.4) converges at frequency ω for any general output $O_d(\omega)$, i.e., the tracking error tends to zero

$$\lim_{k\to\infty} \|E_k(\omega)\|_2 = 0 \tag{3.7}$$

as iterations increase, $k \to \infty$, if and only if the spectral radius $\sigma_G(\omega)$ (maximum magnitude of the eigenvalues $\lambda_i(G(\omega))$) of the the contraction gain

$$G(\omega) = [\mathbf{1} - \rho(\omega)] - \Delta(\omega)\rho(\omega) \tag{3.8}$$

is less than one, i.e.,

$$\sigma_G(\omega) \ = \max_i |\lambda_i(G(\omega))| < 1. \tag{3.9}$$

Remark 5. *Convergence of the tracking error $E_k(\omega)$ to zero in the two norm also results in convergence in any other norm, since all norms are equivalent in finite-dimensional linear vector spaces, and for any norm $\|\cdot\|_p$ there exists a constant γ_p such that [92] $\|E_k(\omega)\|_p \leq \gamma_p \|E_k(\omega)\|_2$.*

The research issue is to develop conditions on the iteration gain $\rho(\omega)$ to ensure convergence of the ILC under given bounds $\overline{\Delta}(\omega)$ on the model uncertainty $\Delta(\omega)$. This can be split into a model and uncertainty estimation problem and the problem of selecting the iteration gain $\rho(\omega)$ to ensure convergence.

1. Given input-output data $(I_O(\omega), O_O(\omega))$, estimate a model $\hat{S}(\omega)$ of the system $S(\omega)$ and bound $\overline{\Delta}(\omega)$ on the model uncertainty $\Delta(\omega)$ in Eq. (3.6).

2. Given a bound $\overline{\Delta}(\omega)$ on the model uncertainty $\Delta(\omega)$, develop conditions on the iteration gain $\rho(\omega)$ to satisfy the ILC convergence condition in Eq. (3.9).

3.2 Solution: ILC design for convergence

3.2.1 Model and uncertainty estimation

The goal is to estimate the model terms $\hat{S}_{j,l}(\omega)$ from observation $O_{j,O}$ given by

$$O_{j,O}(\omega, I_O(\omega)) = O_j(\omega, I_O(\omega)) + \epsilon_j = \sum_{l=1}^{n} S^P_{j,l}(\omega) I_{l,O}(\omega) + \epsilon_j, \tag{3.10}$$

where ϵ_j is the measurement noise and system components $S_{j,l}(\omega)$ are replaced by Gaussian processes $S^P_{j,l}(\omega)$ due to noise as in previous works, e.g., [54, 55].

If only one input, say $I_l(\omega)$, is nonzero, then the system component $S_{j,l}(\omega) = O_j(\omega)/I_l(\omega)$ is a complex function that can be estimated using Gaussian process regression (GPR) with previously-developed complex kernels, e.g., the one in [54, 52, 93]. However, such an approach requires n separate experiments with only one nonzero input in each experiment, which might not be feasible in general. Such approaches are not directly applicable to the multi-input case.

Towards model identification for the multiple-input case, given any set of complex kernels $\hat{k}_l(\cdot,\cdot)$ for $1 \le l \le n$, the following input-weighted kernel $\hat{k}'(\cdot,\cdot)$ is proposed for the composite Gaussian process O_j in Eq. (3.10),

$$\hat{k}'([\omega_r, \{I_{l,O}(\omega_r)\}_{l=1}^n], [\omega_s, \{I_{l,O}(\omega_s)\}_{l=1}^n]) = \sum_{l=1}^{n} I_{l,O}(\omega_r) \hat{k}_l(\omega_r, \omega_s) I^*_{l,O}(\omega_s), \tag{3.11}$$

where the subscripts r, s refer to different frequencies belonging to the observed frequency set

$$\Omega = [\omega_1, \ldots, \omega_q] \in \mathbb{R}^q \tag{3.12}$$

with nonzero accompanying input $\{I_{l,O}(\Omega) \in \mathbb{C}^q\}_{l=1}^n$. A motivation for the proposed kernel is that for the single-input case ($n = 1$), using the input-weighted kernel $\hat{k}'$ to estimate the component $S^P_{j,1}(\omega)$ is equivalent to obtaining an estimate with the unweighted kernel $\hat{k}_1$ based on observed ratios $O_{1,O}(\omega)/I_{1,O}(\omega)$, as shown in [49]. Moreover, $\hat{k}'$ is a valid kernel (Hermetian positive semi-definite) provided the underlying kernels $\hat{k}_l$ associated with each term are valid kernels, as shown in the lemma below under the following assumption.

Assumption 2. *All the complex Gaussian Processes $S^P_{j,l}$ are independent, and have zero-mean prior. Moreover they are assumed to (i) be proper as in [94, 52], resulting in kernel functions $\hat{k}_l$ that have the same variance for the real and imaginary parts at each frequency $\mathbb{V}[\mathrm{Re}(S^P_{j,l}(\omega))] = \mathbb{V}[\mathrm{Im}(S^P_{j,l}(\omega))]$; and (ii) have independent real and imaginary components resulting in $\mathbb{V}[S^P_{j,l}(\omega)] = \mathbb{V}[\mathrm{Re}(S^P_{j,l}(\omega))] + \mathbb{V}[\mathrm{Im}(S^P_{j,l}(\omega))]$.*

Lemma 1. *With nonzero input I_O at each frequency in the observed frequency set Ω, the input-weighted kernel $\hat{k}'$ in Eq. (3.11) is Hermitian positive semi-definite if the kernels $\hat{k}_l$ $1 \le l \le n$ are Hermitian positive semi-definite. Moreover, the self covariance matrix K' in Eq. (3.13), associated with $\hat{k}'$, is positive definite if covariance matrices $K_l(\Omega, \Omega)$ are positive definite for $1 \le l \le n$.*

Proof. The Hermetian property of the input-weighted kernel $\hat{k}'$ follows from Eq. (3.11) since

$$\begin{aligned}
\hat{k}'([\omega_r, \{I_{l,O}(\omega_r)\}_{l=1}^n], [\omega_s, \{I_{l,O}(\omega_s)\}_{l=1}^n]) &= \sum_{l=1}^n I_{l,O}(\omega_r)\hat{k}_l(\omega_r, \omega_s)I^*_{l,O}(\omega_s) \\
&= \sum_{l=1}^n \left\{I_{l,O}(\omega_s)\hat{k}_l(\omega_s, \omega_r)I^*_{l,O}(\omega_r)\right\}^* \\
&= \left\{\hat{k}'([\omega_s, \{I_{l,O}(\omega_s)\}_{l=1}^n], [\omega_r, \{I_{l,O}(\omega_r)\}_{l=1}^n])\right\}^*
\end{aligned}$$

and positive semi-definiteness follows since each term in the summation in Eq. (3.11) is non-negative. The covariance K' can be written as, due to independence of terms from Assumption 2,

$$K'([\Omega, \{I_{l,O}(\Omega)\}_{l=1}^n], [\Omega, \{I_{l,O}(\Omega)\}_{l=1}^n]) = \sum_{l=1}^n \mathrm{diag}(I_{l,O}(\Omega))K_l(\Omega, \Omega)\mathrm{diag}(I^*_{l,O}(\Omega)). \quad (3.13)$$

The r^{th} row and s^{th} column element of the covariance matrix $K' \in \mathbb{C}^{q\times q}$ is the evaluation of the input-weighted kernel $\hat{k}'$ on frequency-input pair $[\omega_r, \{I_{l,O}(\omega_r)\}_{l=1}^n]$ and $[\omega_s, \{I_{l,O}(\omega_s)\}_{l=1}^n]$, which is computed as the input-weighted summation of the corresponding r^{th} row and s^{th} column element of each covariance matrix $K_l \in \mathbb{C}^{q\times q}$ computed as $\hat{k}_l(\omega_r, \omega_s)$. $\mathrm{diag}(I_{l,O}(\Omega))$ creates a q-by-q diagonal matrix with the elements of vector $I_{l,O}(\Omega)$ on the main diagonal. For any vector $v \in \mathbb{C}^q$,

$$
\begin{aligned}
v^* K'([\Omega, I_O(\Omega)], [\Omega, I_O(\Omega)])v &= v^* \sum_{l=1}^{n} \mathrm{diag}(I_{l,O}(\Omega)) K_l(\Omega,\Omega) \mathrm{diag}(I_{l,O}^*(\Omega)) v \\
&= \sum_{l=1}^{n} (\mathrm{diag}(I_{l,O}^*(\Omega))v)^* K_l(\Omega,\Omega) \mathrm{diag}(I_{l,O}^*(\Omega)) v \\
&= \sum_{l=1}^{n} u_l^* K_l(\Omega,\Omega) u_l,
\end{aligned}
$$

where $u_l = \mathrm{diag}(I_{l,O}^*(\Omega))v$ is nonzero if v is nonzero. Thus, $K'([\Omega, I_O(\Omega)], [\Omega, I_O(\Omega)])$ is positive (semi-) definite if for all l, $K_l(\Omega,\Omega)$ is positive (semi-) definite. □

Lemma 2 (Multi-input system identification)**.** *The estimate of each term $S_{j,l}^P(\omega)$ in Eq.* (3.10) *is given by*

$$
\hat{S}_{j,l}(\omega) = \mathbb{E}[O_j(\omega, e_l)] = \mathbb{E}[S_{j,l}^P(\omega)] = K_T'(K' + \sigma_{j,\epsilon}^2 \boldsymbol{1})^{-1} O_{j,O}([\Omega, \{I_{l,O}(\Omega)\}_{l=1}^n]), \tag{3.14}
$$

with estimated variance $\mathbb{V}[S_{j,l}^P(\omega)] = \mathbb{V}[O_j(\omega, e_l)]$ given by

$$
\mathbb{V}[S_{j,l}^P(\omega)] = K_0' - K_T'(K' + \sigma_{j,\epsilon}^2 \boldsymbol{1})^{-1} (K_T')^*, \tag{3.15}
$$

where $\sigma_{j,\epsilon}$ is the noise variance at j^{th} output, e_l is an n-by-1 vector with one in the l^{th} component and zero elsewhere, and

$$
\begin{aligned}
K' &= K'([\Omega, \{I_{l,O}(\Omega)\}_{l=1}^n], [\Omega, \{I_{l,O}(\Omega)\}_{l=1}^n]), \\
K_T' &= K'([\omega, e_l], [\Omega, \{I_{l,O}(\Omega)\}_{l=1}^n]), \\
K_0' &= K'([\omega, e_l], [\omega, e_l]).
\end{aligned} \tag{3.16}
$$

Proof. Since $S_{j,l}^P(\omega) = O_j(\omega, e_l)$ from Eq. (3.10), its estimate and variance follows from standard GPR methods, e.g., [52]. □

Remark 6. *Bounds $\overline{\delta}_{j,l}(\omega)$ on the model estimation error $\delta_{j,l}(\omega) = S_{j,l}(\omega) - \hat{S}_{j,l}(\omega)$, between the estimated model $\hat{S}(\omega)$ and the unknown system $S(\omega)$, can be obtained in terms of the estimated variance $\mathbb{V}[S_{j,l}^P(\omega)]$ in Eq.* (3.15) *with $\mathbb{V}[S_{j,l}^P(\omega)] = 2\mathbb{V}[\mathrm{Re}(S_{j,l}^P(\omega))]$ from Assumption 2, as*

$$
\overline{\delta}_{j,l}(\omega) = \gamma_\delta \sqrt{\mathbb{V}[\mathrm{Re}(S_{j,l}^P(\omega))]}, \tag{3.17}
$$

where the constant γ_δ can be larger for specifying a bound $\overline{\delta}_{j,l}(\omega)$ with a higher confidence level.

Lemma 3. *Each component $\Delta_{j,l}(\omega)$ of the model uncertainty $\Delta(\omega)$ defined in Eq. (3.6) is bounded as*

$$|\Delta_{j,l}(\omega)| < \sum_{k=1}^{n} \left|\hat{S}^{\dagger}_{k,l}(\omega)\right| \overline{\delta}_{j,k}(\omega). \tag{3.18}$$

Proof. This follows by replacing $S(\omega)$ in Eq. (3.6) by $S(\omega) - \hat{S}_{j,l}(\omega) + \hat{S}_{j,l}(\omega)$. □

3.2.2 Convergence conditions for MIMO ILC

Lemma 4 (MIMO ILC convergence conditions)**.** *Let each component $\Delta_{j,l}(\omega)$ of the uncertainty $\Delta(\omega)$ in Eq. (3.6) have the form*

$$\Delta_{j,l}(\omega) = M_{j,l}(\omega) e^{\mathbf{i}\Phi_{j,l}(\omega)} = A_{j,l}(\omega) + \mathbf{i} B_{j,l}(\omega), \tag{3.19}$$

with magnitude $M_{j,l}(\omega)$ and phase $\Phi_{j,l}(\omega)$ where $\mathbf{i} = \sqrt{-1}$. Also let the iteration gain $\rho(\omega)$ in Eq. (3.4) be diagonal, i.e., $\rho(\omega) = diag(\rho_1(\omega), \ldots, \rho_m(\omega))$. Then, the MIMO ILC in Eq. (3.4) converges at frequency ω if, for all $1 \le i \le m$,

$$R_i(\omega) < 1 + A_{i,i}(\omega), \tag{3.20}$$

$$0 < \rho_i(\omega) < 2 \frac{1 + A_{i,i}(\omega) - R_i(\omega)}{1 + 2A_{i,i}(\omega) + M^2_{i,i}(\omega) - R^2_i(\omega)}, \tag{3.21}$$

$$0 < 1 - \rho_i(\omega) R_i(\omega), \tag{3.22}$$

$$\text{where} \qquad R_i(\omega) = \sum_{j \neq i} M_{j,i}(\omega). \tag{3.23}$$

Proof. The conditions of this lemma are used to show that the eigenvalues of the contraction gain $G(\omega)$ have magnitude less than one, and therefore, the MIMO ILC convergence condition in Eq. (3.9) is met. By the Geršgorin theorem [92], all the eigenvalues of the contraction gain $G(\omega)$ are in the union of Geršgorin discs centered at $C_i(\omega) = G_{i,i}(\omega)$ with radius $R_i(\omega) = \sum_{j \neq i} |G_{j,i}(\omega)|$. Then, the eigenvalues of the contraction gain $G(\omega)$ are less

than one if all the Geršgorin discs are bounded by the unit circle centered at the origin, i.e.,

$$|C_i(\omega)| + R_i(\omega) < 1, \tag{3.24}$$

which can be rewritten using Eq. (3.8) as

$$|1 - \rho_i(\omega) - \rho_i(\omega)\Delta_{i,i}(\omega)| < 1 - \sum_{j \neq i} |\rho_i(\omega)\Delta_{j,i}(\omega)|. \tag{3.25}$$

Since the left hand side (LHS) of Eq. (3.25) is nonnegative, and needs to be strictly less than the right hand side (RHS), the RHS is required to be positive, which is satisfied due to the condition in Eq. (3.22) and $\rho_i(\omega)$ being positive from Eq. (3.21). Since both sides of Eq. (3.25) are nonnegative, squaring them and using Eq. (3.19) results in

$$(1 - \rho_i(\omega) - \rho_i(\omega)A_{i,i}(\omega))^2 + (\rho_i(\omega)B_{i,i}(\omega))^2 < (1 - \rho_i(\omega)R_i(\omega))^2.$$

Expanding the squares, and rearranging, yields

$$\rho_i^2(\omega)\left[1 + 2A_{i,i}(\omega) + M_{i,i}^2(\omega) - R_i^2(\omega)\right] < 2\rho_i(\omega)\left[1 + A_{i,i}(\omega) - R_i(\omega)\right]. \tag{3.26}$$

Since the radius $R_i(\omega)$ is non-negative, squaring the condition in Eq. (3.20) and using $A_{i,i}^2(\omega) \leq M_{i,i}^2(\omega)$, results in

$$R_i^2(\omega) < 1 + 2A_{i,i}(\omega) + M_{i,i}^2(\omega). \tag{3.27}$$

Since the iteration gain is positive $\rho_i(\omega) > 0$ from LHS of Condition (3.21) and from Eq. (3.27), $\rho_i(\omega)[1 + 2A_{i,i}(\omega) + M_{i,i}^2(\omega) - R_i^2(\omega)]$ is positive and can be divided from both sides of Eq. (3.26) to obtain

$$\rho_i(\omega) < \frac{2\left[1 + A_{i,i}(\omega) - R_i(\omega)\right]}{1 + 2A_{i,i}(\omega) + M_{i,i}^2(\omega) - R_i^2(\omega)} = P(\omega), \tag{3.28}$$

which is satisfied due to the condition in Eq. (3.21). Thus, the conditions of the lemma ensure that the contraction gain $G(\omega)$ has eigenvalues with magnitude less than one based on Eq. (3.24). □

Remark 7. *The conditions on the iteration gain $\rho(\omega)$ in Lemma 4 are only sufficient and not necessary, and therefore, are conservative and may not result in the fastest possible convergence. Nevertheless, they ensure convergence to exact tracking.*

Lemma 5 (Bounded-uncertainty convergence)**.** *When each component $\Delta_{j,l}(\omega)$ in model uncertainty is bounded in magnitude as in Eq.* (3.18)*, i.e.,*

$$M_{j,l}(\omega) < \overline{\Delta}_{j,l}(\omega), \tag{3.29}$$

the MIMO ILC convergence conditions in Lemma 4 are satisfied if

$$0 < \rho_i(\omega) < \overline{\rho}_i(\omega) \tag{3.30}$$

$$\rho_i(\omega)\overline{\Delta}_{R,i}(\omega) < 1, \tag{3.31}$$

$$\overline{\Delta}_{R,i}(\omega) < \ 1 - \overline{\Delta}_{i,i}(\omega), \tag{3.32}$$

where $\overline{\Delta}_{R,i}(\omega) = \sum_{j\neq i} \overline{\Delta}_{j,i}(\omega)$ and

$$\overline{\rho}_i(\omega) \ = \min_{p=\pm 1} \frac{2\left[1 + p\overline{\Delta}_{i,i}(\omega) - \overline{\Delta}_{R,i}(\omega)\right]}{1 + 2p\overline{\Delta}_{i,i}(\omega) + \overline{\Delta}^2_{i,i}(\omega) - \overline{\Delta}^2_{R,i}(\omega)}. \tag{3.33}$$

Proof. The radius $R_i(\omega) < \overline{\Delta}_{R,i}(\omega)$ from Eq. (3.29). Therefore (i) Eq. (3.31) implies $R_i(\omega)\rho_i(\omega) < 1$ and that the condition in Eq. (3.22) is satisfied since $\rho_i(\omega) > 0$, and (ii) Eq. (3.32) implies $R_i(\omega) < 1 - \overline{\Delta}_{i,i}(\omega) < 1 + A_{i,i}(\omega)$ and that the condition in Eq. (3.20) is met since $\overline{\Delta}_{i,i}(\omega) > M_{i,i} \ \geq |A_{i,i}(\omega)|$. Moreover, the upper bound on the iteration gain in Eq. (3.21), denoted by $P(\omega)$ as in Eq. (3.28), is shown below to be a monotonic function of each variable $R_i(\omega) \in [0, \overline{\Delta}_{R,i}(\omega)]$, $M_{i,i}(\omega) \in [0, \overline{\Delta}_{i,i}(\omega)]$ and $A_{i,i}(\omega) \in [-\overline{\Delta}_{i,i}(\omega), \overline{\Delta}_{i,i}(\omega)]$. The upper bound $P(\omega)$ decreases with increasing $M_{i,i}$ and is therefore minimized at $M_{i,i} = \overline{\Delta}_{i,i}(\omega)$. With $X = 1 + A_{i,i}(\omega)$ and $Y = 1 + 2A_{i,i}(\omega) + M^2_{i,i}$,

$$\frac{\partial P}{\partial R_i(\omega)} = 2\frac{-(Y - R_i^2(\omega)) - (X - R_i)(-2R_i(\omega))}{(Y - R_i^2(\omega))^2} = 2\frac{-(R_i(\omega) - X)^2 - (Y - X^2)}{(Y - R_i^2(\omega))^2} \leq 0$$

since $Y - X^2 \geq 0$ as $M_{i,i}(\omega) \geq A^2_{i,i}(\omega)$. Therefore, for independent of $A_{i,i}(\omega)$ and $M_{i,i}(\omega)$, $P(\omega)$ is minimized when $R_i(\omega) = \overline{\Delta}_{R,i}(\omega)$. Finally, with $X' = 1 - \overline{\Delta}_{R,i}(\omega)$ and $Y' = 1 -$

$\overline{\Delta}^2_{R,i}(\omega) + \overline{\Delta}^2_{i,i}(\omega)$,

$$\frac{\partial P}{\partial A_{i,i}(\omega)} = 2\frac{(Y' + 2A_{i,i}(\omega)) - 2(X' + A_{i,i}(\omega))}{(Y' + 2A_{i,i}(\omega))^2} = 2\frac{Y' - 2X'}{(Y' + 2A_{i,i}(\omega))^2},$$

which does not change sign. Consequently, the smallest $P(\omega)$ occurs at either $A_{i,i}(\omega) = \pm\overline{\Delta}_{i,i}(\omega)$ and thus, satisfying Eq. (3.30) ensures that Eq. (3.21) is satisfied. □

Remark 8. *The MIMO ILC converges from Lemma 5 if (i) the uncertainty bounds are sufficiently small to satisfy Eq.* (3.32) *and (ii) the nonzero iteration gain* $\rho_i(\omega)$*is chosen to be sufficiently small to satisfy Eqs.* (3.30) *and* (3.31) *for all* $1 \le i \le m$.

Remark 9. *If bounds on the modeling error are estimated from data as in Remark 6 with some confidence level, then satisfying the conservative conditions of Lemma 5 ensures convergence to exact tracking with at least the same level of confidence.*

Remark 10. *Noise in measurements can limit the achievable convergence. However, the tracking error with ILC tends to be small if the noise is small [41].*

3.2.3 ILC algorithm

The ILC design and procedure are summarized in Algorithm 1.

Algorithm 1 MIMO ILC through Machine Learning

1. Initialization: Set $I_0(\omega) = O_d(\omega), k = 0$. Select error threshold ϵ and the maximum iteration steps k_{max};

2. Initial input: k = 0.

Apply input $I_0(\omega)$ to the system and measure output $O_0(\omega)$;

3. Perturbed input: k = 1.

Apply perturbed input $I_1(\omega) = I_0(\omega) + I_p(\omega)$ to the system and measure output $O_1(\omega)$ where the perturbation $I_p(\omega)$ adds frequency content to improve model estimation [47];

4. Model estimation:

Compute output perturbation $O_{j,p}(\omega) = O_{j,1}(\omega) - O_{j,0}(\omega)$ for each individual output j, $1 \le j \le m$;

Use observed input $I_p(\omega)$ and output $O_{j,p}(\omega)$ to estimate the j^{th} row of the S through multi-input system identification as in Lemma 2;

5. Iteration gain selection:

Estimate bounds $\overline{\Delta}_{j,l}$ on uncertainty $\Delta_{j,l}$ as in Eq. (3.18);

Compute the upper bound $\overline{\rho}_i(\omega)$ from Eq. (3.33);

Select a nonzero iteration gain $\rho(\omega)$ if conditions of Lemma 5 can be met; otherwise set to zero. Also set to zero if frequency ω is beyond the desired tracking bandwidth;

6. Iterative input correction: $2 \le k \le k_{max}$.

Obtain $I_2(\omega) = I_0(\omega) + \hat{S}^{\dagger}(\omega)\rho(\omega)(O_d(\omega) - O_0(\omega))$;

Apply $I_2(\omega)$ to the system and measure $O_2(\omega)$;

Compute tracking error $E_2(\omega) = O_d(\omega) - O_2(\omega)$ and its time domain representation $E_2(t)$;

Compute maximum tracking error $\overline{E}_{j,2} = \max_t |E_{j,2}(t)|$; **while** There exists $j \in [1, m]$ such that $\overline{E}_{j,k} \ge \epsilon$ **and** $k \le k_{max}$ **do**

$k = k + 1$;

Compute I_k from Eq. (3.4) using I_{k-1} and O_{k-1};

Apply $I_k(\omega)$ to the system and measure $O_k(\omega)$;

Compute $E_{j,k}(t) = O_{j,d}(t) - O_{j,k}(t)$ and $\overline{E}_{j,k}$;

end while

3.3 Experiment results and discussion

The performance of a 3-DOF SEA robot were comparatively evaluated, with and without ILC, in an experiment that mimics pilot hole cleaning in confined spaces, as illustrated in Fig. 3.1.

3.3.1 ILC for hole cleaning

3.3.1.1 Experimental system

A low-profile 3-DOF robotic arm was used in the experiment to mimic pilot hole cleaning in confined spaces, as illustrated in Fig. 3.1. The joint actuators were HEBI X5-4 series elastic actuators, and the links between the joints were PVC black pipes with diameter $\varnothing$ = 1.25 inch and lengths l_1 = 16.90 cm and l_2 = 17.97 cm. The brush (Forney 70485 Tube Brush) had a bristle diameter $\varnothing$ = 12 mm, and the effective length to the tip of the end-effector brush from the center of the joint θ_3 actuator was length l_3 = 15.86 cm as shown in Fig. 3.1. The plate in the front of the robot was drilled with evenly-spaced holes of diameter $\varnothing$ = 6.2 mm to represent a part to be cleaned with the robot. A MATLAB interface with relevant HEBI libraries were used to send commands to and receive data from the robotic arm, and data processing was done with MATLAB. The sampling rate for the input and output were 100 Hz. The internal feedback frequency of the SEA robot was also set as 100 Hz.

3.3.1.2 Task description

The operation studied here is the cleaning of a single hole, which requires the robot to execute a periodic forward-and-backward movement of the brush tip in the Y direction in Fig. 3.1. The desired position $Y = Y_d$ is described by its acceleration $\ddot{Y}_d$, for time $t \in [0,\ t_f]$, as

$$\begin{aligned} \ddot{Y}_d(t) &= A\sin(\omega_T(t-\underline{t}_{k_c})) && \underline{t}_{k_c} \le t < t_{k_c} \\ &= \ddot{Y}_d(t_{k_c}-(t-t_{k_c})) && t_{k_c} \le t < \bar{t}_{k_c} \\ &= 0 && \text{otherwise} \end{aligned} \tag{3.34}$$

with initial conditions $\dot{Y}_d(0) = 0, Y_d(0) = \underline{Y}$, where the amplitude of the acceleration is $A = \frac{8\pi d}{T^2}$ and frequency $\omega_T = \frac{4\pi}{T}$ with T as the time period for each forward-and-backward motion, d as the stroke length, which is kept fixed at 5 cm in the following, and $\underline{Y} = l_b + l_3 = 39.13$ cm represents the situation when the tip of the brush is just touching the plane of the plate with holes as in Fig. 3.1. Moreover, $\underline{t}_{k_c} = t_1 + k_c T$, $t_{k_c} = t_1 + (k_c + 0.5)T$, and $\bar{t}_{k_c} = t_1 + (k_c + 1)T$, with integer $0 \leq k_c < (k_N - 1)$, where $k_N = 20/T$ is the number of cleaning cycles, $t_1 = 20$ s is the amount of initial and final period without motion before and after the cleaning cycles, and the final time is $t_f = 2 * t_1 + k_N T$. An example trajectory Y_d with time period $T = 0.5$ s is shown in Fig. 3.2. The desired position $X = X_d$ of the brush tip is at the center of the hole to be cleaned, and the brush is to be held perpendicular to the plate with the holes, i.e., the angle Θ in Fig. 3.1 is to be kept constant at the desired value $\Theta_d = \pi/2$ rad.

3.3.1.3 System input and output

The controlled output in the experimental system were the local joint angles $O = [\theta_1, \theta_2, \theta_3]^T$ as in Fig. 3.1, and the control input I were the reference joint angles $I = [\theta_{1,r}, \theta_{2,r}, \theta_{3,r}]^T$ applied to the feedback-based controllers at each joint. The brush tip trajectory X, Y, Θ are related to the output O as

$$Y(t) = l_1 \cos(\Phi_1(t)) + l_2 \cos(\Phi_2(t)) + l_3 \cos(\Phi_3(t)), \tag{3.35}$$

$$X(t) = -l_1 \sin(\Phi_1(t)) - l_2 \sin(\Phi_2(t)) - l_3 \sin(\Phi_3(t)), \tag{3.36}$$

$$\Theta(t) = \Phi_3(t) + \pi/2, \tag{3.37}$$

where $\Phi_k(t) = \sum_{i=1}^{k} \theta_i(t)$. Consequently, the desired output O_d (i.e., the desired joint angles $\theta_{j,d}$, $1 \leq j \leq 3$) can be obtained from the known desired tip position X_d, Y_d and orientation

Θ_d, as

$$\theta_1(t) = -\arctan\left(\frac{-l_a}{l_b + l_s(t)}\right) - \arccos\left(\frac{(l_b + l_s(t))^2 + l_a^2 + l_1^2 - l_2^2}{2l_1\sqrt{l_a^2 + (l_b + l_s(t))^2}}\right), \tag{3.38}$$

$$\theta_2(t) = \pi - \arccos\left(\frac{l_1^2 + l_2^2 - (l_a^2 + (l_b + l_s(t))^2)}{2l_1 l_2}\right), \tag{3.39}$$

$$\theta_3(t) = -(\theta_1(t) + \theta_2(t)), \tag{3.40}$$

where $l_s(t) = Y(t) - \underline{Y}$, $l_b = l_1\cos(\Phi_1(0)) + l_2\cos(\Phi_2(0))$ and $l_a = |-l_1\sin(\Phi_1(0)) - l_2\sin(\Phi_2(0))|$, and the initial pose of the robot in Fig. 3.1 yields $\Phi_1(0) = -0.6756$ rad, $\Phi_2(0) = 1.0007$ rad, and $\Phi_3(0) = 0$ rad. Finally, given the initial pose of the robot, θ_1 is always negative for the specific hole to be cleaned.

3.3.2 Need for ILC

If the brush is to be moved slowly (with large time period T), then the robot's joint controllers can successfully track the reference joint angles with sufficient precision, i.e., with the reference input $I = O_d$ at iteration step $k = 0$, the achieved output O is close to the desired output trajectory $O \approx O_d$. However, if the brush is moved vigorously (with small time period T), then the tracking is not as good, as seen in Fig. 3.3. Note that as the operation speed increases, the output O, i.e., joint angles $\theta_{j,0}$ (at initial iteration step) do not follow the desired joint angles $\theta_{j,d}$. In particular, as time period T decreases, the maximum value of the joint-tracking error $\overline{E}_{j,0}$ increases by 4 times from $\overline{E}_{1,0} = 0.028$ rad, $\overline{E}_{2,0} = 0.057$ rad, $\overline{E}_{3,0} = 0.016$ rad at time period $T = 10$ s to $\overline{E}_{1,0} = 0.112$ rad, $\overline{E}_{2,0} = 0.230$ rad, $\overline{E}_{3,0} = 0.103$ rad at time period $T = 0.5$ s as seen in Fig. 3.4 and quantified in Table 3.1.

While the brush is flexible enough to handle some distortion, repeated large errors in the positioning in the X direction and in the orientation angle Θ can damage the brush. Therefore, ILC in Eq. (3.4) is used to improve the positioning precision by correcting for motion-induced errors in the desired output O_d with time period $T = 0.5$ s as in Fig. 3.3. Note that the ILC procedure includes steps to find local models (that includes contact effects) for each hole cleaning task. Both the modeling, and the iterative corrections have

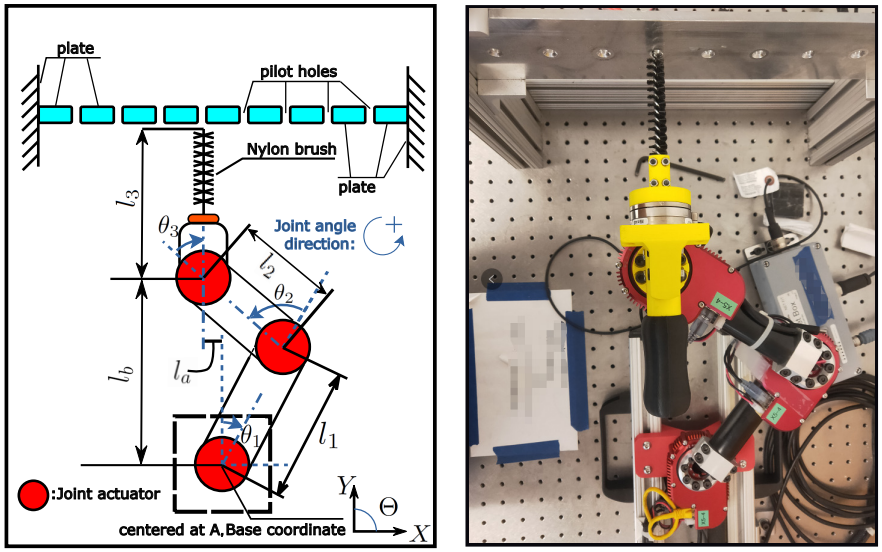

Figure 3.1: Schematic drawing (left) and top view (right) of the experimental SEA robot. The cleaning task for a specific pilot hole consists of letting the brush achieve a periodic forward-backward motion with stroke length d, which should be perpendicular to the plate, i.e, end-effector orientation $\Theta = \pi/2$ rad. The controlled outputs are the local joint angles $\theta_1, \theta_2, \theta_3$. The pose shown in the figure depicts the initial pose at the start of the hole-cleaning task.

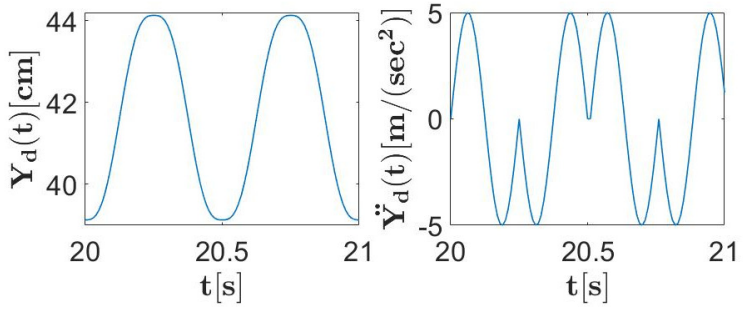

Figure 3.2: Desired motion Y_d (left) and acceleration $\ddot{Y}_d$ (right) of the brush tip in the Y direction during $t \in [20, 21]$ s.

to be repeated for holes that are far from each other if there is substantial change in robot pose. Nevertheless, an advantage in the proposed application is that the ILC can be carried out ahead of time, outside of the confined space, provided the pose and support of the robot is similar when placed in the confined space.

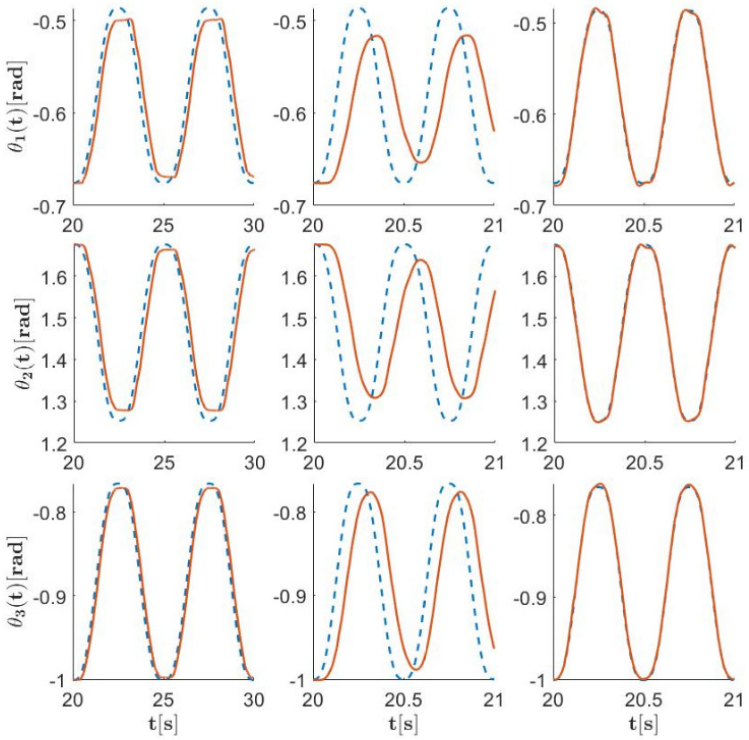

Figure 3.3: Comparison of desired output O_d (dashed line) and achieved output O (solid line) with and without ILC for three cases: (left) slower trajectories with time period $T = 5$ s without ILC; (middle) faster trajectories with time period $T = 0.5$ s without ILC; and (right) faster trajectories with time period $T = 0.5$ s with ILC.

3.3.3 ILC methods

The MIMO ILC experiments followed Algorithm 1 to correct positioning errors during fast cleaning, with time period $T = 0.5$ s.

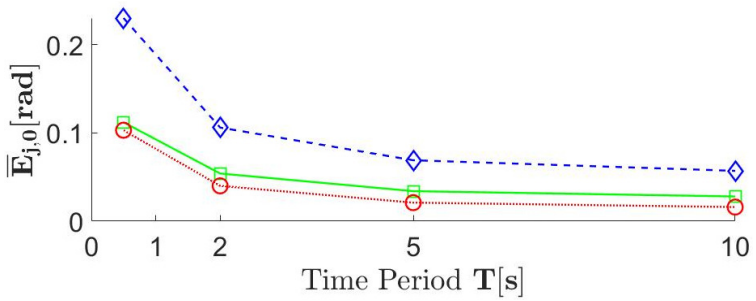

Figure 3.4: Joint-tracking error $\overline{E}_{j,0}$ defined in Algorithm 1 increases as time period T decreases, i.e., for faster cleaning motion: $\overline{E}_{1,0}$ (square), $\overline{E}_{2,0}$ (diamond) and $\overline{E}_{3,0}$ (circle).

Table 3.1: Impact of faster cleaning motion (smaller time period T) on joint-tracking error $\overline{E}_{j,0}$ defined in Algorithm 1.

Time Period, T[s]	$\overline{\mathbf{E}}_{\mathbf{1,0}}$ [rad]	$\overline{\mathbf{E}}_{\mathbf{2,0}}$ [rad]	$\overline{\mathbf{E}}_{\mathbf{3,0}}$ [rad]
0.5	0.112	0.230	0.103
2	0.054	0.106	0.040
5	0.034	0.069	0.021
10	0.028	0.057	0.016

3.3.3.1 Initial input

In the initial ILC step $k = 0$, the desired output O_d was selected as the desired joint angles $\{\theta_{j,d}\}_{j=1}^{3}$ computed from the known desired brush-tip trajectory $\{X_d, Y_d, \Theta_d\}$ using Eqs. (3.38) to (3.40), as shown in Fig. 3.5. The initial input $I_0 = O_d$ was applied to the SEA robot and the output O_0 was measured.

3.3.3.2 Perturbed input and model estimation

The input perturbation I_p in ILC step $k = 1$ can be selected to be frequency rich and provide the persistence of excitation needed for model acquisition [47]. $O_{j,p}$ represents the output caused by the input perturbation I_p. For the experiments, the input perturbationI_p

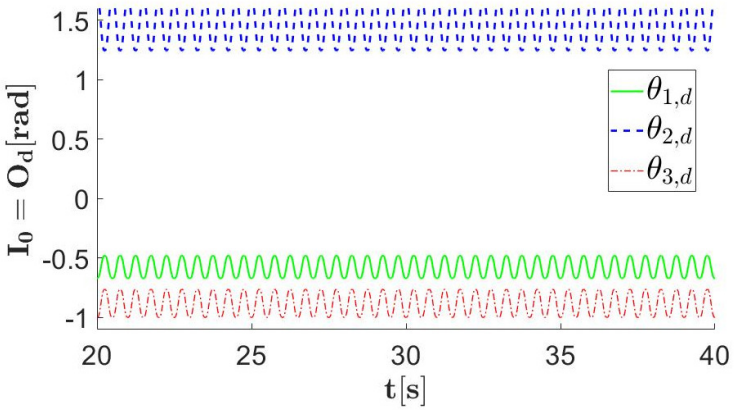

Figure 3.5: The desired output O_d, i.e., desired joint angles $[\theta_{1,d}, \theta_{2,d}, \theta_{3,d}]^T$, which are held constant outside the shown time interval at $[\theta_{1,d}(0), \theta_{2,d}(0), \theta_{3,d}(0)]^T = [-0.6756, 1.6763, -1.0007]^T$rad.

was chosen to be the sum of chirp signals (C_p) and staircase functions (H_p), with different patterns $I_{j,p} = C_{j,p} + H_{j,p}$ for each joint j, as illustrated in Fig. 3.7. The chirp functions were, for time $t \in [0, 60]$ s and frequency $\omega_c = 0.3$ hz,

$$C_{1,p}(t) = 0.012 \sin(2\pi\omega_c t_{1,c}^2), \quad \forall\ 20 \le t \le 40$$

and zero otherwise, where $t_{1,c} = \mod(t - 20 + \sqrt{100/3}, 20)$, with $t \in [0, 60]$ s for joint 1. For joint 2,

$$C_{2,p}(t) = 0.022 \sin(2\pi\omega_c (40 - t)^2), \quad \forall 20 \le t \le 40$$

and zero otherwise, and for joint 3,

$$C_{3,p}(t) = \begin{cases} -0.012 \sin(2\pi\omega_c (t_{3,c} - 10)^2) & 10 \le t_{3,c} \\ 0.012 \sin(2\pi\omega_c (20 - t_{3,c})^2) & t_{3,c} < 10 \\ 0 & \text{otherwise,} \end{cases}$$

where $t_{3,c} = \mod(t - 30 + \sqrt{50}, 20)$, and the staircase functions consist of three consecutive 5-second steps starting from $t = t_H$ with magnitude equaling to h_a, h_b and h_c, and are zero otherwise. The parameters for each staircase function $H_{j,p}$ are tabulated in Table 3.2.

Table 3.2: Parameters for staircase functions H_p.

staircase index	t_H [s]	h_a [rad]	h_b [rad]	h_c [rad]
$H_{1,p}$	24	+0.002	-0.002	+0.002
$H_{2,p}$	23	-0.003	+0.003	-0.003
$H_{3,p}$	21	+0.002	-0.002	+0.002

From linearity, the perturbation input-output relation was, from Eq. (3.1), $O_p(\omega) = S(\omega)I_p(\omega)$.

The observed perturbation O_O for each joint angle $1 \leq j \leq 3$, i.e., $O_{j,O} = O_{j,p}$ along with the input I_p were used to estimate the model subsystems $S_{j,l}$ (with $1 \leq l \leq 3$) and the associated variance $\mathbb{V}_{j,l}$, from Eq. (3.14) and Eq. (3.15), through the input-weighted complex kernel as in Lemma 2. The necessary Fourier transforms and inverse Fourier transforms were computed in MATLAB. Note that, as discussed earlier, the current MIMO ILC approach is valid with any SISO kernel $\hat{k}_{j,l}$, which was selected for subsystems $S_{j,l}$, at different frequencies ω_1, ω_2, as $\hat{k}_{j,l}(\omega_1, \omega_2) = \sigma^2_{f,j,l} \exp\left(-\frac{1}{2}(\omega_1 - \omega_2)^* l^{-2}_{j,l}(\omega_1 - \omega_2)\right)$, where $\sigma_{f,j,l}$ and $l_{j,l}$ denote the output variance and length scale for the sub-system $S_{j,l}$, respectively. Then, the estimated subsystems $\hat{S}_{j,l}$ and their variance $\mathbb{V}_{j,l}$ are shown in Fig. 3.6.

3.3.3.3 Iteration gain selection

The iteration gain $\rho(\omega)$ was selected to ensure ILC convergence based on the estimated model and uncertainty. Bounds $\delta_{j,l}(\omega)$ on the model uncertainty $\Delta_{j,l}(\omega)$ were obtained with $\gamma_\Delta = 3$ in Eq. (3.17) of Remark. 6 to cover most (99.7%) of the potential uncertainties. The iteration gains $\rho_i(\omega)$ $(i = 1, 2, 3)$ were chosen to be 0.7 for $0 \leq \omega \leq 5$. Moreover, since the the desired output O_d did not have significant frequency content beyond 6 Hz, the iteration gains were reduced to zero after 6.5 Hz, as $\rho_i(\omega) = \rho_i(5)(1 - \frac{\omega - 5}{1.5})^2$ for $5 < \omega \leq 6.5$. The upper bound $\overline{\rho}_i(\omega)$ and the selected iteration gain $\rho_i(\omega)$ are shown in Fig. 3.8.

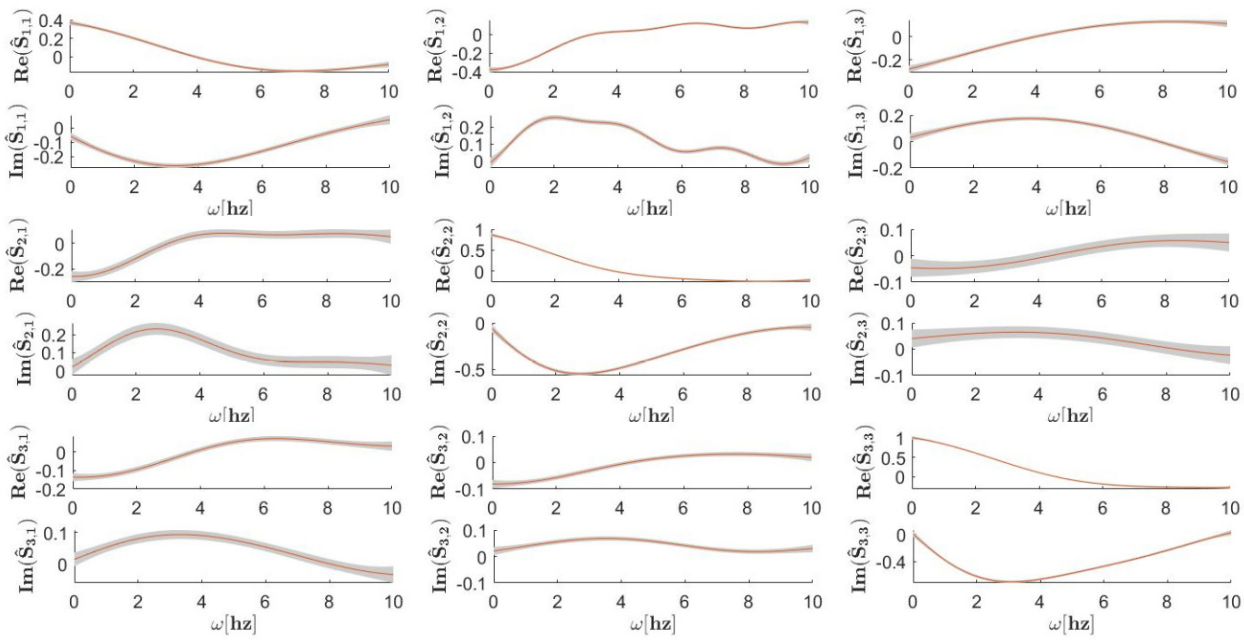

Figure 3.6: Bode frequency-response plots. Estimated model $\hat{S}$ of the system S defined in Eq. (3.1). The red lines are the expected values from Eq. (3.14) and deviation of $\pm\mathbb{V}_{j,l}(\omega)$ shown in gray, with the variance $\mathbb{V}_{j,l}(\omega)$ defined in Assumption 2.

3.3.3.4 Iterative input update

At each iteration step $k \geq 3$, the error $E_{k-1} = O_d(t) - O(t)$ during the active cleaning period ($t \in [20, 40]s$) was computed using Fourier transform in MATLAB, and used to update the input I_{k-1} to find the new input I_k. For iteration step $k = 2$, the new input I_2 was updated based on input I_0 and error E_0. Prior to the Fourier transform, the initial and final settling of the closed-loop controllers beyond the cleaning cycle were removed in all iterations by padding the error signal in time $E_{k-1}(t)$ with zeros before and after the end of the cleaning cycles for 5 s and thereby, the input I_k was updated over the time interval $t \in [15, 45]s$.

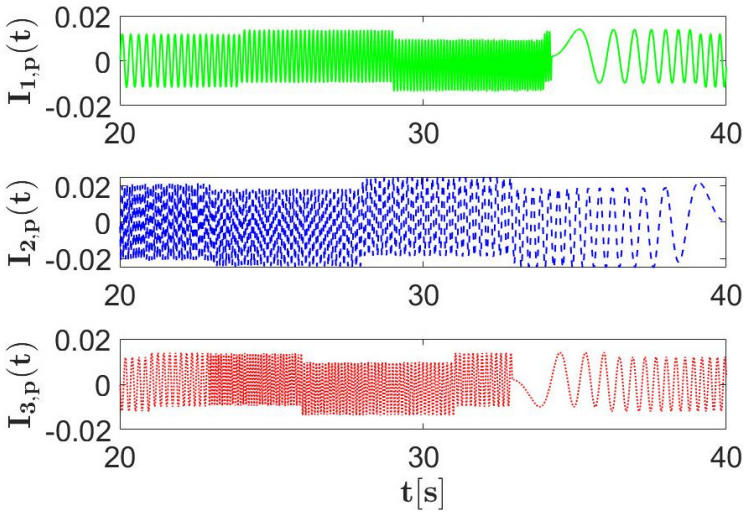

Figure 3.7: Input perturbation $\{I_{l,p}\}_{l=1}^{3}$ at ILC step $k = 1$ with a mixture of chirp and staircase signals were added at ILC step $k = 1$. The input perturbation I_p was zero outside the shown time interval.

3.3.4 Results and discussion

The ILC led to improvement in the positioning precision of the brush with the SEA robot, even in the presence of significant contact effects. The reduction of the joint tracking error $\overline{E}_{j,k}$, with iteration step k is shown in Fig. 3.9 and the tracking results are shown in Fig. 3.3.

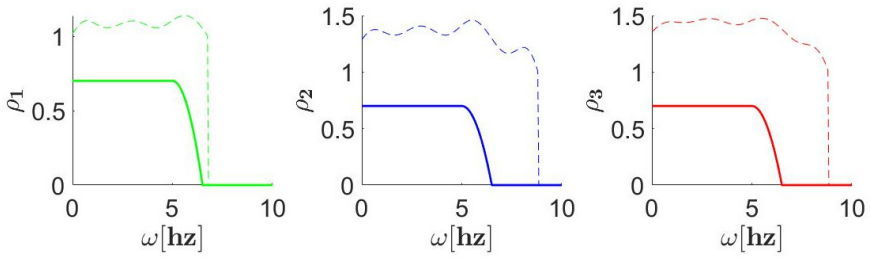

Figure 3.8: Selected iteration gain $\{\rho_i(\omega)\}_{i=1}^{3}$ (solid line) and upper bound $\{\overline{\rho}_i(\omega)\}_{i=1}^{3}$(dashed line) from Eq. (3.33).

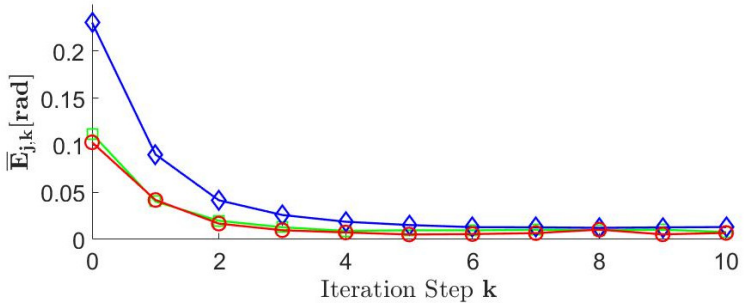

Figure 3.9: Reduction of joint error $\overline{E}_{j,k}$ with iteration step k: $\overline{E}_{1,k}$ (square), $\overline{E}_{2,k}$ (diamond) and $\overline{E}_{3,k}$ (circle).

The joint tracking error decreased from initial values of $\overline{E}_{1,0}$ = 0.112 rad, $\overline{E}_{2,0}$ = 0.230 rad, $\overline{E}_{3,0}$ = 0.103 rad at iteration step k = 0 to final values of $\overline{E}_{1,10}$ = 0.008 rad, $\overline{E}_{2,10}$ = 0.013 rad, $\overline{E}_{3,10}$ = 0.007 rad at iteration step 10. The final tracking errors were close to the repeatability of the system - the non-repeatable errors in the joint positioning of the robot were experimentally estimated to be 0.004 rad at joints 1 and 3, and 0.007 rad at joint 2. Thus, the ILC approach led to substantial reduction of 92% in joint θ_1, 94% in joint θ_2 and 93% in joint θ_3 in the tracking error.

An alternate approach to reduce the tracking error, without ILC, is to slow down the cleaning motion. In particular, with a time period T = 5 s, the tracking error without ILC was $\overline{E}_{1,0}$ = 0.034 rad, $\overline{E}_{2,0}$ = 0.069 rad, $\overline{E}_{3,0}$ = 0.021 rad. This is still larger than the final tracking error with ILC with a time period T = 0.5 s, as seen by comparing the desired and actual output joint angles for the time period T = 5 s without ILC in Fig. 3.3. Thus, the ILC enables at least 10-times increase in the SEA robot's operating speed for similar positioning precision.

3.4 Chapter conclusion

This chapter shows that the proposed complex-kernel Gaussian process regression with a proposed input-weighted kernel can sufficiently capture the model of a robot with series elas-

tic actuators for precision operations even in the presence of contact effects, which in general are challenging to model a priori. Experimental results showed more than an order increase in operating speed and around 90% improvement in the positioning precision. Additionally, this chapter developed theoretical conditions to ensure convergence of an iterative learning controller for multi-input multi-output systems. However, the proposed approach is only valid locally around an operating point where the error caused by nonlinearity is sufficiently small (e.g., for local cleaning operations as demonstrated in the chapter), and is not suitable for large-range motions with substantial nonlinearity.

Chapter 4

PRECISION DATA-ENABLED KOOPMAN-TYPE INVERSE OPERATORS FOR LINEAR SYSTEMS (MC2.1)

This chapter identifies the needed type of output information for Precision Data-enabled Koopman-type Inverse Operators for Linear Systems (**MC2.1**) and is based on a work published in Modeling, Estimation and Control Conference (MECC) 2022 [2]. The advent of easy access to large amount of data has sparked interest in directly developing the relationships between input and output of dynamic systems. A challenge is that in addition to the applied input and the measured output, the dynamics can also depend on hidden states that are not directly measured. The goal of this chapter is to identify the type of output data needed to develop inverse (output-to-input) operators, with a desired level of precision. Rather than the two step processes of first learning forward models and second using model-predictive control (MPC) to optimally select the control input, the proposed approach seeks to solve the inverse problem of directly finding the input for a given output, e.g., similar to [70, 71]. In particular, the relative degree of the system is used to identify the number of time derivatives that need to be added to input-output data to facilitate precision data-enabled learning of the inverse operator.

4.1 Problem formulation

The inverse operator is developed for linear time-invariant (LTI) single-input-single-output (SISO) system. Let the system be

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.1}$$

$$y(t) = Cx(t) \tag{4.2}$$

with states $x(t) \in \mathbb{R}^n$, input $u(t) \in \mathbb{R}$ and output $y(t) \in \mathbb{R}$ with matrices $A \in \mathbb{R}^n \times \mathbb{R}^n, B \in \mathbb{R}^n \times 1, C \in 1 \times \mathbb{R}^n$.

Assumption 3 (System properties)**.** *The system described in (4.1) and (4.2) is stable (i.e., A is Hurwitz), hyperbolic (no zeros on the imaginary axis), and has relative degree $r \leq n$ (i.e., the difference between the number of poles and the number of zeros).*

Assumption 4. *The desired output y_d, specified in inverse operator problems, is sufficiently smooth, and has bounded time derivatives up to the relative degree r.*

4.1.1 Hidden-state dependency

The system state x can split into state components ξ that directly depend on the output and its time derivatives

$$\xi(t) = \left[y(t), \dot{y}(t), \ldots, \tfrac{d^{r-1}y(t)}{dt^{r-1}}\right]' \in \mathbb{R}^{r\times 1} \tag{4.3}$$

and internal states η,

$$\begin{bmatrix}\xi(t)\\ \eta(t)\end{bmatrix} = Sx(t) \tag{4.4}$$

such that in the new coordinates, (4.1) can be written as, e.g., see [95], Example 4.1.3,

$$\dot{\xi}(t) = A_1\xi(t) + A_2\eta(t) + B_1u(t) \tag{4.5}$$

$$\dot{\eta}(t) = A_3y(t) + A_4\eta(t) \tag{4.6}$$

where

$$B_1 = \begin{bmatrix}0 & 0 & \vdots & b_{n-r}\end{bmatrix}' \in \mathbb{R}^{r\times 1}, \quad A_3 = \begin{bmatrix}0 & 0 & \vdots & 1/b_{n-r}\end{bmatrix}',$$

$$A_4 = \begin{bmatrix} 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \\ -b_0/b_{n-r} & -b_1/b_{n-r} & \ldots & -b_{n-r-1}/b_{n-r}, \end{bmatrix}$$

and the eigenvalues of matrix A_4 are the zeros of the transfer function of system (4.1) and (4.2).

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 + b_1 s + \cdots + b_{n-r} s^{n-r}}{a_0 + a_1 s + \cdots + a_{n-1} s^{n-1} + s^n}. \tag{4.7}$$

Note that the internal state η is only driven by the output $y = \xi_1$. Moreover, due to the relative degree r assumption, the input u is directly related to the r^{th} derivative of the output, and therefore, the r^{th} row of (4.5) can be written as

$$\begin{aligned} y^{(r)}(t) \triangleq \frac{d^r y(t)}{dt^r} &= CA^r x + CA^{r-1} B u(t) \\ &= CA^r S^{-1} \begin{bmatrix} \xi(t) \\ \eta(t) \end{bmatrix} + b_{n-r} u(t) \\ &= A_\xi \xi(t) + A_\eta \eta(t) + b_{n-r} u(t), \end{aligned} \tag{4.8}$$

and the matrices A_1 and A_2 in (4.5) are given by

$$A_1 = \left[\begin{array}{cccc} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ \hline & & A_\xi & \end{array}\right], \quad A_2 = \left[\begin{array}{cccc} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ \hline & & A_\eta & \end{array}\right].$$

where A_ξ and A_η are the last rows of matrices A_1 and A_2 respectively.

4.1.2 Research problem

The desired output and its derivatives, $(y_d^{(r)}, \xi_d)$ can be used to predict the inverse input u_d from (4.8) , as

$$u_d(t) = b_{n-r}^{-1} \left[y_d^{(r)}(t) - A_\xi \xi_d(t) - A_\eta \eta_d(t) \right], \tag{4.9}$$

which depends on the internal states η that are hidden or not directly measured. The goal is to minimize the hidden state effects on the inverse model, by addressing the following research problems.

1. Finding the hidden state from output: Develop an operator that maps the time history of the output y with length T to an estimate of the hidden state η at time t

$$\hat{\eta}(t) = \hat{\mathbb{H}}[y(t-T:t)]. \tag{4.10}$$

2. Koopman-type inverse operator: Using the operator in (4.10), develop a data-enabled Koopman-type inverse operator $\hat{\mathbb{G}}^{-1}$ that uses the history of the desired output and its time derivatives to predict the inverse input as

$$\hat{u}_d(t) = \hat{\mathbb{G}}^{-1}[y_d(t-T:t), \xi_d(t), y_d^{(r)}(t)]. \tag{4.11}$$

3. Inverse operator precision: Quantify the error $\|\hat{u}_d(t) - u_d(t)\|_2$ dependence on each argument of $\hat{\mathbb{G}}^{-1}$.

4.2 Solution

4.2.1 Finding the hidden state from output

If the system is minimum-phase (A_4 is Hurwitz), i.e., (4.7) has no zeros on the right half plane, then $\eta(t)$ can be obtained from the history of the output by solving (4.6)

$$\eta(t) = \int_{-\infty}^{t} e^{A_4(t-\tau)} A_3 y(\tau) d\tau \triangleq \mathbb{H}[y(-\infty:t)]. \tag{4.12}$$

In practice, such an operator is hard to capture in a data-enabled way since it requires an infinite window. Therefore, an estimate $\hat{\eta}$ is obtained with an approximate operator $\hat{\mathbb{H}}$ with a finite time history length T is defined

$$\hat{\eta}(t) \triangleq \int_{t-T}^{t} e^{A_4(t-\tau)} A_3 y(\tau) d\tau \triangleq \hat{\mathbb{H}}[y(t-T:t)]. \tag{4.13}$$

The approximate operator $\hat{\mathbb{H}}$ approaches the exact operator $\mathbb{H}$ exponentially as the time history T increases.

Lemma 6. *If the output trajectory is bounded,*

$$M = \max_{\tau \in [-\infty, t-T]} \|y(\tau)\|_2 < \infty, \tag{4.14}$$

then the error in computing the hidden state $\eta(t)$ decays exponentially with the time history T, i.e., there exists positive scalars $\alpha_1 > 0, \beta_1 > 0$ such that

$$\|\Delta\eta(t)\|_2 \triangleq \|\eta(t) - \hat{\eta}(t)\|_2 \leq \beta_1 e^{-\alpha_1 T}. \tag{4.15}$$

Proof. Since the system is assumed to be minimum phase, the and the eigenvalues of matrix A_4, which are the zeros of the transfer function of system (4.1), lie in the open left-half of the complex plane, i.e., the matrix A_4 is Hurwitz. Then, there exists positive scalars $\kappa_1 > 0, \alpha_1 > 0$ such that, [96]

$$\|e^{A_4 t}\|_2 \leq \kappa_1 e^{-\alpha_1 t}. \tag{4.16}$$

Then, from (4.12,4.13), the approximation error can be bounded as

$$\begin{aligned}
\|\eta(t) - \hat{\eta}(t)\|_2 &= \left\| \int_{-\infty}^{t-T} e^{A_4(t-\tau)} A_3 y(\tau) d\tau \right\|_2 \\
&\leq M\|A_3\|_2 \int_{-\infty}^{t-T} \kappa_1 e^{-\alpha_1(t-\tau)} d\tau \quad \text{using (4.14, 4.16)} \\
&= M\|A_3\|_2 \int_{T}^{+\infty} \kappa_1 e^{-\alpha_1 \tau'} d\tau' \\
&= M\|A_3\|_2 \frac{\kappa_1}{\alpha_1} e^{-\alpha_1 T}.
\end{aligned} \tag{4.17}$$

The result follows with

$$\beta_1 = M\|A_3\|_2 \frac{\kappa_1}{\alpha_1}. \tag{4.18}$$

□

4.2.2 Koopman-type inverse operator

Given an estimate $\hat{\eta}$ of the internal state η, the inverse operator prediction in (4.11) can be estimated as

$$\begin{aligned}\hat{u}_d(t) &= b_{n-r}^{-1}\left[y_d^{(r)}(t) - A_\xi \xi_d(t) - A_\eta \hat{\eta}_d(t)\right] \\ &= b_{n-r}^{-1}\left[y_d^{(r)}(t) - A_\xi \xi_d(t) - A_\eta \hat{\mathbb{H}}[y_d(t-T:t)]\right] \quad \text{using (4.13)} \\ &\triangleq \hat{\mathbb{G}}^{-1}[y_d^{(r)}(t), \xi_d(t), y_d(t-T:t)]. \end{aligned} \tag{4.19}$$

Remark 11. *In addition to sufficient time history (large T) of the output to accurately find the internal state (to let $\Delta\eta \longrightarrow 0$), information about the derivatives of the output (upto the relative degree r at time t, i.e., $y_d^{(r)}(t), \xi(t)$) are also needed for precisely computing the inverse input u_d in (4.11) as illustrated in Fig. 4.1.*

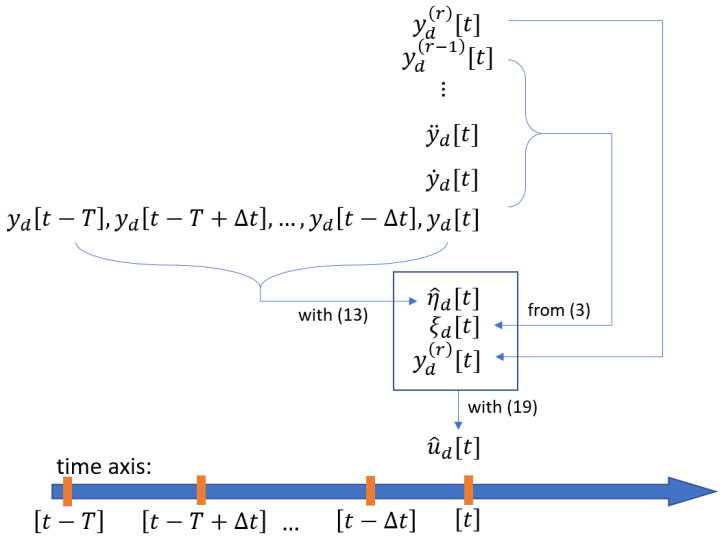

Figure 4.1: The inverse operator's dependence on the hidden state is removed by use of past output history and current time derivatives of the output.

4.2.3 Koopman-type forward operators using output history

The output y can be related to the input as

$$y(t+T_f) = C\int_{-\infty}^{t+T_f} e^{A(t-\tau)}Bu(\tau)d\tau \tag{4.20}$$

and approximated by

$$\hat{y}(t+T_f) = C\int_{t-T}^{t+T_f} e^{A(t-\tau)}Bu(\tau)d\tau. \tag{4.21}$$

Therefore, using arguments similar to the proof of Lemma 1, the error in computing the output using just the history of input u tends to zero as the time history of the input increases, i.e., as $T \to \infty$. Thus, it is possible to find a map that only depends on the input and its past history,

$$\hat{y}(t+T_f) = \hat{\mathbb{G}}_u[u(t-T:t+T_f)], \tag{4.22}$$

which justifies the use of ARX models to capture forward linear system models using past input history (and augmented by the output history). In contrast, with Koopman-type operators where past history of the observable output is used to predict future values, the forward model prediction can be written as

$$\begin{aligned}
\hat{y}(t+T_f) &= Ce^{AT_f}\hat{x}(t) + C\int_t^{t+T_f} e^{A(t+T_f-\tau)}Bu(\tau)d\tau \\
&= Ce^{AT_f}S^{-1}\begin{bmatrix}\xi(t)\\ \hat{\eta}(t)\end{bmatrix} + C\int_t^{t+T_f} e^{A(t+T_f-\tau)}Bu(\tau)d\tau \quad \text{using (4.4)} \\
&= Ce^{AT_f}S^{-1}\begin{bmatrix}\xi(t)\\ \hat{\mathbb{H}}[y_d(t-T:t)](t)\end{bmatrix} + C\int_t^{t+T_f} e^{A(t+T_f-\tau)}Bu(\tau)d\tau \quad \text{using (4.13)} \\
&\triangleq \hat{\mathbb{G}}[y(t-T:t), \xi(t), u(t:t+T_f)].
\end{aligned} \tag{4.23}$$

Therefore, past history of the output can also be used to develop Koopman-type forward operators, provided access is available to current time derivatives of the output $\xi(t)$.

4.2.4 Inverse operator precision

The inverse operator depends not only on the past history of the output (to remove the hidden state η dependency) but also on the output and its time derivatives at the current time instant t. The impact of the time history T, output and its time derivatives on the precision of the operator is quantified in the next lemma.

Lemma 7. *The prediction error of the inverse operator is bounded, i.e there exists positive scalars $L_1 > 0, L_2 > 0, L_3 > 0$ such that the error between the predicted input $\hat{u}_d(t)$ and the true input $u_d(t)$ is*

$$\|\hat{u}_d(t) - u_d(t)\|_2 \leq L_1 \|\Delta y_d^{(r)}(t)\|_2 + L_2 \|\Delta \xi_d(t)\|_2 + L_3 e^{-\alpha_1 T}. \tag{4.24}$$

Proof. From (4.9) and (4.19),

$$\|\hat{u}_d(t) - u_d(t)\|_2 \leq |b_{n-r}^{-1}| \Big(\|\Delta y_d^{(r)}(t)\|_2 + \|A_\xi\|_2 \|\Delta \xi_d(t)\|_2 + \|A_\eta\|_2 \|\Delta \eta_d(t)\|_2 \Big), \tag{4.25}$$

where $\Delta y_d^{(r)}(t) \triangleq \hat{y}_d^{(r)}(t) - y_d^{(r)}(t)$, $\Delta \xi_d(t) \triangleq \hat{\xi}_d(t) - \xi_d(t)$ and $\Delta \eta_d(t) \triangleq \hat{\eta}_d(t) - \eta_d(t)$. The results follows from (4.15) with

$$L_1 = |b_{n-r}^{-1}|, \quad L_2 = L_1 \|A_\xi\|_2, \quad L_3 = L_1 \|A_\eta\|_2 \beta_1. \tag{4.26}$$

□

Remark 12 (Data-enabled algorithm)**.** *Known values of the desired output and its derivatives, specified with a sampling period Δt and time history T can be used to estimate a discrete-time inverse operator from (4.19) as*

$$\hat{u}_d[m] = \mathbb{G}_d^{-1}[y_d[m - m_T : 1 : m], \xi_d[m], y_d^{(r)}[m]], \tag{4.27}$$

where $[m]$ indicates value at time $t_m = m\Delta t$, and $m_T = T/\Delta t$. Data-enabled algorithms can be used to learn the operator $\mathbb{G}_d^{-1}$, since (4.27) maps a finite number of variables (desired output and its time derivatives) to the inverse input at time t_m.

4.3 Simulation results

In this section, an example system is introduced, followed by the data-enabled learning of the inverse operator.

4.3.1 Example system

Consider the following two-mass-spring-damper system, where the input u is the force acting on mass m_2 and its displacement x_2 is the output y, as shown in Fig. 4.2.

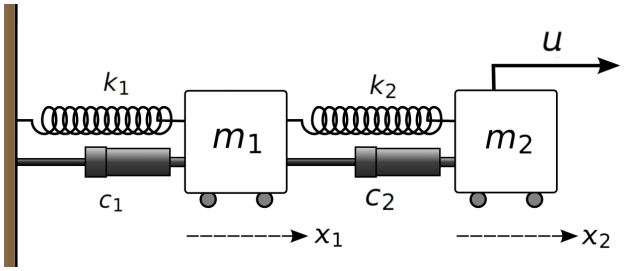

Figure 4.2: Example system plot

The corresponding state space model can be written as

$$\frac{d}{dt}X = AX + Bu \tag{4.28}$$

$$y = x_2 = CX \tag{4.29}$$

where $X \triangleq \begin{bmatrix} x_1 & \dot{x}_1 & x_2 & \dot{x}_2 \end{bmatrix}'$, $C = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1+k_2}{m_1} & -\frac{c_1+c_2}{m_1} & \frac{k_2}{m_1} & \frac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & \frac{c_2}{m_2} & -\frac{k_2}{m_2} & -\frac{c_2}{m_2} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ a/m_2 \end{bmatrix}, \tag{4.30}$$

$m_1 = 10, m_2 = 5, k_1 = 110, c_1 = 68, a = k_1/2, k_2 = 75$ and $c_2 = 60$ in SI units. The relative degree of the system is $r = 2$ and the input-output relation is given by

$$\ddot{y}(t) = -25y(t) - 12\dot{y}(t) + 25x_1(t) + 12\dot{x}_1(t) + 11u(t). \tag{4.31}$$

4.3.2 Preliminary selections

The selection of the data-enabled model types to evaluate, the sampling time (which needs to be sufficiently small to reduce discretization error), the evaluation metric, and sufficiently smooth output trajectories for model evaluation are described below.

1. A two-layer feedforward neural-net (created via MATLAB function **feedforwardnet()** with default activation function) is used to learn the inverse operator from data.

2. For the two-layer neural net, each model pool consists of 5 candidates with different number $N \in \{5, 10, 20, 40, 80\}$ of neurons in the hidden layer

3. The sampling frequency is varied from 5 Hz to 20 Hz, which is substantially higher than the system bandwidth of 1.7 Hz.

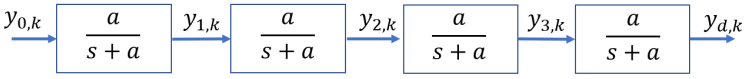

Figure 4.3: Filter process to generate desired trajectories.

4. The inverse operator is assessed using 10 different desired trajectories $y_{d,k}(t), 1 \leq k \leq 10, t \in [0, 10]$ with a fixed prediction sampling time of 0.01 s. Each desired trajectory $y_{d,k}$ used for assessment needs to be sufficiently smooth to investigate the impact of different order of output's time derivatives on the inverse operator, although from (4.24) the expectation is that only output derivatives up to the r^{th} order (r = 2 for this example) are required. Therefore, nominal trajectories $y_{0,k}$ (specified in Section 4.3.3)

are filtered as shown in Fig. 4.3, to obtain desired outputs $y_{d,k}$ and their derivatives as

$$\begin{bmatrix} y_{d,k} \\ \dot{y}_{d,k} \\ \ddot{y}_{d,k} \\ y^{(3)}_{d,k} \\ y^{(4)}_{d,k} \end{bmatrix}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -a & a & 0 & 0 & 0 \\ a^2 & -2a^2 & a^2 & 0 & 0 \\ -a^3 & 3a^3 & -3a^3 & a^3 & 0 \\ a^4 & -4a^4 & 6a^4 & -4a^4 & a^4 \end{bmatrix} \begin{bmatrix} y_{d,k} \\ y_{3,k} \\ y_{2,k} \\ y_{1,k} \\ y_{0,k} \end{bmatrix}(t) \tag{4.32}$$

where $a = 2\pi$ (cut-off frequency as 1 Hz), which is less than the system's bandwidth of 1.7 Hz, and example trajectories are shown in Fig. 4.4.

5. For a given time history T and sampling time Δt, as in Remark 12, the evaluation metrics for the data-enabled inverse operator with N neurons in the hidden layer are selected as the mean $e_{u,N}$ and maximum $\overline{e}_{u,N}$ normalized prediction error over the ten evaluation trajectories $y_{d,k}(\cdot)$ (defined in Section 4.3.3), i.e.,

$$e_{u,N} = \frac{1}{10}\sum_{k=1}^{10} \frac{\max_m |\hat{u}_k[m] - u_{d,k}[m]|}{\max_m |u_{d,k}[m]|} \times 100\% \tag{4.33}$$

$$\overline{e}_{u,N} = \max_{k=1,\ldots,10} \frac{\max_m |\hat{u}_k[m] - u_{d,k}[m]|}{\max_m |u_{d,k}[m]|} \times 100\%, \tag{4.34}$$

where the ideal inverse $u_{d,k}$ was found using (4.9) where η_d was obtained through (4.12). Moreover, the smallest normalized prediction error over different numbers of neurons in the hidden layer is defined as

$$e_u = e_{u,N^*}, \quad \bar{e}_u = \bar{e}_{u,N^*} \quad \text{where} \quad N^* = \arg\min_N e_{u,N} \tag{4.35}$$

to quantify the precision of the inverse operator.

4.3.3 Evaluation trajectories $y_{d,k}$

Expressions of $y_{0,k}(t)$ for $k = 1, 2, \ldots, 10$ and $0 \le t \le 10$, that are sequentially filtered (as in Fig. 4.3) to obtain the evaluation trajectories $y_{d,k}$.

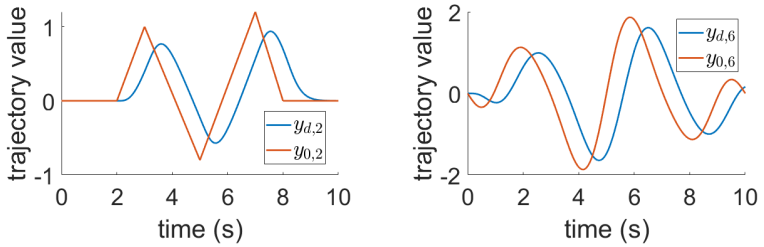

Figure 4.4: Comparison of the example filtered desired output $y_{d,k}$ and nominal trajectories $y_{0,k}$ for $k = 2$ (triangular) and $k = 6$ (sinusoidal).

Trapezoidal shape ($k = 1$), Triangle wave ($k = 2$)

$$y_{0,1}(t) = \begin{cases} 0.4(t-1) & 1 \le t < 3 \\ 0.8 & 3 \le t < 6 \\ 0.4(8-t) & 6 \le t < 8 \\ 0 & \text{otherwise.} \end{cases}, \quad y_{0,2}(t) = \begin{cases} t-2 & 2 \le t < 3 \\ 3.7 - 0.9t & 3 \le t < 5 \\ t - 5.8 & 5 \le t < 7 \\ 1.2(8-t) & 7 \le t < 8 \\ 0 & \text{otherwise.} \end{cases}$$

Square wave ($k = 3$), Serrated wave mixture ($k = 4$)

$$y_{0,3}(t) = \begin{cases} 1 & 2 \le t < 4 \\ -1 & 4 \le t < 6 \\ 1 & 6 \le t < 8 \\ 0 & \text{otherwise.} \end{cases}, \quad y_{0,4}(t) = \begin{cases} 2(t-1)/3 & 1 \le t < 2.5 \\ 2(4-t)/3 & 2.5 \le t < 4 \\ 8(t-4)/15 & 4 \le t < 5 \\ 8(6-t)/15 & 5 \le t < 6 \\ 0.4(t-6) & 6 \le t < 7.5 \\ 0.4(9-t) & 7.5 \le t < 9 \\ 0 & \text{otherwise.} \end{cases}$$

Monotonic ($k = 5$): $y_{0,5}(t) = 0.001(x^{3.2} - x^2)$

Sine wave #1 ($k = 6$): $y_{0,6}(t) = \sin(0.4\pi t) - 0.9\sin(0.6\pi t) + 0.2\sin(\pi t)$

Sine wave #2 ($k = 7$): $y_{0,7}(t) = 1.5\sin(0.7\pi t) - 0.5\sin(0.4\pi t)$

Sine wave #3 ($k = 8$): $y_{0,8}(t) = -0.5\sin(0.3\pi t) - 0.6\sin(0.7\pi t) + 0.2\sin(1.2\pi t)$

Sine wave #4 ($k = 9$): $y_{0,9}(t) = 0.7\sin(0.26\pi t) + 0.3\sin(1.3\pi t) - 0.2\sin(1.4\pi t)$

Slow chirp wave ($k = 10$): $y_{0,10}(t) = 0.35\sin(x^{1.5})$.

4.3.4 Data collection

The inverse operators are trained using input-output data collected from simulations. Both noisy and noise free output data are used to assess the impact of noise. The input signal u applied to the system is constructed by concatenating 20 cycles of $p_{(f_i,\alpha_i)}(\cdot)$ ($i = 1, 2, 3, \ldots, 20$) with different parameters, which are tabulated in Table. 4.1.

$$p_{(f_i,\alpha_i)}(t) = \alpha_i[4\sin(\pi c t^2) + s(t) + r(t)] \tag{4.36}$$

where $c = f_i/10$,

$$\mathrm{s}(t) = \begin{cases} 1 & 2 \le t < 4 \\ -0.9 & 4 \le t < 6 \\ 0.5 & 6 \le t < 8 \\ 0 & \text{otherwise,} \end{cases} \qquad \mathrm{r}(t) = \begin{cases} 0.4t & 0 \le t < 1 \\ 0.4 & 1 \le t < 9 \\ \mathrm{r}(10-t) & 9 \le t \le 10. \end{cases}$$

Table 4.1: Parameters of $p_{(f_i,\alpha_i)}$ in Eq. (4.36).

i	f_i	α_i	i	f_i	α_i	i	f_i	α_i	i	f_i	α_i
1	6	0.75	6	0.3	0.3	11	1	0.25	16	0.5	-0.1
2	3	0.5	7	0.1	0.3	12	0.5	0.25	17	2	0.25
3	2	0.5	8	0.5	-0.3	13	1	-0.1	18	1	0.1
4	0.5	0.5	9	0.3	-0.3	14	0.5	-0.05	19	0.5	0.05
5	0.5	0.3	10	0.1	-0.3	15	0.5	0.1	20	1	0.5

For the noisy case, additive white gaussian noise with signal-to-noise ratio of 20 is separately added to each output and its time derivatives. Simulations were done in MATLAB

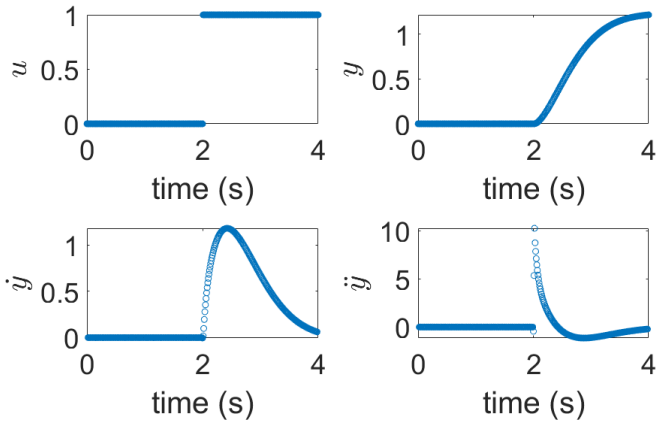

Figure 4.5: Identifying the relative degree r from input-output data, based on discontinuity in the r^{th} derivative of the output for a step input.

with `ode45()` with sampling rate of 100 Hz (to be consistent with the evaluation metrics from (4.33) to (4.35)). Input, output and the output's time derivatives (up to the fourth order) were collected. Second order derivative was obtained from (4.31). Third and fourth order derivatives for training purposes were estimated from the data, using finite difference as,

$$\begin{bmatrix} y^{(3)}[m] \\ y^{(4)}[m] \end{bmatrix} = \frac{1}{12(\Delta t)} \begin{bmatrix} -1 & 8 & 0 & -8 & 1 \\ -\frac{1}{\Delta t} & \frac{16}{\Delta t} & -\frac{30}{\Delta t} & \frac{16}{\Delta t} & -\frac{1}{\Delta t} \end{bmatrix} \begin{bmatrix} \ddot{y}[m+2] \\ \ddot{y}[m+1] \\ \ddot{y}[m] \\ \ddot{y}[m-1] \\ \ddot{y}[m-2] \end{bmatrix}. \tag{4.37}$$

4.3.5 Reducing the impact of hidden states using output history

To investigate the reduction of the impact of the hidden states on the prediction precision of data-enabled inverse operators, the performance of the data-enabled inverse operators was assessed for different time history T of the output. In this part of the study, the number of time derivatives of the output used was the same as the relative degree of the example system.

The relative degree $r = 2$ can be established by applying a step input — a corresponding discontinuity will appear in $y^{(r)}$, while the lower order derivatives ($y, \dot{y}$ in this example) remain continuous as seen in Fig. 4.5. Then, from (4.27),

$$\hat{u}_d[m] = \mathbb{G}_d^{-1}[y_d[m - m_T : 1 : m], \dot{y}_d[m], \ddot{y}_d[m]]. \tag{4.38}$$

The inverse operator's prediction error e_u (4.35) was obtained for varying output time history T ([0.1, 0.2, 0.4, 0.8, 1.6, 3.2, ...] s), for different sampling time $\Delta t \in \{0, 05s, 0.1s, 0.2s\}$, and for different number N of neurons in the hidden layer, and plotted in Fig. 4.6 for the case without noise in the training data. The associated prediction errors are tabulated in Table 4.2 for the fastest sampling time $\Delta t = 0.05$ s.

The precision of the inverse operator improves with larger output time history T, as seen in Table 4.2, where the evaluation values of the two-layer neural net with different N neurons in the hidden layer are listed. Note that typically $N^* \leq 20$ yields good precision for this application from Table 4.2. Over all selections of neuron numbers N, the variation of the smallest prediction error $e_u = e_{u,N^*}$ (4.35) with sampling time of $\Delta t = 0.05$ s (20 Hz) fits an exponential decay curve $e_u(T) \approx 1.88e^{-2.18T}$, shown in red in Fig. 4.6. This exponential improvement in precision is expected from Lemma 7, which predicts an exponential decay of error in the estimation of the hidden states, dependent on $\|e^{AT}\|_2$ from (4.16), and shown in Fig. 4.6. Thus, the impact of hidden states on the prediction precision of data-enabled inverse operator can be reduced by using sufficient time history of the desired output.

Remark 13 (Reducing hidden state dependence)**.** *In the following simulations, the time history T is chosen to be sufficiently large $T^* = 3.2$ s, which results in a normalized error $e_u \approx 0.01\%$.*

4.3.6 Need to include output time derivatives

From (4.24) in Lemma 7, even if the hidden state error is reduced by having sufficiently large time history T, (as shown in the previous subsection), current time derivatives of the output

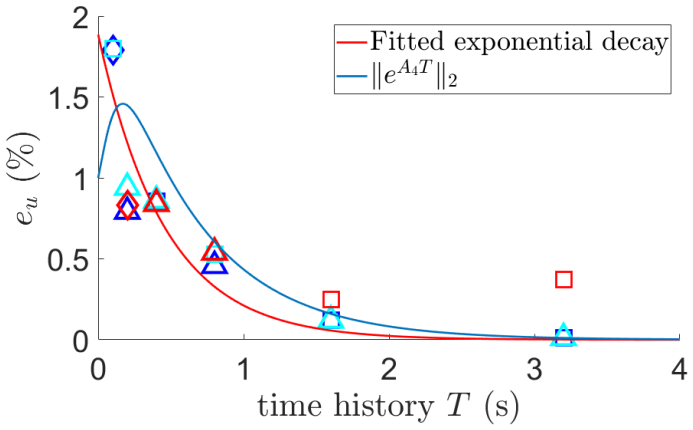

Figure 4.6: Inverse operator's precision in terms of prediction error e_u (4.35) exponentially improves with respect to different window length T of output history, for different sampling times, $\Delta t = 0.05s$(20Hz, blue), $0.1s$(10Hz, cyan), $0.2s$(5Hz, red). Similar results are seen over different N^* neurons in the hidden layer: 5 triangle ($\triangle$), 10 (square $\square$),20 (diamond $\diamond$),40 (pentagram $\star$), and 80 (circle $\bigcirc$). The fitted exponential decay (red line) is obtained with sampling time of $\Delta t = 0.05$ s (20 Hz, blue).

$\xi_d(t), y^{(r)}(t)$ are needed to achieve precision prediction with the inverse operator. Therefore, the impact of adding time-derivative information is investigated through the following two steps, for different sampling periods $\Delta t \in \{0, 05s, 0.1s, 0.2s\}$ and for different number N of neurons in the hidden layer.

1. Incrementally including higher-order time derivatives of the output when learning the inverse operator $\mathbb{G}_{d,l}^{-1}$ that predicts the inverse input $\hat{u}_d$ similar to (4.38), where output time derivatives till order l $(0 \leq l \leq 4)$ are included in the data-enabled operator learning, e.g., with $l = i \geq 0$,

$$\hat{u}_d[m] = \mathbb{G}_{d,i}^{-1}[y_d[m - m_T : 1 : m], y_d^{(i)}[m], y_d^{(i-1)}[m], \cdots y_d^{(0)}[m]], \tag{4.39}$$

where $\mathbb{G}_{d,2}^{-1} = \mathbb{G}_d^{-1}$ in (4.38).

Table 4.2: Inverse operator's precision improvement in terms of prediction error $e_{u,N}$ (4.33) and $\overline{e}_{u,N}$ (4.34) for varying output time history T and number N of neurons in the hidden layer, with sampling time $\Delta t = 0.05$ s.

T \ N	5	10	20	40	80	5	10	20	40	80
	$e_{u,N}(\%)$ as in (4.33)					$\overline{e}_{u,N}(\%)$ as in (4.34)				
0.1	1.78	2.23	1.64	2.05	2.43	3.17	3.72	4.73	5.61	6.28
0.2	0.79	0.88	0.87	0.88	0.98	1.22	1.59	1.56	1.48	1.76
0.4	0.95	0.85	0.88	0.92	0.91	1.17	1.10	1.33	1.75	1.69
0.8	0.46	0.51	0.49	0.48	0.52	0.54	0.65	0.61	0.67	1.01
1.6	0.14	0.12	0.12	0.14	0.16	0.20	0.16	0.18	0.33	0.44
3.2	0.05	0.01	0.01	0.01	0.05	0.08	0.02	0.02	0.02	0.14

2. Adding the output's time derivatives $\dot{y}_d(t), \ddot{y}_d(t)$ to NARX-type inverse operators where the inverse operator is learned using both input and output time history, i.e., to compare

$$\hat{u}_d[m] = \text{NARX}[y_d[m - m_T : 1 : m], u_d[m - m_T : 1 : m-1]], \tag{4.40}$$

$$\hat{u}_d[m] = \text{NARX}^*[y_d[m - m_T : 1 : m], \dot{y}_d[m], \ddot{y}_d[m], u_d[m - m_T : 1 : m-1]]. \tag{4.41}$$

The corresponding prediction performance, in terms of errors e_u and $\bar{e}_u$ in (4.35), for $T^* = 3.2$ s and $\Delta t = 0.05$ s are tabulated in Table 4.3, and plotted in Fig 4.7 for $T^* = 3.2$ s and different sampling time $\Delta t \in \{0.05s, 0.1s, 0.2s\}$.

4.3.6.1 Impact of including derivatives

The precision of the inverse operator depends on the inclusion of the output derivative up to order r (the relative degree). When the number of derivatives l (included in the training and evaluation) is increased from $l = 0$ to $l = 4$, the precision of the inverse operator improves significantly when all the required number ($l = 2 = r$) of time derivative features are included

Table 4.3: Prediction error $e_u, \bar{e}_u$ (4.35) for inverse operators from (4.39) to (4.41) with $\Delta t = 0.05$ s.

	$e_u(\%)$	$\bar{e}_u(\%)$		$e_u(\%)$	$\bar{e}_u(\%)$
	Noise free training data				
$\mathbb{G}_{d,0}^{-1}$	3.13	9.82	$\mathbb{G}_{d,4}^{-1}$	0.01	0.02
$\mathbb{G}_{d,1}^{-1}$	0.74	2.10	NARX	1.60	5.93
$\mathbb{G}_{d,2}^{-1} = \mathbb{G}_{d}^{-1}$	0.01	0.02	NARX*	0.01	0.02
$\mathbb{G}_{d,3}^{-1}$	0.01	0.02			
	Noisy training data				
$\mathbb{G}_{d,0}^{-1}$	53.91	114.68	$\mathbb{G}_{d,4}^{-1}$	0.41	0.78
$\mathbb{G}_{d,1}^{-1}$	11.53	37.82	NARX	3.89	17.95
$\mathbb{G}_{d,2}^{-1} = \mathbb{G}_{d}^{-1}$	0.53	1.05	NARX*	0.21	0.45
$\mathbb{G}_{d,3}^{-1}$	0.65	1.32			

in the training and evaluation data. In particular, the maximum error $\overline{e}_u$ in (4.35) reduces from 9.82% to 0.02% for the case with noise free training data and from 114.68% to 1.05% for the case with noisy training data as seen in Table 4.3. Therefore, there is substantial improvement in the inverse operator's precision (especially in the presence of noise) when time derivatives up to the required order of 2 are included.

4.3.6.2 Impact on NARX-type inverse operator

The inclusion of time derivatives is also important for NARX-type inverse operators where both input and output time history are used in the inverse operator. This can be seen by comparing NARX (4.40) without time derivatives and NARX* (4.41) with the derivatives in Table 4.3 and in Fig 4.7. When time derivatives $l = 2$ are included in the training and evaluation, the precision of the inverse operator improves significantly. In particular, the maximum error $\overline{e}_u$ in (4.35) reduces from 5.93% to 0.02% for the case with noise free training data and from 17.95% to 0.45% for the case with noisy training data as seen in Table 4.3. Therefore, there is substantial improvement in the precision of the NARX-type

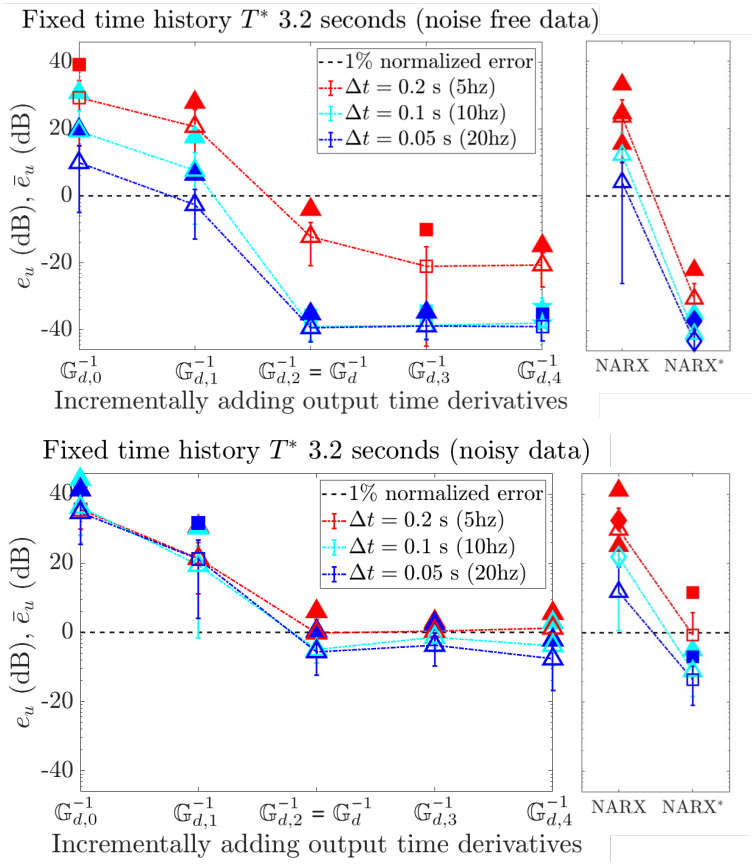

Figure 4.7: Inverse operator's precision in terms of prediction error $e_u, \overline{e}_u$ (4.35) improves for all cases with the addition of derivative information. (Top: noise free training data. Bottom: noisy training data). Similar results are seen for different number N^* of neurons in the hidden layer, with symbols as in Fig. 4.6, where the filled symbols correspond to $\overline{e}_u$ and unfilled correspond to e_u. Performance of NARX-type operator with input and output history but without derivative information is also improved with the addition of derivative information in NARX*, as in (4.40,4.41)

inverse operator when the output time derivatives up to the required order of 2 are included.

4.3.6.3 Derivative information in output time history

Conceptually, information about the derivatives up to $r-1$ (one less than the relative degree r) are available in the time history of the output and only the r^{th} time derivative $y_d^{(r)}[m]$ is directly affected by the input $u[m]$. In particular, output derivatives can be related to the output time history using finite difference techniques, especially in the noise free case, and hence direct computation of the derivatives might not appear to be critical if time history of the output is used during training. Nevertheless, including computed or measured values (even with some noise) of the time derivative $\dot{y}[m]$ (which is not directly affected by the input $u[m]$) still can improve the precision of the inverse operator as seen in Fig. 4.7 and Table 4.3. In particular, the maximum error $\overline{e}_u$ in (4.35) reduces from 9.82% to 2.10% for the case with noise free training data and from 114.68% to 37.82% for the case with noisy training data as seen in Table 4.3. Therefore, while the noise free case precision could be improved by smaller sampling time Δt without the inclusion of $\dot{y}$, for the noisy case, direct measurements of the output time derivatives can substantially improve the inverse operator training, and lead to better precision in its predictions. Moreover, the precision of the inverse operator is further improved by including time derivatives up to the required order of r (relative degree).

4.4 Chapter conclusion

This chapter showed that the Koopman-type data-enabled inverse operators can have high precision if a sufficiently large time history of the output is included to reduce the impact of hidden internal states. Additionally, measurements of the instantaneous output time derivatives (upto the relative degree) are required during training to improve the data-enabled inverse operator precision.

Chapter 5

WHAT OBSERVABLES ARE NEEDED FOR PRECISION DATA-ENABLED LEARNING OF INVERSE OPERATORS?(MC2.2)

This chapter extends the results in Chapter 4 to Multi-input-multi-output square systems (**MC2.2**), i.e., identifying the observables needded to achieve precision data-enabled inverse operators. This Chapter is based on the work accepted by the ASME journal of Dynamical systems, Measurements and Control 2024 [3]. With both SISO and MIMO simulation examples, this chapter is to show that irrespective of the selected model, removing the hidden-state dependence (which is the major challenge in **SC2**) and achieving a desired precision of inverse operators require (i) a sufficiently-long past history of the output and (ii) sufficiently-precise estimates of the output's instantaneous time derivatives that are necessary and sufficient for linear systems, and under some conditions, for nonlinear systems. This insight, about the required observables (output history and derivative) for removing the hidden-state dependence and achieving precision, is used to develop a data-enabled algorithm to learn the inverse operator for multi-input multi-output square systems. The neural nets (with universal approximation property) are used to approximate the inverse operators to show that the learned inverse operator with sufficient precision can be achieved only if the required observables are included in training.

5.1 Problem formulation

The inverse operator is developed for nonlinear multi-input-multi-output (MIMO) square systems given by

$$\frac{d}{dt}x(t) = f(x(t)) + \sum_{p=1}^{P} g_p(x(t))u_p(t) \tag{5.1}$$

$$y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_p(t) \\ \vdots \\ y_P(t) \end{bmatrix} = h(x(t)) \ = \begin{bmatrix} h_1(x(t)) \\ \vdots \\ h_p(x(t)) \\ \vdots \\ h_P(x(t)) \end{bmatrix}, \tag{5.2}$$

where states $x(t) \in \mathbb{R}^n$, input and output $u(t), y(t) \in \mathbb{R}^{P\times 1}$ with smooth $f(\cdot), g_p(\cdot)$ and $h_p(\cdot)$.

Assumption 5. *The origin of the system described in Eq.* (5.1) *is stable and has well-defined relative degree (see e.g., [95])* $\mathbf{r} = \{r_1, r_2, \ldots, r_P\}$ *and the size* n_r *of the relative degree* r *is*

$$n_r = \sum_{p=1}^{P} r_p \ \leq n. \tag{5.3}$$

5.1.1 Hidden-state dependence

The system states x can be split into state components ξ that directly depend on: (i) the output and its time derivatives

$$\xi(t) = \left[\xi_1'(t), \xi_2'(t), \ldots, \xi_P'\right]' \in \mathbb{R}^{n_r\times 1}, \tag{5.4}$$

which has n_r components, $'$ denotes the transpose,

$$\xi_p(t) \triangleq \left[y_p(t), \dot{y}_p(t), \ldots, \tfrac{d^{r_p-1}y_p(t)}{dt^{r-1}}\right]' \in \mathbb{R}^{r_p\times 1},$$

and (ii) hidden states $\eta \in \mathbb{R}^{(n-n_r)\times 1}$,

$$\left[\xi'(t) \quad \eta'(t)\right]' = S(x(t)), \tag{5.5}$$

such that in the new coordinates, Eqs. (5.1)-(5.2) can be written in the tracking form as, e.g., see [95], Lemma 4.6.1,

$$\frac{d}{dt}\xi_{p,i}(t) = \xi_{p,i+1}(t), \quad 1 \le p \le P,\ 1 \le i \le r_p - 1 \tag{5.6}$$

$$y^{(r)}(t) = L_f^r h(x(t)) + \mathbb{D}(x)u(t) \tag{5.7}$$

$$\frac{d}{dt}\eta(t) = \phi_0(\eta(t), \xi(t)) + \Phi^T(\eta(t), \xi(t))u(t), \tag{5.8}$$

where

$$y^{(r)}(t) \triangleq \left[\frac{d^{r_1}y_1}{dt^{r_1}} \quad \ldots, \frac{d^{r_P}y_P}{dt^{r_P}}\right]', \tag{5.9}$$

$$L_f^r h(x(t)) \triangleq \left[L_f^{r_1}h_1(x(t)), \quad \ldots, L_f^{r_P}h_P(x(t))\right]' \tag{5.10}$$

and the Lie derivatives are defined as $L_f^0 h(x) \triangleq h(x)$, $L_f h(x) \triangleq \frac{\partial h(x)}{\partial x} f(x)$, $L_f^k h(x) \triangleq \frac{\partial L_f^{k-1}h(x)}{\partial x} f(x)$ for all $k \ge 1$. The decoupling matrix $\mathbb{D}(x) \in \mathbb{R}^{P\times P}$ is nonsingular due to well-defined-relative degree assumption [95], and therefore, the inverse input u can be found from (5.7), as

$$\begin{aligned} u(t) &= \mathbb{D}^{-1}(x(t))[y^{(r)}(t) - L_f^r h(x(t))] \\ &= \mathbb{D}^{-1}(S^{-1}(\xi(t), \eta(t))\left[y^{(r)}(t) - L_f^r h(S^{-1}(\xi(t), \eta(t)))\right] \\ &\triangleq u_{ff}(\xi(t), y^{(r)}(t), \eta(t)) \ = u_{ff}(\mathbb{Y}(t), \eta(t)), \end{aligned} \tag{5.11}$$

which depends on (a) the observable output and its time derivatives

$$\mathbb{Y} \triangleq \left[\xi'(t) \quad (y^{(r)})'(t)\right]' \tag{5.12}$$

and (b) the hidden (not directly observed) states η.

Remark 14 (Hidden-state dependence)**.** *The inverse relationship from the observed output and its time derivatives $\mathbb{Y}$ to the input u in Eq.* (5.11)*, depends on the current state $x(t) = S^{-1}(\xi(t), \eta(t))$, which in turn depends on the hidden states $\eta(t)$. Similarly, the forward relationship from input u to the output derivative $y^{(r)}$ is also directly influenced by the states $\eta(t)$ as seen in Eq.* (5.7)*.*

Assumption 6. *The p^{th} outputs y_p is sufficiently smooth, and its time derivatives up to the relative degree r_p are bounded uniformly in time, i.e., for $0 \leq i \leq r_p$, the i^{th} order time derivative of the p^{th} output satisfies, $\left|y_p^{(i)}(t)\right| < M_{p,i} < \infty, \forall t$, and the resulting $\mathbb{Y}$ in Eq.* (5.12) *is bounded, i.e. $\|\mathbb{Y}(t)\| < M_{\mathbb{Y}}$.*

Substituting the input $u(t)$ in Eq. (5.8) with Eq. (5.11) results in the hidden-state dynamics ϕ as

$$\frac{d}{dt}\eta(t) = \phi(\eta(t), \mathbb{Y}(t)). \tag{5.13}$$

Remark 15 (Non-square systems)**.** *With the system in the tracking form as in Eqs.* (5.6)-(5.8)*, if the number of inputs is more than the number of outputs, then many choices are possible for the inverse. Therefore, additional constraints are needed to select the input. One approach is to minimize the input size by replacing the inverse $\mathbb{D}^{-1}$ in Eq.* (5.11) *with the psuedo-inverse $\mathbb{D}^{\#}$ [97]. If the number of inputs is less than the number of outputs, then general trajectories cannot be tracked and the inverse does not exist.*

Assumption 7. *The hidden-state dynamics ϕ in Eq.* (5.13) *is trajectory stable (minimum-phase) and input-to-state stable.*

Remark 16 (Bounded hidden states)**.** *The hidden-states in Eq.* (5.13) *are bounded,*

$$\|\eta(t)\| < M_\eta < \infty, \tag{5.14}$$

since the output and its derivatives are bounded from Assumption 6 and the input-to-state stability condition from Assumption 7.

From trajectory stability of the hidden-state dynamics in Eq. (5.13) (Assumption 7), the effect of initial conditions decay over time, stated formally below.

Definition 1 (Trajectory stability)**.** *For any two trajectories $\eta_1(\cdot), \eta_2(\cdot)$ driven by the same output and its time derivatives $\mathbb{Y}(\cdot)$, i.e.,*

$$\frac{d}{dt}\eta_1(t) = \phi(\eta_1(t), \mathbb{Y}(t)) \tag{5.15}$$

$$\frac{d}{dt}\eta_2(t) = \phi(\eta_2(t), \mathbb{Y}(t)) \tag{5.16}$$

satisfying the bound in Eq. (5.14) *and*

$$\|\eta_1(t_0) - \eta_2(t_0)\| < D \tag{5.17}$$

for all time t_0 *and* $T > 0$,

$$\|\eta_1(t_0 + T) - \eta_2(t_0 + T)\| < \beta_0(\|\eta_1(t_0) - \eta_2(t_0)\|, T) \tag{5.18}$$

where $\beta_0(\cdot, \cdot)$ *is class* $\mathcal{KL}(t)$.

In the above definition, a class $\mathcal{K}$ function is defined as "a continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$", as stated in [98], and a class $\mathcal{KL}(t)$ function is defined as "a continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ is said to belong to class $\mathcal{KL}(t)$ if, for each fixed s, the mapping $\beta(q, s)$ belongs to class $\mathcal{K}$ with respect to q and, for each fixed q, the mapping $\beta(q, s)$ is decreasing with respect to s and $\beta(q, s) \to 0$ as $s \to \infty$", as stated in [98].

Remark 17 (Minimum-phase system)**.** *Trajectory stability implies that the system is minimum phase since the origin* $\eta = 0$ *is stable, which follows from Definition 1 by setting the desired output and its derivatives to zero* $\mathbb{Y}(t) = 0$ *and selecting* $\eta_1(t) = 0$ *in Eq.* (5.15) *for all time* t.

5.1.2 Research problem

The goal is to minimize the error

$$\Delta u(t) \triangleq u(t) - \hat{u}(t) \tag{5.19}$$

in the prediction $\hat{u}(t)$ of the inverse input $u(t)$ in Eq. (5.11). The input prediction error $\Delta u(t)$ arises due to two causes: (i) the error $\Delta\mathbb{Y}(t) \triangleq \mathbb{Y}(t) - \tilde{\mathbb{Y}}(t)$ in measurement (or estimation) of the outputs y_p and their time derivatives $\mathbb{Y}(t)$, where $\tilde{\mathbb{Y}}(t)$ represents the measured values; and (ii) the error $\Delta\eta(t) \triangleq \eta(t) - \hat{\eta}(t)$ in estimating the hidden states $\eta(t)$, where $\hat{\eta}(t)$ is the

estimate, since from Eq. (5.11),

$$\Delta u(t) = u(t) - \hat{u}(t) \approx \frac{\partial u_{ff}}{\partial \mathbb{Y}}|_{\mathbb{Y}(t)} \Delta \mathbb{Y}(t) + \frac{\partial u_{ff}}{\partial \eta}|_{\eta(t)} \Delta \eta(t). \tag{5.20}$$

The measurement error $\|\Delta \mathbb{Y}(t)\|$ can be made small by having a sufficient number of sensors and by the use of noise filtering techniques, and is assumed to be small.

Assumption 8. *The noise in the measurement of the output and its time derivatives is bounded, i.e.,* $\|\Delta \mathbb{Y}(t)\| < N_{\mathbb{Y}} < \infty$.

The input prediction error Δu, caused by the error $\Delta \eta$ in the estimate $\hat{\eta}$ of the hidden states η in Eq. (5.20) , is addressed through the following two research problems.

(i) Quantify the time-history requirement: Quantify the error in the operator $\hat{\mathbb{H}}$ that maps the time history of the output y and its derivatives to an estimate of the hidden states η at time t

$$\hat{\eta}(t) = \hat{\mathbb{H}}[\mathbb{Y}(t - T : t)], \tag{5.21}$$

in terms of the length T of the time history.

(ii) Algorithm development: Develop an algorithm to find the inverse operator $\hat{\mathbb{G}}^{-1}$ from observables to the input using time history of the desired output and its time derivatives as in Eq. (5.21),

$$\hat{u}(t) = \hat{\mathbb{G}}^{-1}[\mathbb{Y}(t - T : t)]. \tag{5.22}$$

5.2 Quantify the time-history requirement

The hidden states $\eta(t)$ can be related to the observables through the history of the output and its time derivatives from Eq. (5.13), as

$$\eta(t) = \int_{-\infty}^{t} \phi(\eta(\tau), \mathbb{Y}(\tau)) d\tau \triangleq \mathbb{H}[\mathbb{Y}(-\infty : t)]. \tag{5.23}$$

In practice, such an operator cannot be computed from data since it requires an infinite time window and noise-free measurements of the output and its time derivatives $\mathbb{Y}$. Therefore, an estimate $\hat{\eta}$ can be obtained with an approximate operator $\hat{\mathbb{H}}$ with the noisy measurements of the output and its time derivatives $\tilde{\mathbb{Y}}$ of finite time history of length T defined as

$$\hat{\eta}(t) = \int_{t-T}^{t} \phi(\hat{\eta}(\tau), \tilde{\mathbb{Y}}(\tau))d\tau \triangleq \hat{\mathbb{H}}[\tilde{\mathbb{Y}}(t-T:t)] \tag{5.24}$$

with initial condition $\hat{\eta}(t-T) = \mathbf{0}^{(n-n_r)\times 1}$. The error ($\Delta\eta$) dynamics, in estimating the hidden states η, can be found as

$$\begin{aligned}\frac{d}{dt}\Delta\eta(t) &= \phi(\eta(t), \mathbb{Y}(t)) - \phi(\hat{\eta}(t), \tilde{\mathbb{Y}}(t)) \\ &= \phi(\eta(t), \mathbb{Y}(t)) - \phi(\eta(t) - \Delta\eta(t), \tilde{\mathbb{Y}}(t)) \\ &\triangleq \hat{\phi}_m(t, \Delta\eta(t)),\end{aligned} \tag{5.25}$$

where $\Delta\eta(t) = \eta(t) - \hat{\eta}(t)$.

5.2.1 Nonlinear MIMO case

The hidden states can be found from measured output data with a desired precision, provided a sufficiently large time history of the output and its time derivatives are available and the noise in output measurement is sufficiently low. This is shown in the next two lemmas.

Lemma 8 (Converse result)**.** *If the hidden-state dynamics ϕ in Eq.* (5.13) *is trajectory stable with respect to D, there exists a Lyapunov function $V(t, \Delta\eta) : [0, \infty) \times B_D \to R$ for the nominal (measurement-noise free) error dynamics $\hat{\phi}$, i.e., the error dynamics $\hat{\phi}_m$ in Eq.* (5.25) *in the absence of measurement noise*

$$\begin{aligned}\frac{d}{dt}\Delta\eta(t) &= \phi(\eta(t), \mathbb{Y}(t)) - \phi(\hat{\eta}(t), \mathbb{Y}(t)) \\ &= \phi(\eta(t), \mathbb{Y}(t)) - \phi(\eta(t) - \Delta\eta(t), \mathbb{Y}(t)) \\ &\triangleq \hat{\phi}(t, \Delta\eta(t)),\end{aligned} \tag{5.26}$$

that satisfies the inequalities

$$\alpha_1(\|\Delta\eta\|) \le V(t,\Delta\eta) \le \alpha_2(\|\Delta\eta\|) \tag{5.27}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial \Delta\eta}\hat{\phi}(t,\Delta\eta) \le -\alpha_3(\|\Delta\eta\|) \tag{5.28}$$

$$\left\|\frac{\partial V}{\partial \Delta\eta}\right\| \le \alpha_4(\|\Delta\eta\|) \tag{5.29}$$

where ball $B_D = \{\Delta\eta \in R^{n-r} | \|\Delta\eta\| < D\}$ *and* $\alpha_i(\cdot), i = 1,2,3,4$ *are class* $\mathcal{K}$ *functions.*

Proof. From the definition of trajectory stability in Definition 1, given two trajectories $\eta(\cdot), \hat{\eta}(\cdot)$ driven by the same inputs,

$$\frac{d}{dt}\eta(t) = \phi(\eta(t), \mathbb{Y}(t)) \tag{5.30}$$

$$\frac{d}{dt}\hat{\eta}(t) = \phi(\hat{\eta}(t), \mathbb{Y}(t)), \tag{5.31}$$

the error dynamics (resulting from subtracting Eq. (5.31) from Eq. (5.30)) is uniformly asymptotically stable since, from Eq. (5.18),

$$\|\eta(t) - \hat{\eta}(t)\| < \beta_0(\|\eta(t-T) - \hat{\eta}(t-T)\|, T) \tag{5.32}$$

or $\|\Delta\eta(t)\| < \beta_0(\|\Delta\eta(t-T)\|, T)$. Then, the existence of the Lyapunov function $V(t,\Delta\eta) : [0,\infty) \times B_D \to R$ for the nominal error dynamics $\hat{\phi}$ that satisfies Eq. (5.27) to (5.29) follows from the converse Theorem 4.16 in [98]. □

Lemma 9. *Given desired precision* $\epsilon_\eta > 0$ *in the hidden-state estimate* $\hat{\eta}(t)$ *generated by the approximate operator* $\hat{\mathbb{H}}$ *in Eq.* (5.24)*, there exists a time history length* T^**, such that the hidden state estimation error satisfies the precision requirement*

$$\|\Delta\eta(t)\| < \epsilon_\eta$$

if the following conditions are met.

1. *The hidden-state dynamics ϕ in (5.13) is trajectory stable with sufficiently large D (see Definition 1), i.e.,*

$$D > \alpha_1^{-1}(\alpha_2(M_\eta)) > M_\eta, \tag{5.33}$$

where M_η is the bound on the hidden states in Eq. (5.14) and $\alpha_i(\cdot), i = 1, 2, 3, 4$ are class $\mathcal{K}$ functions from the converse result in Lemma 8.

2. *The function $\phi(\cdot)$ in (5.13) is Lipschitz in the output and its time derivatives $\mathbb{Y}$ in region $\{\mathbb{Y}(t) \in \mathbb{R}^{n_r+P} | \ \|\mathbb{Y}(t)\| < M_{\mathbb{Y}} + N_{\mathbb{Y}}\}$, where $M_{\mathbb{Y}}$ is the bound on $\mathbb{Y}$ from Assumption 6 and $N_{\mathbb{Y}}$ is the bound on the measurement noise from Assumption 8,*

$$\|\phi(\hat{\eta}(t), \mathbb{Y}(t)) - \phi(\hat{\eta}(t), \tilde{\mathbb{Y}}(t))\| \leq L_\phi \|\mathbb{Y}(t) - \tilde{\mathbb{Y}}(t)\| \leq L_\phi N_{\mathbb{Y}} \tag{5.34}$$

where $L_\phi > 0$ is the Lipschitz constant,

3. *The measurement noise $N_{\mathbb{Y}}$ is sufficiently small to satisfy*

$$L_\phi N_{\mathbb{Y}} < \frac{\theta \alpha_3(M_\eta)}{\alpha_4(2M_\eta)}, \quad \text{with } 0 < \theta < 1, \tag{5.35}$$

$$\rho(N_{\mathbb{Y}}) \triangleq \alpha_1^{-1}\left(\alpha_2\left(\alpha_3^{-1}\left(\frac{L_\phi \alpha_4(2M_\eta)}{\theta} N_{\mathbb{Y}}\right)\right)\right) < \epsilon. \tag{5.36}$$

4. *The available time history $T > T^*$ is sufficiently large.*

Proof. The error dynamics $\Delta\eta$ in estimating the hidden states in Eq. (5.25) can be written as

$$\begin{aligned} \frac{d}{dt}\Delta\eta(t) &= \phi(\eta(t), \mathbb{Y}(t)) - \phi(\hat{\eta}(t), \mathbb{Y}(t)) + \phi(\hat{\eta}(t), \mathbb{Y}(t)) - \phi(\hat{\eta}(t), \tilde{\mathbb{Y}}(t)) \\ &= \hat{\phi}(t, \Delta\eta(t)) + p(t), \end{aligned} \tag{5.37}$$

where $\hat{\phi}$ is the nominal error dynamics defined in Eq. (5.26) and the measurement-error perturbation term p is defined

$$p(t) \triangleq \phi(\hat{\eta}(t), \mathbb{Y}(t)) - \phi(\hat{\eta}(t), \tilde{\mathbb{Y}}(t)). \tag{5.38}$$

Then, the derivative of $V(t, \Delta\eta)$, by applying Condition 1 in the current lemma and Lemma 8, along the trajectories of the perturbed system in Eq. (5.37), satisfies

$$\begin{aligned}
\dot{V}(t, \Delta\eta) &= \frac{\partial V}{\partial t} + \frac{\partial V}{\partial \Delta\eta}\hat{\phi}(t, \Delta\eta) + \frac{\partial V}{\partial \Delta\eta}p(t) \\
&\leq -\alpha_3(\|\Delta\eta\|) + \alpha_4(\|\Delta\eta\|)\|p(t)\| \text{using Eqs. (5.28),(5.29)} \\
&\leq -\alpha_3(\|\Delta\eta\|) + L_\phi N_{\mathbb{Y}}\alpha_4(\|\Delta\eta\|) \text{using Eqs. (5.34) and (5.38)} \\
&\leq -(1-\theta)\alpha_3(\|\Delta\eta\|) - \theta\alpha_3(\|\Delta\eta\|) + L_\phi N_{\mathbb{Y}}\alpha_4(2M_\eta) \\
&\quad \text{since } \|\Delta\eta(t)\| = \|\eta(t) - \hat{\eta}(t)\| \leq \|\eta(t)\| + \|\hat{\eta}(t)\| \\
&\quad\quad \text{and } \|\eta(t)\| + \|\hat{\eta}(t)\| < 2M_\eta \text{ from Eq. (5.14)} \\
&\leq -(1-\theta)\alpha_3(\|\Delta\eta\|) < 0, \quad \forall\ \|\Delta\eta\| \geq \mu
\end{aligned} \tag{5.39}$$

where

$$\mu \triangleq \alpha_3^{-1}\left(\frac{L_\phi N_{\mathbb{Y}}\alpha_4(2M_\eta)}{\theta}\right) < M_\eta \text{ from Eq. (5.35)} \tag{5.40}$$

since, $\forall\|\Delta\eta\| \geq \mu$,

$$\begin{aligned}
-\theta\alpha_3(\|\Delta\eta\|) + L_\phi N_{\mathbb{Y}}\alpha_4(2M_\eta) &\leq -\theta\alpha_3(\mu) + L_\phi N_{\mathbb{Y}}\alpha_4(2M_\eta) \\
&\leq -\theta\alpha_3\left(\alpha_3^{-1}\left(\frac{L_\phi N_{\mathbb{Y}}\alpha_4(2M_\eta)}{\theta}\right)\right) + L_\phi N_{\mathbb{Y}}\alpha_4(2M_\eta) \\
&= 0.
\end{aligned} \tag{5.41}$$

If $d = \alpha_1^{-1}(\alpha_2(M_\eta)) > 0$, then the set $B_d \triangleq \{\Delta\eta \in \mathbb{R}^{n-n_r} | \|\Delta\eta\| < d\}$ is contained in D, i.e., $B_d \subset D$ from Eq. (5.33). Note that the Lyapunov function V is decreasing provided $\|\Delta\eta\| \geq \mu$ from Eq. (5.39), where μ becomes small as the noise becomes small from Eq. (5.40).

Moreover, $\mu < \alpha_2^{-1}(\alpha_1(d)) = M_\eta$ from Eq. (5.40). Then, there exists a finite T^* which depends on (M_η, μ), e.g., see Theorem 4.18 in [98], such that the perturbed system Eq. (5.37) satisfies

$$\|\Delta\eta(t)\| \leq \rho(N_{\mathbb{Y}}) \tag{5.42}$$

for all sufficiently large time history $T > T^*$ and sufficiently bounded initial errors

$$\|\Delta\eta(t - T^*)\| = \|\eta(t - T) - \hat{\eta}(t - T)\| < M_\eta. \tag{5.43}$$

The initial error condition in Eq. (5.43) is satisfied since $\hat{\eta}(t - T) = 0$ from Eq. (4.13) and $\|\eta(t)\| < M_\eta$ from Eq. (5.14). The result follows by applying Eq. (5.36) to Eq. (5.42). □

Thus, for the nonlinear case, estimation of the hidden states η only requires sufficiently-precise time history of the output and its derivatives $\mathbb{Y}$.

5.2.2 Relaxing the time-history requirement

Conditions for relaxing the time history requirement on the output and its derivatives $\mathbb{Y}$ for precision estimation of the hidden states η are discussed in this subsection.

5.2.2.1 Avoiding highest-order derivatives (MIMO) nonlinear

If the distribution $\mathcal{G}_0 = \text{span}\{g_1, \ldots, g_P\}$ is involutive and of constant rank, e.g, see [95] Remark 4.6.3, then the state transformation $S(x(t)) = \begin{bmatrix}\xi'(t) & \eta'(t)\end{bmatrix}'$ in Eq. (5.5) can be chosen such that the system equations Eqs. (5.1)-(5.2) become Eqs. (5.6)-(5.8) with $\Phi^T = \varnothing$. Hence, the hidden-state dynamics η only depends on ξ and not $\mathbb{Y}$, by not depending on the highest-order derivatives $y^{(r)}$ in Eq. (5.9), i.e.,

$$\begin{aligned}
\frac{d}{dt}\xi_{p,i}(t) &= \xi_{p,i+1}(t), \quad 1 \leq p \leq P,\ 1 \leq i \leq r_p - 1 \\
y^{(r)}(t) &= L_f^r h(x(t)) + \mathbb{D}(x)u(t) \\
\frac{d}{dt}\eta(t) &= \phi_0(\eta(t), \xi(t))
\end{aligned} \tag{5.44}$$

Corollary 1 (Relaxed time-history requirement)**.** *The estimate $\hat{\eta}$ of the hidden state η in Eq.* (5.24) *does not depend on the highest-order derivatives $y^{(r)}$ and can be rewritten using Eq.* (5.44) *as*

$$\hat{\eta}(t) = \int_{t-T}^{t} \phi_0(\hat{\eta}(\tau), \tilde{\xi}(\tau))d\tau \triangleq \hat{\mathbb{H}}_{\xi}[\tilde{\xi}(t-T:t)]. \tag{5.45}$$

Moreover, the estimation error $\Delta\eta$ can be made arbitrarily small if: (i) hidden-state dynamics ϕ_0 in Eq. (5.44) *is trajectory stable, and (ii) a sufficiently long and precise time history of the output and its time derivatives ξ are available.*

Proof. This follows from arguments similar to those in proof of Lemma 9. □

5.2.2.2 Single-input single-output (SISO) nonlinear case

A nonlinear SISO system will satisfy the involutive conditions in Section 5.2.2.1 above, provided $g(x) \neq 0$. The dependence of the hidden-states ($\hat{\eta}$) estimation operator $\hat{\mathbb{H}}$ on the history of the output and its time derivatives $\mathbb{Y}$ in Eq. (5.24) can be relaxed to just the history of the output y for the SISO case under the conditions of the following lemma.

Lemma 10. *For the SISO case, the hidden states η at time t can be related to the history of the measured output $\tilde{y}(t) \in \mathbb{R}$ through an operator $\hat{\mathbb{H}}_y$*

$$\hat{\eta}(t) \triangleq \hat{\mathbb{H}}_y[\tilde{y}(t-T:t)]. \tag{5.46}$$

if and only if the Lie brackets below are zero

$$[\tau_i, \tau_j]_{Lie}(x) = \frac{\partial \tau_j}{\partial x}\tau_i - \frac{\partial \tau_i}{\partial x}\tau_j = 0 \quad \forall 1 \leq i, j \leq r, \tag{5.47}$$

where $r \in \mathbb{R}$ is the relative degree, vectors $\tau_i(x), \tau_j(x)$ are given by

$$\tau_i \triangleq (-1)^{i-1} ad_f^{i-1} \overline{g}(x) \tag{5.48}$$

$$\overline{g}(x) \triangleq \frac{g(x)}{L_g L_f^{r-1} h(x)}, \tag{5.49}$$

with $g(x) \in \mathbb{R}$ *and the adjoint operator* ad_f^i *given by*

$$\begin{aligned} ad_f^0 g(x) &\triangleq g(x) \\ ad_f g(x) &\triangleq [f,g]_{Lie}(x) = \frac{\partial g}{\partial x} f(x) - \frac{\partial f}{\partial x} g(x) \\ ad_f^i g(x) &\triangleq [f, ad_f^{i-1} g]_{Lie}(x). \end{aligned}$$

Proof. When the Lie bracket conditions in Eq. (5.47) are met, there exists a coordinate transformation, e.g., see [99], Proposition 9.1.1, such that the tracking form of the dynamics in Eq. (5.1) for SISO case with relative degree $r \in \mathbb{R}$ is equivalent to the dynamics of $\xi(t) \triangleq [y(t), \dot{y}(t), \ldots, y^{(r)}(t)]' \in \mathbb{R}^{r\times 1}$ in Eqs. (5.6), (5.7), and the hidden-state dynamics ϕ in Eq. (5.13) becomes just dependent on the output y (and not its time derivatives),

$$\frac{d}{dt}\eta(t) = \phi_1(\eta(t), \xi_1(t)) = \phi_1(\eta(t), y(t)). \tag{5.50}$$

Therefore, an estimate of the hidden states η can be developed, similar to Eq. (5.24), as

$$\hat{\eta}(t) \triangleq \int_{t-T}^{t} \phi_1(\hat{\eta}(\tau), \tilde{y}(\tau)) d\tau \triangleq \hat{\mathbb{H}}_y[\tilde{y}(t-T:t)]. \tag{5.51}$$

□

5.2.2.3 SISO linear case

A linear SISO system always satisfies the conditions of Lemma 10 as noted in the remark below.

Remark 18 (Linear SISO systems satisfy Lemma 10)**.** *When the SISO system is linear, i.e.,*

$$f(x) = Ax, \quad A \in R^{n\times n}, \quad g(x) = B \in R^{n\times 1} \tag{5.52}$$

$$h(x) = Cx, \quad C \in R^{1\times n} \tag{5.53}$$

the vectors τ_i *in Eq.* (5.48) *become*

$$\tau_i = (-1)^{i-1} \frac{A^{i-1}B}{CA^{r-1}B}, \quad \forall 1 \le i \le r, \tag{5.54}$$

which satisfy the Lie bracket conditions in Eq. (5.47) *of Lemma 10.*

Therefore, a coordinate transformation can always be found such that the hidden states are driven only by the history of the output, and do not depend on the history of the output's derivatives, as in Eq. (5.46). In particular, a coordinate transformation matrix S_0 can be found such that Eq. (5.5) becomes

$$\begin{bmatrix} \xi'(t) & \eta'(t) \end{bmatrix}' = \begin{bmatrix} S_0 \end{bmatrix} x(t), \tag{5.55}$$

and in the new coordinates $\xi(t) \in \mathbb{R}^{r\times 1}$, $\eta(t) \in \mathbb{R}^{(n-r)\times 1}$, Eq. (5.1) in tracking form, is

$$\dot{\xi}(t) = A_1\xi(t) + A_2\eta(t) + B_1u(t) \tag{5.56}$$

$$\dot{\eta}(t) = A_3y(t) + A_4\eta(t). \tag{5.57}$$

Note that the hidden state η is only driven by the output y. Thus, the approximate operator $\hat{\mathbb{H}}$ in Eq. (5.24) becomes

$$\hat{\eta}(t) \triangleq \int_{t-T}^{t} e^{A_4(t-\tau)}A_3\tilde{y}(\tau)d\tau \triangleq \hat{\mathbb{H}}_y[\tilde{y}(t-T:t)]. \tag{5.58}$$

The resulting error dynamics in Eq. (5.25) becomes

$$\begin{aligned} \frac{d}{dt}\Delta\eta(t) &= A_3y(t) + A_4\eta(t) - A_3\tilde{y}(t) - A_4\hat{\eta}(t) \\ &= A_4\Delta\eta(t) + A_3n_y(t) \end{aligned} \tag{5.59}$$

where $n_y(t) \triangleq y(t) - \tilde{y}(t)$ is the noise in the output measurement, which is bounded from Assumption 8

$$|n_y(t)| < N_0. \tag{5.60}$$

Note that from the trajectory stability assumption in Assumption 7, the matrix A_4 in Eq. (5.58) is Hurwitz, and there exists $\alpha > 0, \kappa > 0$ such that [96]

$$\left\|e^{A_4T}\right\|_2 \leq \kappa e^{-\alpha T}. \tag{5.61}$$

The following lemma shows that the hidden states can be found from measured output data with the desired precision, provided a sufficiently large time history of the output is available and the noise in output measurement is sufficiently small.

Lemma 11. *Given desired precision $\epsilon_\eta > 0$ in the hidden-state estimate $\hat{\eta}(t)$ generated by the approximate operator $\hat{\mathbb{H}}_y$ in Eq. (5.58), there exists a time history length T^*, such that the hidden state estimation error satisfies the precision requirement*

$$\|\Delta\eta(t)\|_2 < \epsilon_\eta \tag{5.62}$$

if

1. *the time history T is sufficiently large*

$$T > T^* = \frac{1}{\alpha} \ln \frac{M_\eta \kappa}{\epsilon_{\eta,1}}, \tag{5.63}$$

2. *and the measurement noise N_0 in Eq. (5.60) is sufficiently small*

$$N_0 < \frac{\epsilon_{\eta,2}\alpha}{\kappa \|A_3\|_2}, \tag{5.64}$$

where M_η is the bound on the hidden states in Eq. (5.14), N_0 is the bound on the output measurement noise in Eq. (5.60), α, κ define the exponential decay of the hidden dynamics in Eq. (5.61), and $\epsilon_{\eta,1} > 0,\ \epsilon_{\eta,2} > 0$ is any partition of the needed precision ϵ such that

$$\epsilon_{\eta,1} + \epsilon_{\eta,2} \le \epsilon_\eta. \tag{5.65}$$

Proof. From the error dynamics in Eq. (5.59),

$$\begin{aligned} \|\Delta\eta(t)\|_2 &= \left\| \Delta\eta(t-T)e^{A_4 T} + \int_{t-T}^{t} e^{A_4(t-\tau)} A_3 n_y(\tau) d\tau \right\|_2 \\ &\le \left\| \Delta\eta(t-T)e^{A_4 T} \right\| + \left\| \int_{t-T}^{t} e^{A_4(t-\tau)} A_3 n_y(\tau) d\tau \right\|_2 . \end{aligned} \tag{5.66}$$

Then, the first term on the right-hand side (RHS) of Eq. (5.66) is bounded by

$$\begin{aligned} \left\| \Delta\eta(t-T)e^{A_4 T} \right\| &= \left\| (\eta(t-T) - \hat{\eta}(t-T))e^{A_4 T} \right\|_2 \\ &= \left\| \eta(t-T)e^{A_4 T} \right\|_2 \text{ as } \hat{\eta}(t-T) = 0 \text{ from Eq. (5.24)} \\ &\le \|\eta(t-T)\|_2 \left\| e^{A_4 T} \right\|_2 \\ &\le M_\eta \kappa e^{-\alpha T} \text{ from Eq. (5.14) and (5.61)} \\ &\le M_\eta \kappa e^{-\alpha T^*} \text{ since } T > T^* \\ &\le \epsilon_{\eta,1} \text{ from Eq. (5.63) .} \end{aligned} \tag{5.67}$$

Similarly, the second RHS term in Eq. (5.66) is bounded by

$$\begin{aligned}\left\| \int_{t-T}^{t} e^{A_4(t-\tau)} A_3 n_y(\tau) d\tau \right\|_2 &\leq N_0 \|A_3\|_2 \int_{t-T}^{t} \left\| e^{A_4(t-\tau)} \right\|_2 d\tau \\ &\leq N_0 \|A_3\|_2 \int_{t-T}^{t} \kappa e^{-\alpha(t-\tau)} d\tau \\ &\leq N_0 \kappa \|A_3\|_2 \frac{1}{\alpha} e^{-\alpha(t-\tau)}|_{t-T}^{t} \\ &\leq N_0 \kappa \|A_3\|_2 \frac{1}{\alpha} \\ &\leq \epsilon_{\eta,2} \text{ from Eq. (5.64)} \end{aligned} \tag{5.68}$$

Using the last two equations, the error bound on the hidden-state estimate $\hat{\eta}$ in Eq. (5.66), for a sufficiently large time history $T > T^*$ (Condition 1 in lemma) and sufficiently small measurement noise bound N_0 (Condition 2 in lemma), becomes

$$\|\Delta\eta(t)\|_2 < \epsilon_{\eta,1} + \epsilon_{\eta,2} \leq \epsilon_\eta$$

from Eq. (5.65) resulting in the claim of the lemma. □

Thus, for the linear SISO case, estimation of the hidden states η only requires sufficiently-precise time history of the output as in Lemma 11.

Remark 19 (Minimal required information)**.** *The conditions of Lemmas 9 and 11 are used to quantify the needed amount of time history and measurement accuracy. In general, the hidden state can be estimated with desired accuracy using a sufficiently long time history of the output and its time derivatives* $\mathbb{Y}$ *provided the hidden dynamics is stable, i.e., the system is minimum phase. These sufficient conditions could be relaxed as long as the system can be rewritten in the output-tracking form in Eqs.*(5.6)-(5.8)*, and the hidden-state dynamics is input-to-state stable.*

Remark 20 (No hidden states)**.** *When the system's relative degree size is the number of states, i.e.,* $n_r = n$ *in Eq.* (5.3)*, then there are no hidden states* η*, and the inverse input* $u(t)$ *at time* t *in Eq.* (5.11) *only depends on the instantaneous output and its time derivatives* $\tilde{\mathbb{Y}}(t)$

$$u(t) = u_{ff}(\mathbb{Y}(t)) \triangleq \hat{\mathbb{G}}_{d,-}^{-1}[\tilde{\mathbb{Y}}(t)]. \tag{5.69}$$

5.3 *Algorithm development*

5.3.1 Data-enabled learning of inverse operator

The data-enabled inverse operator (as proposed in Eq.(5.22)) $\hat{\mathbb{G}}^{-1}$ can be expressed as

$$\begin{aligned} u(t) &= u_{ff}(\tilde{\mathbb{Y}}(t), \hat{\eta}(t)) \quad \text{with Eq. (5.11)} \\ &= u_{ff}\big(\tilde{\mathbb{Y}}(t), \hat{\mathbb{H}}[\tilde{\mathbb{Y}}(t-T:t)]\big) \quad \text{using Eq. (5.24)} \\ &\triangleq \hat{\mathbb{G}}^{-1}\big[\tilde{\mathbb{Y}}(t-T:t)\big]. \end{aligned} \tag{5.70}$$

Given a sampling period Δt, the inverse operator $\hat{\mathbb{G}}_d^{-1}$, i.e., a discretized version of $\hat{\mathbb{G}}^{-1}$ in Eq. (5.70), can be trained with sequences of the input u and measurements of the output and its derivatives $\tilde{\mathbb{Y}}$ up to order $\mathbf{r}$

$$u[n] = \hat{\mathbb{G}}_d^{-1}\big[\tilde{\mathbb{Y}}[n-n_T:n]\big], \tag{5.71}$$

where $[n]$ indicates value at time $t_n = n\Delta t$ and $n_T = T/\Delta t$, as illustrated in Fig. 5.1.

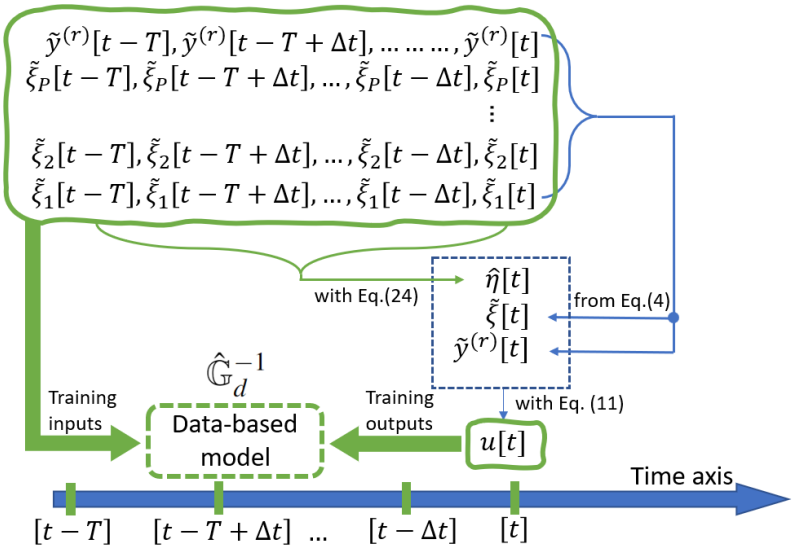

Figure 5.1: Training of the inverse operator $\hat{\mathbb{G}}_d^{-1}$ in Eq. (5.71), whose dependence on the hidden states can be removed by use of the measured output and its derivatives history $\tilde{\mathbb{Y}}$.

Remark 21 (Nonminimum-phase case)**.** *If the system is nonminimum-phase, and the hidden-state dynamics ϕ Eq.* (5.13) *can be partitioned into stable and unstable parts*

$$\frac{d}{dt}\begin{bmatrix}\hat{\eta}_s(t)\\ \hat{\eta}_u(t)\end{bmatrix} = \begin{bmatrix}\phi_s(\eta_s(t), \mathbb{Y}(t))\\ \phi_u(\eta_u(t), \mathbb{Y}(t))\end{bmatrix} \tag{5.72}$$

then the hidden-state estimate can be related to the finite past and preview of the output and its time derivatives [72], as

$$\begin{bmatrix}\hat{\eta}_s(t)\\ \hat{\eta}_u(t)\end{bmatrix} = \begin{bmatrix}\int_{t-T}^{t}\phi(\hat{\eta}_s(\tau), \tilde{\mathbb{Y}}(\tau))d\tau\\ -\int_{t+T}^{t}\phi(\hat{\eta}_u(\tau), \tilde{\mathbb{Y}}(\tau))d\tau\end{bmatrix} \tag{5.73}$$

$$\triangleq \hat{\mathbb{H}}_{pp}[\tilde{\mathbb{Y}}(t-T:t+T)]$$

and can be made sufficiently precise if past and preview information of the output and its derivatives are available. When such decoupling is not possible, iterative methods can be used to solve the hidden-state dynamics with required precision if sufficient time history is available (when the model is known), e.g., as in [70, 73]. Therefore, the bidirectional history (past T and preview T) of the output and its derivatives can be used to approximate the hidden state estimate and train the inverse operator as

$$u(t) = \hat{\mathbb{G}}_{d,pp}^{-1}\left[\tilde{\mathbb{Y}}[n-n_T:n+n_T]\right]. \tag{5.74}$$

The time-history requirements for training the inverse operator $\hat{\mathbb{G}}_d^{-1}$ for the various cases investigated here are summarized in Table 5.1.

5.3.2 Prediction with inverse operator

The inverse operator $\hat{\mathbb{G}}_d^{-1}$ in Eq. (5.71) can be used for predicting the inverse input as

$$\hat{u}[n] = \hat{\mathbb{G}}_d^{-1}\left[\mathbb{Y}[n-n_T:n]\right], \tag{5.75}$$

where the desired output and its time derivatives $\mathbb{Y}$ are noise free since they can be obtained from prescribed smooth values. Thus, the prediction error $\Delta u[n] \triangleq u[n] - \hat{u}[n]$ defined

Table 5.1: Information needed in inverse operator.

Inverse $\hat{\mathbb{G}}^{-1}$	Operator arguments	
$\hat{\mathbb{G}}_d^{-1}$	$\tilde{\mathbb{Y}}[n-n_T:n]$	Section 5.2.1
$\hat{\mathbb{G}}_{d,\xi}^{-1}$	$\tilde{\xi}[n-n_T:n], \tilde{\mathbb{Y}}[n]$	Section 5.2.2.1
$\hat{\mathbb{G}}_{d,y}^{-1}$	$\tilde{y}[n-n_T:n], \tilde{\mathbb{Y}}[n]$	Section 5.2.2.2 Lemma 10
$\hat{\mathbb{G}}_{d,-}^{-1}$	$\tilde{\mathbb{Y}}[n]$	Remark 20
$\hat{\mathbb{G}}_{d,pp}^{-1}$	$\tilde{\mathbb{Y}}[n-n_T:n+n_T]$	Remark 21

in Eq. (5.19) reflects how well the inverse operator $\hat{\mathbb{G}}_d^{-1}$, which is trained with noisy measurements as in Eq. (5.71), approximates the true map $\hat{\mathbb{G}}^{-1}$ in Eq. (5.22). Conditions for achieving a sufficiently small prediction error are established in the following lemma.

Lemma 12. *Given desired input prediction error $\epsilon_u > 0$ in the prediction $\hat{u}(t)$ generated by the inverse operator $\hat{\mathbb{G}}_d^{-1}$ in Eq. (5.71), there exists a time history length T^*, such that the inverse input prediction error $\Delta u = u(t) - \hat{u}(t)$ as in Eq. (5.19), where $u(t)$ is found via $u_{ff}(\cdot)$ in Eq. (5.11) and $\hat{u}$ is found via $u_{ff}(\cdot)$ as well but with estimated and measured values $\hat{\eta}, \tilde{\mathbb{Y}}$, satisfies the precision requirement, i.e.,*

$$\|\Delta u(t)\| = \|u_{ff}(\mathbb{Y}(t), \eta(t)) - u_{ff}(\tilde{\mathbb{Y}}(t), \hat{\eta}(t))\| < \epsilon_u$$

if

1. *$u_{ff}(\cdot)$ is Lipschitz in $\mathbb{Y}, \eta$, which are bounded under Assumption 6 and Remark 16,*

$$\|u_{ff}(\mathbb{Y}(t), \eta(t)) - u_{ff}(\tilde{\mathbb{Y}}(t), \hat{\eta}(t))\| \leq L_{\mathbb{Y}}\|\mathbb{Y}(t) - \tilde{\mathbb{Y}}(t)\| + L_\eta\|\eta(t) - \hat{\eta}(t)\|. \tag{5.76}$$

2. *Conditions in Lemma 9 are met, i.e., given $\epsilon_1/L_\eta > 0$, with sufficiently small measurement noise, there exists a time history $T^*_{\epsilon_1/L_\eta}$, such that the hidden state estimation*

error satisfies

$$\|\Delta\eta(t)\| < \epsilon_1/L_\eta. \tag{5.77}$$

where $\hat{\eta}(t)$ *is generated via* $\hat{\mathbb{H}}$ *as in Eq.* (5.24) *with* $T > T^*_{\epsilon_1/L_\eta}$.

3. *Bounds on the noise (see Assumption 8) in the output and its derivatives* $\mathbb{Y}$ *are sufficiently small*

$$\|\mathbb{Y}(t) - \tilde{\mathbb{Y}}(t)\| \leq N_{\mathbb{Y}}, \quad N_{\mathbb{Y}} < \epsilon_2/L_{\mathbb{Y}} \tag{5.78}$$

where $\epsilon_u = \epsilon_1 + \epsilon_2$ *and* $\epsilon_1 > 0,\ \epsilon_2 > 0$ *is any partition of the needed precision* ϵ_u.

Proof. The precision requirement ϵ_u follows by applying Eq. (5.77) and (5.78) to Eq. (5.76) and ensuring that the selected time history $T^* \geq T^*_{\epsilon_1/L_\eta}$. □

Remark 22 (Precision of inverse)**.** *For SISO linear systems, the bound on the inverse input prediction error* Δu *can be found from Eq.* (5.76)*, to be exponentially decreasing with the time history* T^* *(from Eq.* (5.67)*) and linearly decreasing with the measurement noises* N_0 *in Eq.* (5.68) *and* $N_{\mathbb{Y}}$ *in Eq.* (5.78) *as*

$$\|\Delta u(t)\|_2 \leq L_{\mathbb{Y}} N_{\mathbb{Y}} + L_\eta \left[\frac{\kappa \|A_3\|_2}{\alpha} N_0 + M_\eta \kappa e^{-\alpha T^*} \right]. \tag{5.79}$$

Remark 23 (Estimating bounds with Gaussian noise models)**.** *Measurement noise bounds* $(N_{\mathbb{Y}}, N_0)$ *in the analysis of the inverse-input prediction error* Δu*, e.g., as in Eq.* (5.79)*, can be estimated bounds with some specified probability using the estimated standard deviation if the noise is modeled as a Gaussian.*

Remark 24 (Nonsquare systems)**.** *Clarification that the output-to-input operator depends on the hidden-state dynamics through Eq.* (5.44) *does not require the square input-output condition, provided the system meets the involutive and constant rank conditions in Section 5.2.2.1. However, the inverse input might not be unique (for the actuator redundant case) and the learned inverse operator will reflect the input selection in the training data.*

5.3.3 Algorithm for learning the inverse operator

The analysis identifies the type of information needed for precision predictions using the observables, i.e., with the inverse operator $\hat{\mathbb{G}}^{-1}$ proposed in Eq. (5.22) and derived in Eq. (5.70) (when the model is known) and with results tabulated in Table 5.1. If such required information is not included, then even with good learning, the resulting models are not expected to have sufficient precision. Since substantial knowledge of the system might not be available, after the relative degree is found experimentally, different types of inverse operators (in Table 5.1) should be included in the model selection process. The steps for training the inverse operator are summarized in Algorithm 2.

Algorithm 2 Data-enabled learning of inverse operator

1. Determine the relative degree r of the system.

2. Prepare training data: Apply training input $u(\cdot)$ (covering the frequencies and amplitudes of interest) to the system and collect the output and its time derivatives $\mathbb{Y}(\cdot)$ for training.

3. Model evaluation criteria: Select K different smooth trajectories $\mathcal{Y}k(\cdot)$ with bounded time derivatives up to order $\mathbf{r}$ for evaluation. Select the evaluation metric e_u to assess the prediction error Δu and the desired error threshold $\underline{e}_u$.

4. Initialize a model pool for each combination of the variation of the inverse operator $\hat{\mathbb{G}}^{-1}$ in Table 5.1, sampling time Δt and the time history T. Different model candidates in the same pool can have different model types, e.g., neural net or Gaussian process, or different structures, e.g., neural nets with different structures and activation functions.

5. Train different models $\hat{\mathbb{G}}^{-1}$ and **select** the model candidate with the minimum evaluation error e_u.

6. Redo Step 5 with larger time history T. Stop if the evaluation error e_u does not decrease significantly or is below the desired threshold $\underline{e}_u$.

5.3.4 Connection to the Koopman operator

The same type of dependence on the hidden states affects both the forward operator (and potential linearized models using say the Koopman operator) [2] as well as the "Koopman-like"inverse operator investigated in the current article. This is because the dependence of the input u and output y relation on the hidden states η follows from Eq. (5.7). Therefore, the dependence on the hidden states can be managed by including the output and its time derivatives when learning the forward model. It is noted that instantaneous time-derivatives also were selected as lifted observables in [31] to better linearlize the nonlinearties. The previous sections showed that in order to achieve model precision, a sufficiently large time history of these variables are required when learning the inverse operator from data.

It is noted that the use of time history of the output (and its time derivatives) to learn the input-output relationship (forward or backward) with the hidden states is also reflected in previous works when the models are learned in the frequency domain. For example, the entire time history is used to: model the input-output relationship with complex kernels and Fourier transforms in [1], find the structure of state interactions with Z-transforms in [100], and for precision learning of the inverse with iterative methods, e.g., [70, 73]. The current work shows that the dependence on the hidden states in the time domain can be reduced by including a sufficiently large time history of the output and its time derivatives in the model.

5.3.4.1 Koopman operator for case without hidden states

Using derivatives of the observables lifted variables as in [31] can lead to a linearized forward Koopman operator when there are no hidden states η and $n_r = n$ in Eq. (5.3), as in Remark 20, and the coordinate transformation in Eq. (5.5) becomes $\xi(t) = S(x(t))$. Then, the dynamics in the output observable space $y \in \mathbb{R}^{P\times 1}$ can be linearized in the lifted derivative observable space $\xi \in \mathbb{R}^{n_r\times 1}$ defined in Eq. (5.4). Specifically, the linearized dynamics of the p^{th} output

becomes

$$\frac{d}{dt}\xi_p(t) = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \xi_p(t) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} v_p(t) \tag{5.80}$$

where the bottom row of Eq. (5.80) comes from the p^{th} row of the input-output relation in Eq. (5.8), as

$$\frac{d}{dt}\xi_{p,r_p}(t) = y_p^{(r)} = L_f^{r_p} h_p(x(t)) + \sum_{p=1}^{P} L_{g_p} L_f^{r_p-1} h(x(t)) u_p(t)$$

with $x = S^{-1}(\xi)$ and v_p defined as a lifted input [101]

$$v_p = L_f^{r_p} h_p(S^{-1}(\xi(t))) + \sum_{p=1}^{P} L_{g_p} L_f^{r_p-1} h(S^{-1}(\xi(t))) u_p(t).$$

5.3.4.2 Lifted observable impacts (on partial state output)

The lifted observables don't change the system's relative degrees r_p since, given a differentiable lifted function $z_l(\cdot)$ on the p^{th} output y_p, its r_i-th order derivative can be computed

$$\frac{d}{dt^{r_i}} z_l(y_p(t)) = \sum_{i=0}^{r_i-1} \binom{r_i - 1}{i} z_y^{(i)} y_p^{(r_i-i)} \quad z_y = \frac{\partial z}{\partial y}, \tag{5.81}$$

and the r_p^{th} output time derivative $y_p^{(r_p)}$ (that is directly related to the input at time t from Eq. (5.7)) will only appear after differentiating the lifted observable z_k for r_p times. Therefore, the relative degree for the p^{th} output remains the same, and the requirement on the p^{th} output time derivatives up to the relative degree r_p are still required as in Eq. (5.70). So the inverse operator $\hat{\mathbb{G}}_d^{-1}$ in Eq. (5.71) is still needed.

However, with the lifted observables, there are chances that the lifted dynamics satisfies the conditions in Section 5.2.2.1. Therefore, the time history of the output highest-order derivatives is not needed as in $\hat{\mathbb{G}}_{d,\xi}^{-1}$ in Table 5.1 but at the cost of an increased dimension of the observable space, i.e.,

$$u(t) = \mathbb{G}_{d,\xi}^{-1}[\xi_z(t - T : t), \mathbb{Z}(t)] \tag{5.82}$$

where $\xi_z \triangleq \left[\xi_{z_1}, \xi_{z_2}, \ldots, \xi_{z_\mathbf{l}}\right]$ and $\mathbb{Z} = (\xi_z, z^{(r)})$. $\mathbf{l}$ is the number of the lifted functions and $\xi_{z_l} \triangleq \left[z_l, \dot{z}_l, \ldots, \frac{d^{r_p-1}}{dt^{r_p-1}} z_l\right]$ where r_p is the relative degree of the p^{th} output from Eq. (5.81).

5.3.4.3 Forward operator

The forward model prediction can be written as

$$\begin{aligned}
\hat{y}(t+T_f) &= h\left(\hat{x}(t) + \int_t^{t+T_f} f(\hat{x}(t)) + g(\hat{x}(t))u(t)d\tau\right) \\
&\triangleq \mathbb{F}[\hat{x}(t), u(t:t+T_f)] \\
&= \mathbb{F}[S^{-1}(\xi(t), \eta(t)), u(t:t+T_f)] \text{ using Eq. (5.5)} \\
&= \mathbb{F}\Big[S^{-1}\big(\xi(t), \mathbb{H}[\mathbb{Y}(t-T:t)]\big), u(t:t+T_f)\Big] \text{ using Eq. (5.24)} \\
&\triangleq \hat{\mathbb{G}}[\mathbb{Y}(t-T:t), u(t:t+T_f)]. \qquad (5.83)
\end{aligned}$$

The discretized version for one-step prediction from Eq. (5.83) can be denoted as

$$\hat{y}[n+1] = \hat{\mathbb{G}}_d\big[\mathbb{Y}[n-n_T:n], u[n], u[n+1]\big]. \qquad (5.84)$$

Besides, following the process in Eq. (5.83) above but with the relaxation of the time-history requirement of the hidden-state estimate $\hat{\eta}$ in Section 5.2.2, the forward model prediction can require less arguments with cases summarized in Table 5.2.

Table 5.2: Information needed in forward operator $\hat{\mathbb{G}}_d$.

Forward	Operator arguments	
$\hat{\mathbb{G}}_d$	$\tilde{\mathbb{Y}}[n-n_T:n]$,$u[n]$	Section 5.2.1
$\hat{\mathbb{G}}_{d,\xi}$	$\tilde{\xi}[n-n_T:n]$,$u[n]$	Section 5.2.2.1
$\hat{\mathbb{G}}_{d,y}$	$\tilde{y}[n-n_T:n], \hat{\xi}[n]$,$u[n]$	Section 5.2.2.2 Lemma 10
$\hat{\mathbb{G}}_{d,-}$	$\hat{\xi}[n]$,$u[n]$	Remark 20

Remark 25. *$\hat{\mathbb{G}}_{d,pp}$ for the nonminimum-phase system does not hold since the future output and its time derivatives are required to estimate hidden states η from Remark 21.*

5.4 SISO Simulation results and discussion

5.4.1 Simulation preliminaries

5.4.1.1 Example system

The benchmark two-mass-spring-damper system [102, 103] is used for simulation studies, where the system output y is the position x_2 of the second mass m_2, and the position x_1 and velocity $\dot{x}_1$ of mass m_1 can be considered as the hidden states, as shown in Fig. 5.2.

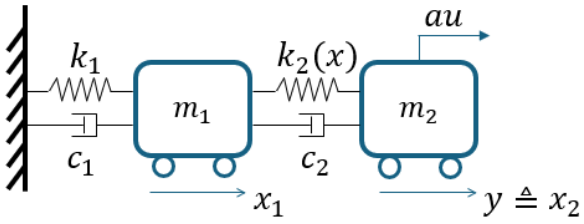

Figure 5.2: Schematic of the example SISO simulation system.

The system dynamics is

$$m_1\ddot{x}_1 = -k_2(x) - c_2(\dot{x}_1 - \dot{x}_2) - k_1x_1 - c_1\dot{x}_1 \tag{5.85}$$

$$m_2\ddot{x}_2 = +k_2(x) - c_2(\dot{x}_2 - \dot{x}_1) + au, \tag{5.86}$$

where $x \triangleq \begin{bmatrix} x_1 & \dot{x}_1 & x_2 & \dot{x}_2 \end{bmatrix}'$ and the spring force $k(x)$ is given by $k(x) = k_0(x_1 - x_2) + k_p\mathbf{p}(x_1, x_2)$ with nonlinearity $\mathbf{p}$ defined as

$$\mathbf{p}(x_1, x_2) \triangleq \begin{cases} (x_1 - x_2)^\pi & 0 \le x_1 - x_2 < 1 \\ 2 - (2 - (x_1 - x_2))^\pi & 1 \le x_1 - x_2 < 2 \\ 2 & 2 \le x_1 - x_2 \\ -p(x_2, x_1) & x_1 - x_2 < 0. \end{cases} \tag{5.87}$$

The corresponding state space model can be written as

$$\frac{d}{dt}x = f(x) + g(x)u \tag{5.88}$$

$$y = h(x), \tag{5.89}$$

where

$$f(x) = \begin{bmatrix} \dot{x}_1 \\ -\frac{1}{m_1}\left(k(x) + c(\dot{x}_1 - \dot{x}_2) + k_0 x_1 + c_0 \dot{x}_1\right) \\ \dot{x}_2 \\ +\frac{1}{m_2}(k(x) - c(\dot{x}_2 - \dot{x}_1)) \end{bmatrix}$$

$$g(x) = \begin{bmatrix} 0 & 0 & 0 & a/m_2 \end{bmatrix}', \qquad h(x) = x_2$$

with parameters selected as $m_1 = 10, m_2 = 5, k_1 = 110, c_1 = 68, a = k_1/2, k_0 = 55, c_2 = 60$ and $k_n = 80$ such that the spring deformation magnitudes are similar $\|x_1(\cdot)\|_\infty \approx \|x_1(\cdot) - x_2(\cdot)\|_\infty$, the maximum forces of the springs are similar $\|k_1 x_1(\cdot)\|_\infty \approx \|k_2(x(\cdot))\|_\infty$ and the linear and nonlinear components of the spring $k(x)$ have similar magnitudes, i.e., $\|k_0(x_1(\cdot) - x_2(\cdot))\|_\infty \approx \|k_n \mathbf{p}(x_1(\cdot), x_2(\cdot))\|_\infty$ for the training data used in this article. The relative degree of the system is $r = 2$ and the input and output are related by

$$\ddot{y}(t) = -11y(t) - 12\dot{y}(t) + 11x_1(t) + 12\dot{x}_1(t) + 11u(t) - 16\mathbf{p}(x_2, x_1). \tag{5.90}$$

5.4.1.2 Trajectory stable

Since the relative degree is $r = 2$ and output y is the displacement of mass m_2, the hidden states $\eta \in \mathbb{R}^{2\times 1}$ can be selected as the motion of the other mass m_1 as

$$\eta(t) \triangleq \begin{bmatrix} x_1(t) & \dot{x}_1(t) \end{bmatrix}' \tag{5.91}$$

with dynamics

$$\frac{d}{dt}\begin{bmatrix} \eta_1(t) \\ \eta_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -16.5 & -12.8 \end{bmatrix}\begin{bmatrix} \eta_1(t) \\ \eta_2(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 5.5 & 6 \end{bmatrix}\begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -8\mathbf{p}(\eta_1(t), y(t)) \end{bmatrix}. \tag{5.92}$$

Let $\hat{\eta}(\cdot)$ be another trajectory driven by the same output y and derivative $\dot{y}$,

$$\frac{d}{dt}\begin{bmatrix}\hat{\eta}_1(t)\\ \hat{\eta}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -16.5 & -12.8\end{bmatrix}\begin{bmatrix}\hat{\eta}_1(t)\\ \hat{\eta}_2(t)\end{bmatrix} + \begin{bmatrix}0 & 0\\ 5.5 & 6\end{bmatrix}\begin{bmatrix}y(t)\\ \dot{y}(t)\end{bmatrix} + \begin{bmatrix}0\\ -8\mathbf{p}(\hat{\eta}_1(t), y(t))\end{bmatrix}. \tag{5.93}$$

The error dynamics of $\Delta\eta(t) \triangleq [\Delta\eta_1(t) \quad \Delta\eta_2(t)]^T$ can be obtained by subtracting Eq. (5.93) from Eq. (5.92) as

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix}\Delta\eta_1(t)\\ \Delta\eta_2(t)\end{bmatrix} &= \begin{bmatrix}0 & 1\\ -16.5 & -12.8\end{bmatrix}\begin{bmatrix}\Delta\eta_1(t)\\ \Delta\eta_2(t)\end{bmatrix} - \begin{bmatrix}0\\ 8\mathbf{p}(\eta_1(t), y(t)) - 8\mathbf{p}(\hat{\eta}_1(t), y(t))\end{bmatrix} \\ &\triangleq A\Delta\eta - \mathcal{P}(\eta(t), \Delta\eta_1(t), y(t)) \\ &\triangleq \phi_h(\Delta\eta).\end{aligned} \tag{5.94}$$

It can be shown that the hidden-state dynamics in Eq. (5.94) is trajectory stable. Since A in Eq. (5.94) is Hurwitz, there exists a positive-definite matrix J such that

$$JA + A^T J = -I \tag{5.95}$$

resulting in

$$J = \begin{bmatrix}1739/1623 & 1/33\\ 1/33 & 175/4224\end{bmatrix}, \tag{5.96}$$

where $I \in \mathbb{R}^{2\times 2}$ is the identity matrix. Consider the positive definite function $V(\Delta\eta) = \Delta\eta^T J \Delta\eta$. The time derivative of the function V along the trajectories of the system Eq. (5.94) becomes

$$\begin{aligned}\dot{V}(\Delta\eta) &= \Delta\eta^T J\left(A\Delta\eta - \mathcal{P}(\cdot)\right) + \left(\Delta\eta^T A^T - \mathcal{P}^T(\cdot)\right) J\Delta\eta \quad \text{from Eq. (5.94)} \\ &= \Delta\eta^T (JA + A^T J)\Delta\eta - 2\Delta\eta^T J\mathcal{P}(\cdot) \\ &= -\Delta\eta^T \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\Delta\eta - 2\Delta\eta^T J\begin{bmatrix}0\\ 8P(t)\Delta\eta_1\end{bmatrix},\end{aligned} \tag{5.97}$$

using Eq. (5.95) and applying the Mean Value Theorem to the second row of $\mathcal{P}$ in Eq. (5.94) to obtain

$$P(t) \triangleq \frac{\partial}{\partial\eta_1}\mathbf{p}(\eta_1, y)\Big|_{\eta_P(t)\triangleq\eta_1(t)+\theta(t)\Delta\eta_1(t)} \tag{5.98}$$

for some $0 \le \theta(t) \le 1$. The expression in Eq. (5.97) can be simplified using the matrix J in Eq. (5.96) to

$$\begin{aligned}\dot{V}(\Delta\eta) &= -\Delta\eta^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Delta\eta - \Delta\eta^T \begin{bmatrix} 16P(t)\Delta\eta_1/33 \\ 175P(t)\Delta\eta_1/264 \end{bmatrix} \\ &= -\Delta\eta^T \begin{bmatrix} 1 + 16P(t)/33 & 175P(t)/528 \\ 175P(t)/528 & 1 \end{bmatrix} \Delta\eta \\ &= -\Delta\eta^T \Big[P_\eta(t) \Big] \Delta\eta.\end{aligned} \tag{5.99}$$

Note that the time derivative $\dot{V}(\Delta\eta)$ in Eq. (5.99) is negative definite as the matrix $\Big[P_\eta(t) \Big]$ is positive definite, which can be checked through pivot tests, i.e., $1 + 16P(t)/33 > 0$ and $1 + 16P(t)/33 - (175P(t)/528)^2 > 0$, with $0 \le P(t) \le \pi$ since from Eq. (5.87) and (5.98), i.e.,

$$P(t) = \begin{cases} \pi(|\eta_P(t) - y(t)|)^{\pi-1} & 0 \le |\eta_P(t) - y(t)| < 1, \\ \pi(2 - (|\eta_P(t) - y(t)|))^{\pi-1} & 1 \le |\eta_P(t) - y(t)| < 2, \\ 0 & 2 \le |\eta_P(t) - y(t)|. \end{cases}$$

Therefore, the hidden dynamics in Eq. (5.94) is trajectory stable.

Remark 26. *The analysis in Section 5.1.2 is to show that the Assumption 7 is met —— to clarify what type of data is needed for precisely modeling the inverse operator. The analysis is not used during training, which does not assume knowledge of the underlying physics.*

5.4.1.3 Data needed to train the inverse operator

Since the example system in Eq. (5.88)-(5.89) is a SISO, the inverse operator $\hat{\mathbb{G}}^{-1}_{d,\xi}$ in Table 5.1 without the highest-order derivatives is used for training based on Section 5.2.2.2. Therefore, the training requires the input u and measurements of the output and its derivatives up to the second order $(\tilde{y}, \tilde{\dot{y}}, \tilde{\ddot{y}})$, to fit

$$u[n] = \hat{\mathbb{G}}^{-1}_{d,\xi}\big[\tilde{y}[n - n_T : n], \tilde{\dot{y}}[n - n_T : n], \tilde{\ddot{y}}[n]\big]. \tag{5.100}$$

Additionally, similar to Eq. (5.75), prediction of the inverse input $\hat{u}$ requires the output and its derivatives $(y, \dot{y}, \ddot{y})$,

$$\hat{u}[n] = \hat{\mathbb{G}}_{d,\xi}^{-1}\big[y[n - n_T : n], \dot{y}[n - n_T : n], \ddot{y}[n]\big]. \tag{5.101}$$

5.4.2 Data-enabled learning of inverse operator $\hat{\mathbb{G}}_{d,\xi}^{-1}$

The inverse operator is trained using input-output data collected from simulations done via **ode45()** in MATLAB. The sampling rate of the simulation is 100 Hz, i.e., sampling period $\Delta t_s = 0.01$ s. The inverse operator $\hat{\mathbb{G}}_{d,\xi}^{-1}$ in Eq. (5.100) is trained following the steps from Algorithm 2.

5.4.2.1 Determine the relative degree

The relative degree $r \in \mathbb{R}$ can be established by applying a steep ramp input to mimic a step input — a corresponding steep rise (jump) will appear in the r^{th} time derivative of the output i.e., $y^{(r)}$ since it is directly affected by the input, while lower-order derivatives will be more smooth as illustrated in Fig. 5.3. This is similar to the identification of the relative degree by applying a Heaviside step function in [104]. Other approaches include the use of smooth triangle-shaped input as in [105] as well as the use of trained neural networks in [106].

5.4.2.2 Prepare training data

The training input signal u applied to the system needs to excite the system in the frequency range of interest with sufficient strength. The specific input u chosen for this example is constructed by concatenating 20 cycles of $p_{(f_i,\alpha_i)}(\cdot)$ $(i = 1, 2, 3, \ldots, 20)$ with different parameters specified in Eq. (4.36) and Table 4.1.

The output y, and its time derivatives $\dot{y}$ were found from simulations. The second order derivative $\ddot{y}$ was obtained from (5.90). Furthermore, to mimic noisy measurements in practice, the output and its derivatives were all corrupted with white Gaussian noise via the command **awgn()** in MATLAB with a signal-to-noise ratio of 20. The input u and noisy

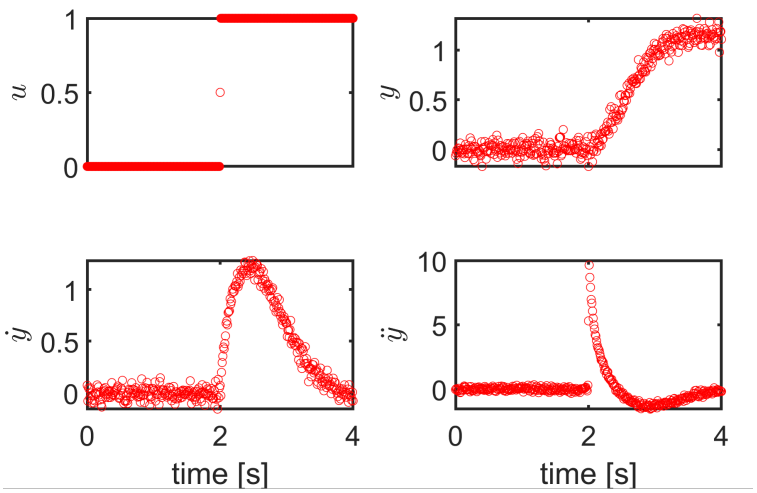

Figure 5.3: Identifying the relative degree $r = 2$ from input-output data, based on steep change (close to a discontinuity) in the acceleration $\tilde{\ddot{y}}$ for a steep ramp input u (close to a step input).

output and its time derivatives, $\tilde{y}, \tilde{\dot{y}}, \tilde{\ddot{y}}$, were used for training as in Eq. (5.100). The output and its derivatives are corrupted with Gaussian white noise, e.g., as shown in Fig. 5.3.

5.4.2.3 Model evaluation criteria

The evaluation data consists of 10 different desired trajectories $\mathcal{Y}k(t), 1 \leq k \leq 10,\ t \in [0, 10]$ with sampling period $\Delta t_s = 0.01$ s. Each desired trajectory $\mathcal{Y}k(t)$ used for evaluation needs to be sufficiently smooth to investigate the impact of different order of output's time derivatives on the inverse operator, although from Eq. (5.100) the expectation is that only output derivatives up to the r^{th} order ($r = 2$ for this example) are required. Therefore, the nominal trajectories $y_{0,k}$ (specified in Section 4.3.3) are filtered as shown in Fig. 5.4, to obtain desired

outputs $\mathcal{Y}k(t)$ and their derivatives as

$$\begin{bmatrix} \mathcal{Y}k \\ \dot{\mathcal{Y}}k \\ \ddot{\mathcal{Y}}k \\ \{\mathcal{Y}k\}^{(3)} \\ \{\mathcal{Y}k\}^{(4)} \end{bmatrix}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -a & a & 0 & 0 & 0 \\ a^2 & -2a^2 & a^2 & 0 & 0 \\ -a^3 & 3a^3 & -3a^3 & a^3 & 0 \\ a^4 & -4a^4 & 6a^4 & -4a^4 & a^4 \end{bmatrix} \begin{bmatrix} \mathcal{Y}k \\ y_{3,k} \\ y_{2,k} \\ y_{1,k} \\ y_{0,k} \end{bmatrix}(t), \tag{5.102}$$

where $a = 2\pi$ results in a cut-off frequency as 1 Hz.

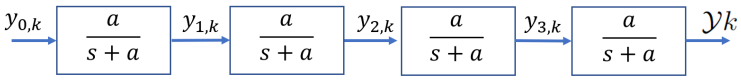

Figure 5.4: Filter to generate evaluation trajectories $\mathcal{Y}k$ differentiable up to fourth order.

The evaluation metric was selected as $e_{u,N}$, the mean of the normalized prediction error over the ten evaluation trajectories $\mathcal{Y}k(\cdot)$, i.e.,

$$e_{u,N} \triangleq \frac{1}{10}\sum_{k=1}^{10} \frac{\max_n \left|\mathcal{U}k[n] - \hat{\mathcal{U}}k[n]\right|}{\max_n \left|\mathcal{U}k[n]\right|} \times 100\% \tag{5.103}$$

for the model candidate with N hidden neurons, where the ideal inverse $\mathcal{U}k$ for each desired output $\mathcal{Y}k$ was found via the inversion-based method in [70] with exact system information and $\hat{\mathcal{U}}k$ is the predicted input for output $\mathcal{Y}k$ (computed from Eq. (5.102)) using Eq. (5.101).

5.4.2.4 Initialize the model pool

A two-layer feedforward neural-net (created through MATLAB function **feedforwardnet()**), as used for modeling and predicting in [107, 108], is used to learn the inverse operator from data with the selected 3-tuple Inverse $\hat{\mathbb{G}}^{-1}_{d,\xi}$, sampling time $\Delta t = 0.05$ s and the varying time history T. Each model pool of Eq. (5.100) consists of 5 candidates with different number $N \in \{5, 10, 20, 40, 80\}$ of neurons in the hidden layer.

5.4.2.5 Train and select model

The neural-net with N^* hidden neurons was selected from the trained candidates to quantify the prediction precision e_u of the inverse operator by minimizing the prediction error $e_{u,N}$ among the model candidates in the pool ($N \in \{5, 10, 20, 40, 80\}$ as in Section 5.4.2.4), i.e.,

$$e_u = e_{u,N^*}, \quad \bar{e}_u = \bar{e}_{u,N^*} \quad \text{where} \quad N^* = \arg\min_N e_{u,N} \tag{5.104}$$

where $\bar{e}_u$ stands for the worst case, defined as

$$\bar{e}_{u,N^*} = \max_{k=1,\dots,10} \frac{\max_n |\mathcal{U}k[n] - \hat{\mathcal{U}}k[n]|}{\max_n |\mathcal{U}k[n]|} \times 100\%.$$

5.4.2.6 Repeat with larger time history T

The evaluation metric e_u in Eq. (5.104) is shown for the selected model (two-layer neural-net with N^* hidden neurons) with different time history T in Fig. 5.5. The optimal time history T^+ was chosen as 1.6 s since the prediction precision e_u does not decrease significantly for larger time history T. Two examples of the comparison between the inverse input prediction $\hat{\mathcal{U}}k$ and the ideal inverse $\mathcal{U}k$, for $k = 1, 6$ can be seen in Fig. 5.6 (top), along with the prediction error $\Delta\mathcal{U}k$ in Fig. 5.6 (bottom).

5.4.3 Need to include output time derivatives

From Lemma 9, the hidden state error is reduced by having a sufficiently large time history T of the output and its derivatives as illustrated in Fig. 5.5. In the current subsection, the impact of adding time derivative information is investigated (with a fixed time history $T^+ = 1.6$ s) through the following two steps.

1. Incrementally including the time history T^+ of the higher-order time derivatives of the output when learning the inverse operator $\hat{\mathbb{G}}_{d,l}^{-1}$ that relates the inverse input u and the measured values of the output and its derivatives till order l ($0 \le l \le 4$), i.e.,

$$u[n] = \hat{\mathbb{G}}_{d,l}^{-1}\big[\tilde{y}[n - n_{T^+} : n], \tilde{\dot{y}}[n - n_{T^+} : n], \cdots, \tilde{y}^{(l)}[n - n_{T^+} : n]\big]. \tag{5.105}$$

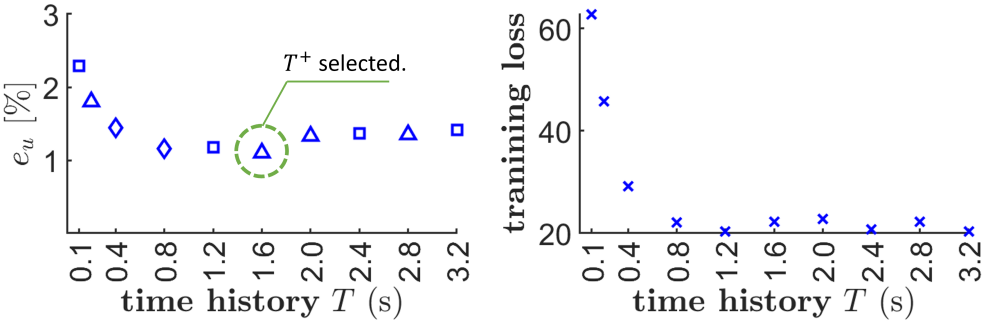

Figure 5.5: Inverse operator's precision e_u (left) in Eq. (5.104) and training loss (right) do not improve substantially beyond time history $T > 1.6$ s. The selected model with N^* hidden neurons given time history T (selected from model candidates pool as in Section 5.4.2.4) is marked in different shapes to indicate different values for N^*: 5 (triangle △), 10 (square □),20 (diamond ◇),40 (pentagram ☆), and 80 (circle ◯). Training loss is the summation of the squared error, i.e., $\sum_i \left(\Delta u[i]\right)^2 = \sum_i (u[i] - \hat{u}[i])^2$ for all training samples.

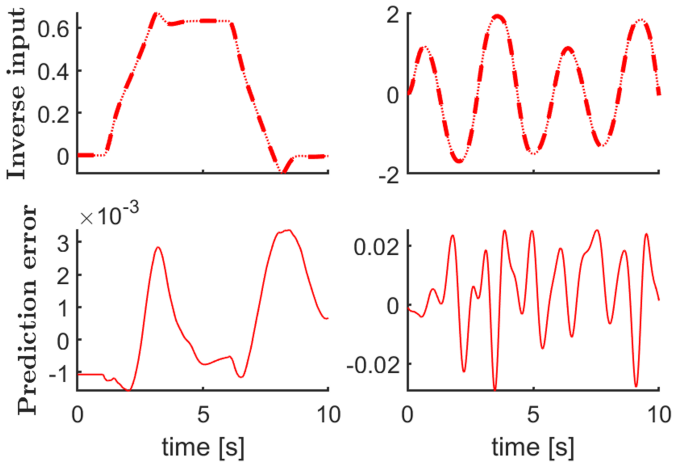

Figure 5.6: Comparison of predicted input $\hat{\mathcal{U}}k$ (dotted line) and the ideal inverse input $\mathcal{U}k$ (dashed line) for the evaluation trajectory $\mathcal{Y}k$ (top) and the prediction error $\Delta\mathcal{U}k$ (bottom). Case $k = 1$ (left) and case $k = 6$ (right).

2. Adding the instantaneous (or time history of) output derivatives $\tilde{\dot{y}}[n], \tilde{\ddot{y}}[n]$ to NARX-type inverse operators $\text{NARX}^{-1}(\cdot)$ (defined in Eq. (4.40)), where the inverse operator is learned using both input and output time history of length T^+, i.e., to compare

$$u[n] = \text{NARX}^{-1}\big[\tilde{y}[n - n_{T^+} : n], u[n - n_{T^+} : n-1]\big] \tag{5.106}$$

$$u[n] = \text{NARX}_*^{-1}\big[\tilde{y}[n - n_{T^+} : n], \tilde{\dot{y}}[n], \tilde{\ddot{y}}[n], u[n - n_{T^+} : n-1]\big], \tag{5.107}$$

$$u[n] = \text{NARX}_+^{-1}\big[\tilde{y}[n - n_{T^+} : n], \tilde{\dot{y}}[n - n_{T^+} : n], \tilde{\ddot{y}}[n], u[n - n_{T^+} : n-1]\big]. \tag{5.108}$$

For a given model from Eq. (5.105) to (5.108) and sampling periods $\Delta t \in \{0, 05s, 0.1s, 0.2s\}$, a model pool of the two-layer neural-net was created (as in Section 5.4.2.4) and the neural-net with optimal N^* hidden neurons was selected to quantify the prediction precision e_u (as in Section 5.4.2.5).

5.4.3.1 Incremental derivative feature impacts

Third and fourth order derivatives $\tilde{y}^{(3)}, \tilde{y}^{(4)}$ for training purposes were the numeric derivatives $y^{(3)}[n], y^{(4)}[n]$ computed based on Eq. (4.37) with Gaussian white noise and the acceleration $\ddot{y}$ was found from the noise-free simulation data as in Section 5.4.2.2.

Remark 27 (Higher-order derivatives)**.** *To avoid differentiating noisy signals to get estimates of the higher-order derivatives it is preferable to measure the output time derivatives directly from sensors when possible. When not directly available, higher order derivatives could be pre-estimated from the measured output and included during training. Several methods are available for such estimation, e.g., spectral methods [109], local or global smoothing [110], sliding-mode differentiator [111], neural-net-based differentiation [112], and optimization [113].*

The corresponding prediction precision e_u and $\bar{e}_u$ in Eq. (5.104) for models in Eq. (5.105) with $T^+ = 1.6$ s (selected as in Fig. 5.5) and sampling period $\Delta t \in \{0, 05s, 0.1s, 0.2s\}$, are tabulated in Table 5.3 and plotted in Fig. 5.7. Moreover, since the Lie bracket conditions in

Eq. (5.47) (Lemma 10) are met, i.e., with $r = 2$, $[\tau_1, \tau_2]_{\text{Lie}} = 0$

$$\tau_1 = \overline{g}(x) = \frac{g(x)}{L_g L_f h(x)} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}' \tag{5.109}$$

$$\tau_2 = -ad_f \overline{g}(x) = -[f, \overline{g}(x)]_{\text{Lie}}(x) = \begin{bmatrix} 0 & 6 & 1 & -12 \end{bmatrix}' \tag{5.110}$$

from system descriptions in Eq. (5.88) and (5.89). Therefore, from Table 5.1, the inverse operator $\hat{\mathbb{G}}_{d,y}^{-1}$ can be trained as

$$u[n] = \hat{\mathbb{G}}_{d,y}^{-1}\big[\tilde{y}[n - n_T : 1 : n], \tilde{\dot{y}}[n], \tilde{\ddot{y}}[n]\big], \tag{5.111}$$

and the prediction can be made as

$$\hat{u}[n] = \hat{\mathbb{G}}_{d,y}^{-1}\big[y[n - n_T : 1 : n], \dot{y}[n], \ddot{y}[n]\big], \tag{5.112}$$

with precision $e_u, \overline{e}_u$ tabulated in Table 5.3 and shown in Fig. 5.7.

Impact of including time derivative information Precision of the learned inverse operator depends on the inclusion of the history of the output time derivatives up to order $(r-1)$ (r is the relative degree) and the instantaneous r^{th} order time derivative $y^{(r)}$, as in Eq. (5.101). When the number of derivatives l (included in the training and evaluation) is increased from $l = 0$ to $l = 4$, the precision of the inverse operator $\hat{\mathbb{G}}_{d,l}^{-1}$ (with output's time derivative history up to order l) in Eq. (5.105) improves significantly when all the required number ($l = 2 = r$, e.g., $\hat{\mathbb{G}}_{d,2}^{-1}$) of time derivatives are included in the training and evaluation data, as seen in Table 5.3. For example, the maximum prediction error $\overline{e}_u$ in Eq. (5.104) improves substantially from 116.1% ($l = 0$) to 1.7% ($l = 2$) for sampling period $\Delta t = 0.05$ s. Therefore, there is substantial improvement in precision learning of the inverse operator when the time history of the output derivatives up to the required order of 2 are included, along with the history of the output.

The inclusion of the history of the higher-order output derivatives (equal or greater than the relative degree $r = 2$) is not critical to the precision of the prediction of the learned model. This is expected since the inverse input u is related to the history of the derivatives up to order $(r-1)$ via $\hat{\mathbb{G}}_{d,\xi}^{-1}$ as in Eq. (5.100). As anticipated, for sampling period $\Delta t = 0.05$ s,

the prediction precision achieved by the inverse operator $\hat{\mathbb{G}}_{d,\xi}^{-1}$ (with time derivative history velocity $\dot{y}$) in Eq. (5.101) of $e_u = 1.3\%$ is similar to the prediction precision with inverse operator $\hat{\mathbb{G}}_{d,l}^{-1}$ (with time derivative history $\dot{y}, \ddot{y}, \ldots, y^{(l)}$ till order l) in Eq. (5.105) for $2 \leq l \leq 4$ of $e_u = 0.9\%(l = 2), 0.8\%(l = 3)$, and $0.7\%(l = 4)$, as seen in Table 5.3.

The history of the output time derivatives (lower than the relative degree $r = 2$, e.g., $\dot{y}$ in this case) is not critical to the precision of the learned inverse operator. Rather only the instantaneous values of the velocity $\dot{y}$ and acceleration $\ddot{y}$ are needed since the inverse input u can be related to the history of output and only instantaneous time derivatives up to order r using $\hat{\mathbb{G}}_{d,y}^{-1}$ from Table 5.1 if the Lie bracket conditions in Eq. (5.47) are met, e.g., for the example system as verified in Eq. (5.109)-(5.110). As anticipated, the inverse operator $\hat{\mathbb{G}}_{d,y}^{-1}$ (without history of velocity $\tilde{\dot{y}}$) in Eq. (5.112) and the inverse operator $\hat{\mathbb{G}}_{d,\xi}^{-1}$ (with history of velocity $\tilde{\dot{y}}$) in Eq. (5.101) achieve similar prediction precision of $e_u = 1.3\%$ for sampling period $\Delta t = 0.05$ s, as seen in Table 5.3.

Derivative information in output time history Conceptually, information about the derivatives up to $r-1$ (one less than the relative degree r) are available in the time history of the output and only the r^{th} time derivative $y^{(r)}(t)$ is directly affected by the input $u(t)$ as in Eq. (5.7). In particular, output derivatives can be estimated from the output time history, and hence direct access to measurements of the output derivatives might not appear to be critical if the history of the output is used during training. Nevertheless, including the measured values of the time derivative $\tilde{\dot{y}}[m]$ (which are not directly affected by the input $u[m]$) still can improve the precision of the inverse operator as seen in Table 5.3 and Fig. 5.7. In particular, the comparison of inverse operators $\hat{\mathbb{G}}_{d,0}^{-1}$ (with only history of $\tilde{y}$) and $\hat{\mathbb{G}}_{d,1}^{-1}$ (with history of $\tilde{y}$ and $\tilde{\dot{y}}$) as in Eq. (5.105) shows that, with the inclusion of the directly measured values of the time derivative $\tilde{\dot{y}}$, the maximum prediction error $\overline{e}_u$ in Eq. (5.104) improves from 99.8% to 24.4% for sampling period $\Delta t = 0.2$ s and from 116.1% to 5.4% for sampling period $\Delta t = 0.05$ s as seen in Table 5.3. The improvements resulting from the direct inclusion of the time derivative $\tilde{\dot{y}}$ in the learning can be attributed to the challenge in precisely relating the history of noisy measurement $\tilde{y}$ with higher order derivatives, e.g., $\dot{y}, \ddot{y}$, which are required

to achieve precision prediction of the inverse input u as in Eq. (5.101). Therefore, there is substantial improvement in the inverse operator's precision when direct measurements of the output time derivatives are included in learning the inverse operator.

Table 5.3: Prediction precision $e_u, \bar{e}_u$ (Eq. (5.104)) for inverse operators in Eq. (5.101), Eq. (5.105) - (5.108) and Eq. (5.112) with sampling periods $\Delta t \in \{0.2 \text{ s}, 0.1 \text{ s}, 0.05 \text{ s}\}$ and time history $T^+ = 1.6$ s.

$T^+ = 1.6$ s	$\Delta t = 0.2$ s		$\Delta t = 0.1$ s		$\Delta t = 0.05$ s	
	$e_u[\%]$	$\bar{e}_u[\%]$	$e_u[\%]$	$\bar{e}_u[\%]$	$e_u[\%]$	$\bar{e}_u[\%]$
$\hat{\mathbb{G}}_{d,0}^{-1}$	51.8	99.8	47.0	95.8	57.3	116.1
$\hat{\mathbb{G}}_{d,1}^{-1}$	8.5	24.4	4.2	15.1	1.9	5.4
$\hat{\mathbb{G}}_{d,\xi}^{-1}$	1.8	3.5	1.5	4.4	1.3	2.6
$\hat{\mathbb{G}}_{d,y}^{-1}$	1.3	2.3	1.2	2.3	1.3	3.3
$\hat{\mathbb{G}}_{d,2}^{-1}$	2.0	2.8	1.3	4.3	0.9	1.7
$\hat{\mathbb{G}}_{d,3}^{-1}$	2.2	6.3	1.0	1.9	0.8	1.3
$\hat{\mathbb{G}}_{d,4}^{-1}$	1.7	2.4	1.2	2.5	0.7	2.0
NARX^{-1}	29.7	118.5	12.6	60.0	4.9	23.5
NARX_*^{-1}	2.2	6.8	1.6	3.5	0.9	1.4
NARX_+^{-1}	1.7	2.3	1.0	2.1	0.5	1.1

5.4.3.2 Impact on NARX-type inverse operator

Inclusion of output time derivatives up to order r (relative degree) is also important for NARX-type inverse operator, which uses both input and output time history. The corresponding prediction precision e_u and $\bar{e}_u$ in Eq. (5.104), for the NARX-type inverse operator $\text{NARX}^{-1}(\cdot)$, $\text{NARX}_*^{-1}(\cdot)$ (with instantaneous time derivatives up to order r, $\tilde{\dot{y}}, \tilde{\ddot{y}}$ in this example) and $\text{NARX}_+^{-1}(\cdot)$ (with time derivatives' history ($\tilde{\dot{y}}$ in this example) up to order $(r-1)$ and instantaneous $\tilde{y}^{(r)}$ ($\tilde{\ddot{y}}$ in this example)) in Eq. (5.106)-(5.108) with time history $T^+ = 1.6$ s and

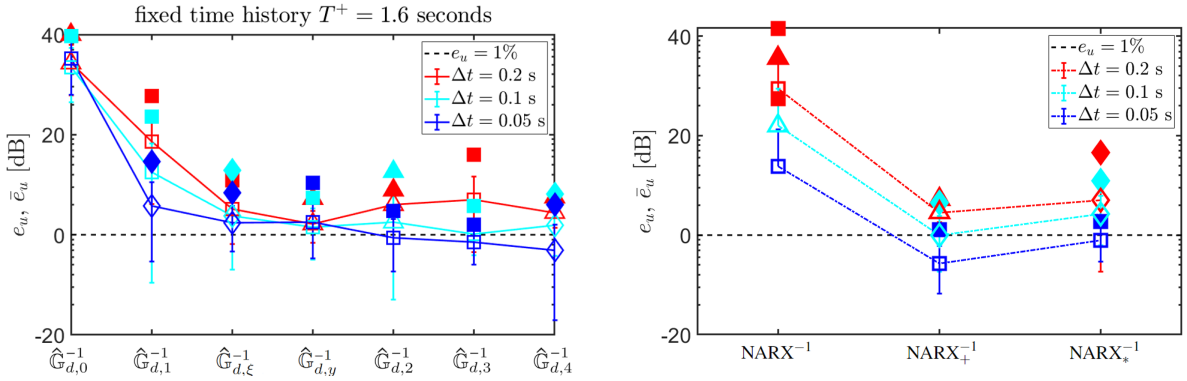

Figure 5.7: Top: Comparison of the precision $e_u, \overline{e}_u$ (in Eq. (5.104)) with different inverse operator settings $\hat{\mathbb{G}}^{-1}_{d,l}$ (in Eq. (5.105)), $\hat{\mathbb{G}}^{-1}_{d,\xi}$ (in Eq. (5.101)) and $\hat{\mathbb{G}}^{-1}_{d,y}$ (in Eq. (5.112)). Bottom: Comparison, with and without output derivative information, of the NARX-type inverse operators NARX^{-1} (in Eq. (5.106)) in terms of prediction precision $e_u, \overline{e}_u$ (in Eq. (5.104)). The selected model with N^* hidden neurons given time history T^+ and sampling period Δt (from model candidates pool in 5.4.2.4) is marked with different symbols to indicate different values for N^* (as in Fig. 5.5), where the filled symbols correspond to $\overline{e}_u$ and unfilled correspond to e_u. The bar plots correspond to the standard deviation.

sampling period $\Delta t \in \{0,05s, 0.1s, 0.2s\}$, are tabulated in Table 5.3 and plotted in Fig. 5.7. The precision of the inverse operator improves significantly when time derivatives of order $r = 2$ are included in the training and evaluation as seen by comparing the NARX-type inverse operator NARX^{-1} (without time derivatives) in Eq. (5.106) and the NARX-type inverse operator NARX$^{-1}_{+}$ (with time derivative history $\tilde{\dot{y}}$ and instantaneous $\tilde{\ddot{y}}$) in Eq. (5.108) in Table 5.3 and Fig. 5.7. In particular, with the inclusion of the output time derivatives, the maximum prediction error $\overline{e}_u$ in Eq. (5.104) improves from 118.5% to 2.3% for sampling period $\Delta t = 0.2$ s and from 23.5% to 1.1% for sampling period $\Delta t = 0.05$ s as seen in Table 5.3. From Fig. 5.7, the NARX-type inverse operator NARX$^{-1}_{*}$ (with instantaneous time derivatives $\tilde{\dot{y}}, \tilde{\ddot{y}}$) in Eq. (5.107) has the similar precision as the NARX-type inverse operator NARX$^{-1}_{+}$ (with time derivative history $\tilde{\dot{y}}$ and instantaneous $\tilde{\ddot{y}}$) in Eq. (5.108) when the

sampling rate is relatively high (Δt = 0.05 s). Therefore, there is substantial improvement in the precision of the NARX-type inverse operator when the output time derivatives up to the required order of 2 are included.

Inclusion of input history The NARX-Type inverse operators in Eq. (5.106)-(5.108) include input history in addition to output history. When no output time derivative is included in the learning of the inverse operator, the inclusion of the input history improves the precision of the learned inverse operator. This can be seen by comparing the inverse operator $\hat{\mathbb{G}}_{d,0}^{-1}$ (without input history) in Eq. (5.105) and the NARX-type inverse operator NARX^{-1} (with input history) in Eq. (5.106). The inclusion of the input history improves the prediction precision e_u from 51.8% to 29.7% for sampling period Δt = 0.2 s and from 57.3% to 4.9% for sampling period Δt = 0.05 s. However, when the output time derivatives up to order r (relative degree) are included in the learning of the inverse operator, the addition of the input history does not lead to significant improvement (less than 1%) of precision e_u. The inverse operator $\hat{\mathbb{G}}_{d,y}^{-1}$ (without input history) in Eq. (5.112) and the NARX-type inverse operator NARX_*^{-1} (with input history) in Eq. (5.107) have similar precision. The inclusion of the input history changes the prediction precision e_u from 1.3% to 2.2% for sampling period Δt = 0.2 s and from 1.3% to 0.9% for sampling period Δt = 0.05 s as seen in Table 5.3. Therefore, the inclusion of the input history is not significant for precision learning of the inverse operator when the output time derivatives up to order r (relative degree) are included.

5.5 MIMO simulation results and discussion

5.5.1 Simulation preliminaries

5.5.1.1 Example system

The MIMO system is generated by adding a mass m_3 to the end of the SISO system as in Fig. 5.8. The system outputs $y \triangleq [y_1, y_2]$ are the positions x_2, x_3 of the second and third mass m_2, m_3, respectively, and the inputs $u \triangleq [u_1, u_2]$ are the forces applied on the masses m_2, m_3, respectively, as shown in Fig. 5.8.

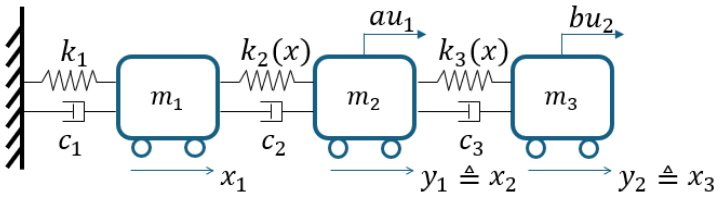

Figure 5.8: Schematic of the example MIMO simulation system.

The system dynamics of mass m_1 is the same as Eq. (5.85). The dynamics of mass m_2 and m_3 are given by

$$\begin{aligned} m_2\ddot{x}_2 &= k_2(x) + c_1(\dot{x}_1 - \dot{x}_2) - k_3(x) - c_3(\dot{x}_2 - \dot{x}_3) + au_1 \\ m_3\ddot{x}_3 &= k_3(x) - c_3(\dot{x}_3 - \dot{x}_2) + bu_2, \end{aligned}$$

where state $x \triangleq \begin{bmatrix} x_1 & \dot{x}_1 & x_2 & \dot{x}_2 & x_3 & \dot{x}_3 \end{bmatrix}'$ and the spring force $k_3(x)$ is given by $k_3(x) = k_0(x_2 - x_3) + k_n\mathbf{p}(x_2, x_3)$ with nonlinearity $\mathbf{p}$ defined in Eq. (5.87), the parameters listed in Section 5.4.1.1 and $m_3 = m_2, c_3 = c_2, b = a$. The relative degree of the system is $\mathbf{r} = \{2, 2\}$ and the input and output are related by

$$\begin{aligned} \ddot{y}_1(t) &= -22y_1(t) - 24\dot{y}_1(t) + 11x_1(t) + 12\dot{x}_1(t) + 11u_1(t) \\ &\quad + 11y_2(t) + 12\dot{y}_2(t) - 16\mathbf{p}(y_1, x_1) - 16\mathbf{p}(y_1, y_2), \\ \ddot{y}_2(t) &= -11y_2(t) - 12\dot{y}_2(t) + 11y_1(t) + 12\dot{y}_1(t) + 11u_2(t) - 16\mathbf{p}(y_2, y_1). \end{aligned}$$

The hidden states, i.e., position x_1 and velocity $\dot{x}_1$ of mass m_1, remain the same as in Section 5.4.1.2, and therefore, the hidden dynamics is trajectory stable, i.e., Assumption 7 is met.

5.5.2 Data-enabled learning of inverse operator $\hat{\mathbb{G}}_d^{-1}$

The procedures and simulation settings are the same as Section 5.4.2, which is summarized in Algorithm 2.

5.5.2.1 Determine the relative degree **r**

Similar to the method of identifying relative degree in Section 5.4.2.1, the relative degree vector **r** is determined by applying a steep ramp to each input channel sequentially and observing the first time the discontinuity happened for each output, as illustrated in Fig. 5.9.

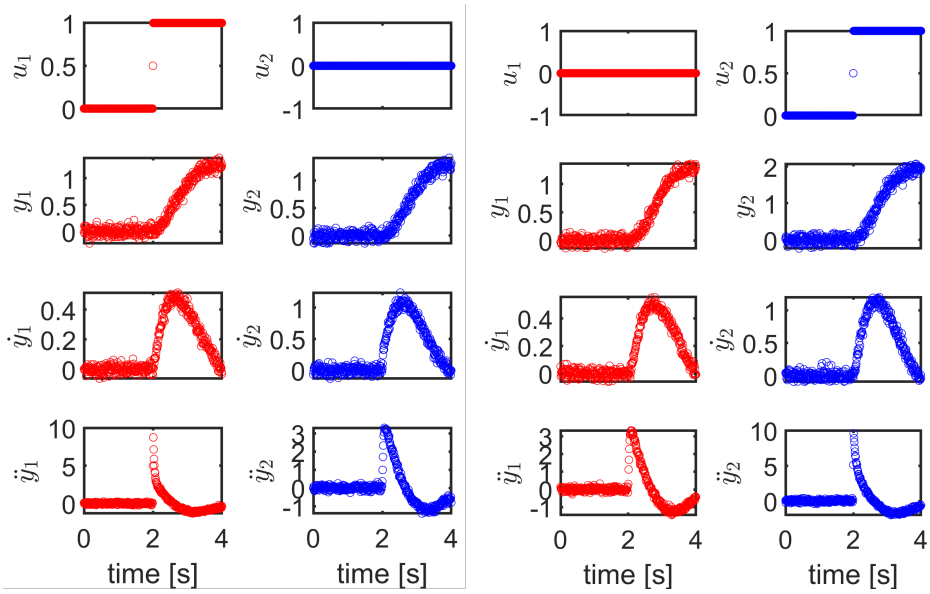

Figure 5.9: Identifying relative degree $\mathbf{r} = \{r_1, r_2\}$ by applying a steep ramp to each input channel separately. (Left two columns: a steep ramp in u_1 and no u_2) The relative degree of y_1 is $r_1 = 2$ by observing a steep change at the same time when u_1 suddenly changes. (Right two columns: a steep ramp in u_2 and no u_1) The relative degree of y_2 is $r_2 = 2$ by observing a steep change at the same time when u_2 suddenly changes.

5.5.2.2 Data needed to train the inverse operator

Since the example system in Fig. 5.8 is MIMO, the inverse operator $\hat{\mathbb{G}}_d^{-1}$ in Table 5.1 with the time history till the highest-order derivatives is used for training based on Section 5.2.1.

Therefore, the training requires the input u and measurements of the time history of the output and its derivatives up to the second order $(\tilde{y}, \tilde{\dot{y}}, \tilde{\ddot{y}})$, to fit

$$u[n] = \hat{\mathbb{G}}_d^{-1}\big[\tilde{y}_1[n-n_{T^+}:n], \tilde{\dot{y}}_1[n-n_{T^+}:n], \tilde{\ddot{y}}_1[n-n_{T^+}:n] \\ \tilde{y}_2[n-n_{T^+}:n], \tilde{\dot{y}}_2[n-n_{T^+}:n], \tilde{\ddot{y}}_2[n-n_{T^+}:n]\big] \tag{5.113}$$

Additionally, similar to Eq. (5.75), prediction of the inverse input $\hat{u}$ requires the output and its derivatives $(y, \dot{y}, \ddot{y})$ as,

$$\hat{u}[n] = \hat{\mathbb{G}}_d^{-1}\big[y_1[n-n_{T^+}:n], y_1[n-n_{T^+}:n], y_1[n-n_{T^+}:n] \\ y_2[n-n_{T^+}:n], y_2[n-n_{T^+}:n], y_2[n-n_{T^+}:n]\big] \tag{5.114}$$

The length of the time history is chosen as T^+ as in Fig 5.5 in Section 5.4.2.6 since the hidden dynamics doesn't change.

5.5.2.3 Prepare training data

The excitation input form $p_{f_i,\alpha_i}(\cdot)$ in Eq. (4.36) used in Section 5.4.2.2 is utilized here. A large number of cycles (50) are used for this MIMO case since is expected to require additional training when compared to the SISO case. The choices of the parameters (f_i, α_i) are tabulated in Table 5.4 and 5.5. The collection of the output and its derivatives with noise is conducted in the same manner as in Section 5.4.2.2.

5.5.2.4 Model evaluation criteria

The model pool is the same as Section 5.4.2.4, and the evaluation trajectories, for the output $y = [y_1, y_2]$, are designed as $\mathcal{Z}l(t) \triangleq \{0.5\mathcal{Y}k_1, 0.5\mathcal{Y}k_2\}$, $1 \le l \le 100$, reusing the evaluation trajectories $\mathcal{Y}_k(t)$, $1 \le k \le 10$ in Section 5.4.2.3, i.e., $1 \le k_1, k_2 \le 10$ and $l = 10(k_1 - 1) + k_2$. A scaling factor of 0.5 is added to adjust the magnitude of the inverse input so it is within the

Table 5.4: Parameters selected for the training input u_1 to the example system

i #	f_i	α_i	i #	f_i	α_i	i #	f_i	α_i
1	0.2	0.2	18	1	0.2	35	0	0
2	1	0.05	19	0.4	0.05	36	0.5	-0.9
3	1	0.5	20	0.2	0.05	37	0.8	-0.08
4	0.4	0.2	21	0.6	0.01	38	0.5	-0.03
5	0.6	0.1	22	0.6	0.5	39	0.3	-0.2
6	1	0.01	23	0.2	1	40	0.5	-0.7
7	0.4	0.1	24	0.6	0.2	41	0.8	-0.7
8	0.4	1	25	0.6	0.05	42	0.8	-0.9
9	0.6	1	26	0.8	-0.2	43	0.3	-0.9
10	1	0.1	27	0.3	-0.08	44	0.5	-0.2
11	0.2	0.5	28	0.3	-0.03	45	0.1	-0.2
12	0.2	0.01	29	0.8	-0.4	46	0.1	-0.7
13	0.4	0.01	30	0.5	-0.4	47	0.3	-0.7
14	0.4	0.5	31	0.1	-0.08	48	0.1	-0.03
15	0	0	32	0.1	-0.4	49	0.3	-0.4
16	0.2	0.1	33	0.5	-0.08	50	0.8	-0.03
17	1	1	34	0.1	-0.9			

Table 5.5: Parameters selected for the training input u_2 to the example system

i #	f_i	α_i	i #	f_i	α_i	i #	f_i	α_i
1	1	-0.01	18	0.2	-0.05	35	0.8	0.08
2	0.4	-0.5	19	1	-1	36	0.3	0.9
3	0.6	-0.5	20	0.2	-0.01	37	0.1	0.2
4	0.6	-0.2	21	0	0	38	0.8	0.2
5	1	-0.1	22	0.4	-0.05	39	0.3	0.08
6	0.2	-1	23	0.4	-0.01	40	0.5	0.7
7	0.6	-1	24	0.6	-0.1	41	0.3	0.7
8	0.6	-0.01	25	0.4	-1	42	0.1	0.4
9	0.4	-0.1	26	0.1	0.08	43	0.5	0.9
10	1	-0.5	27	0.3	0.2	44	0.8	0.4
11	0.4	-0.2	28	0.3	0.4	45	0.5	0.03
12	0.2	-0.2	29	0.1	0.7	46	0.5	0.08
13	0.2	-0.5	30	0.8	0.9	47	0.3	0.03
14	0.2	-0.1	31	0.8	0.7	48	0.8	0.03
15	1	-0.2	32	0.5	0.2	49	0.5	0.4
16	0.6	-0.05	33	0.1	0.03	50	0.1	0.9
17	1	-0.05	34	0	0			

range of the training data. Additionally, the evaluation metric $e_{u,N}$ in Eq. (4.33) is adapted by replacing the absolute value with the 2-norm to accommodate the vector-valued inverse input, i.e.,

$$\mathbf{e}_{u,N} \triangleq \frac{1}{100}\sum_{l=1}^{100}\frac{\max_n \left\|\mathcal{U}l[n]-\hat{\mathcal{U}}l[n]\right\|_2}{\max_n \left\|\mathcal{U}l[n]\right\|_2}\times 100\%, \tag{5.115}$$

where the ideal inverse $\mathcal{U}l$ is founded through the exact inverse with the system information, as discussed in Section 5.4.2.3.

5.5.2.5 Train and select model

Similar to Section 5.4.2.5, prediction error $\mathbf{e}_{u,N}$ in Eq. (5.115) is minimized to obtain the prediction precision $\mathbf{e}_u$, i.e.,

$$\mathbf{e}_u = \mathbf{e}_{u,N^*}, \quad \bar{\mathbf{e}}_u = \bar{\mathbf{e}}_{u,N^*} \quad \text{where} \quad N^* = \arg\min_N \mathbf{e}_{u,N}, \tag{5.116}$$

where $\bar{\mathbf{e}}_u$ stands for the worst case, defined as

$$\bar{\mathbf{e}}_{u,N^*} = \max_{l=1,\ldots,100}\frac{\max_n \left\|\mathcal{U}l[n]-\hat{\mathcal{U}}l[n]\right\|_2}{\max_n \left\|\mathcal{U}l[n]\right\|_2}\times 100\%.$$

Two example comparisons of (i) the inverse input prediction $\hat{\mathcal{U}}l$ (from the inverse operator $\hat{\mathbb{G}}_d^{-1}$ in Eq. (5.114) with sampling period $\Delta t = 0.05$ s) and (ii) the ideal inverse $\mathcal{U}l$ are shown in Fig. 5.10 (top), along with the prediction error $\Delta\mathcal{U}l$ in Fig. 5.10 (bottom).

5.5.3 Need to include output time derivatives

Two more choices of observables are used for training besides the inverse operator $\hat{\mathbb{G}}_d^{-1}$ in Eq. (5.117) to illustrate the need to include the output time derivatives in the observables, as predicted by the current article.

1. Inclusion of only the time history of the output, $\hat{\mathbb{G}}_{d,0}^{-1}$ as in Eq. (5.105). The MIMO extension is

$$u[n] = \hat{\mathbb{G}}_{d,0}^{-1}\big[\tilde{y}_1[n-n_{T^+}:n], \tilde{y}_2[n-n_{T^+}:n]\big]. \tag{5.117}$$

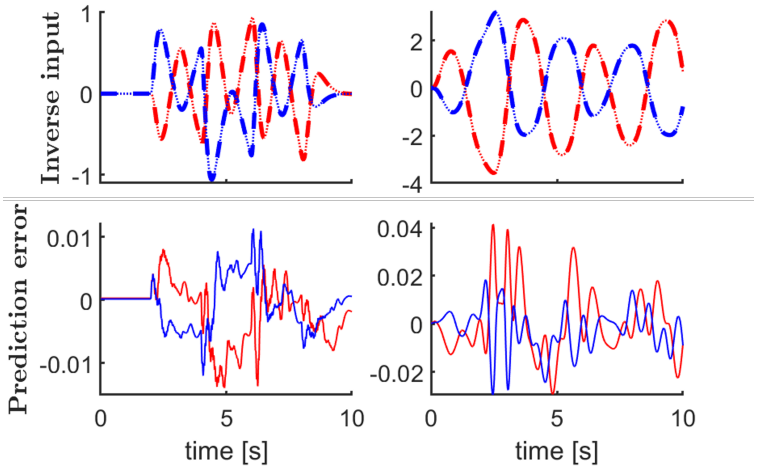

Figure 5.10: Comparison of predicted input $\hat{\mathcal{U}}l$ (red and blue dotted lines) and the ideal inverse input $\mathcal{U}l$ (red and blue dashed lines) for the evaluation trajectory $\mathcal{Z}l$ (top) and the prediction error $\Delta\mathcal{U}l$ (bottom). Case $l = 13$ (left) and case $l = 54$ (right).

2. NARX-type inclusion of the time history of both the output and the input, i.e., the MIMO extension of Eq. (4.40)

$$u[n] = \text{NARX}^{-1}\big[\tilde{y}_1[n - n_{T^+} : n], \tilde{y}_2[n - n_{T^+} : n], u_1[n - n_{T^+} : n-1], u_2[n - n_{T^+} : n-1]\big]. \tag{5.118}$$

Table 5.6: Prediction precision $\mathbf{e}_u, \bar{\mathbf{e}}_u$ (Eq. (5.116)) for inverse operators in Eq. (5.114), Eq. (5.117) and (5.118) with sampling periods $\Delta t \in \{0.2\text{ s}, 0.1\text{ s}, 0.05\text{ s}\}$ and time history $T^+ = 1.6$ s.

$T^+ = 1.6$ s	$\Delta t = 0.2$ s		$\Delta t = 0.1$ s		$\Delta t = 0.05$ s	
	$\mathbf{e}_u$[%]	$\bar{\mathbf{e}}_u$[%]	$\mathbf{e}_u$[%]	$\bar{\mathbf{e}}_u$[%]	$\mathbf{e}_u$[%]	$\bar{\mathbf{e}}_u$[%]
$\hat{\mathbb{G}}_{d,0}^{-1}$	25.5	79.1	27.7	111.6	19.2	149.2
$\hat{\mathbb{G}}_{d}^{-1}$	1.9	5.9	1.8	5.9	1.2	3.7
NARX^{-1}	16.2	109.6	9.0	60.1	5.7	30.3

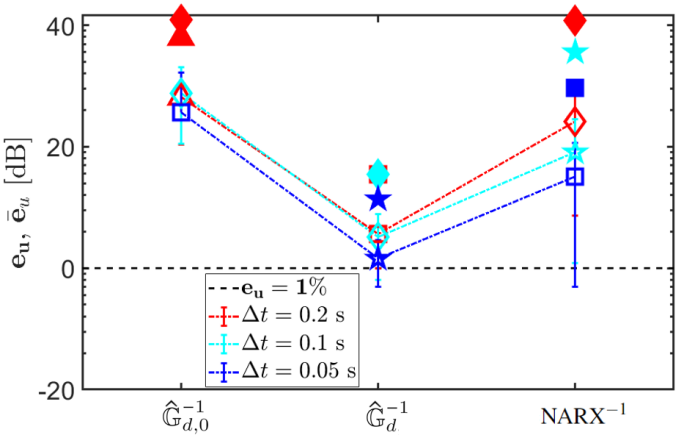

Figure 5.11: Comparison of the precision $\mathbf{e}_u, \overline{\mathbf{e}}_u$ (in Eq. (5.116)) with different inverse operator settings $\hat{\mathbb{G}}_{d,0}^{-1}$ (in Eq. (5.117)), $\hat{\mathbb{G}}_{d}^{-1}$ (in Eq. (5.114)) and NARX^{-1} (in Eq. (5.118)). The selected model with N^* hidden neurons given time history T^+ and sampling period Δt (from model candidates pool) is marked with different symbols to indicate different values for N^* (as in Fig. 5.5), where the filled symbols correspond to $\overline{\mathbf{e}}_u$ and unfilled correspond to $\mathbf{e}_u$. The bar plots correspond to the standard deviation.

Impact of including time history of time derivatives Similar to the discussion in Section 5.4.3, adding the input time history (to the output time history) can improve the precision of the learned inverse operator, but precision is substantially improved when including the required observables identified in this article. In particular, from Table 5.6 for sampling period Δt = 0.05 s, inclusion of the input history as in the NARX-type inverse operator NARX^{-1} in Eq. (5.118) improves the prediction precision $\mathbf{e}_u$ from 19.2% to 5.7% and the maximum prediction error $\overline{\mathbf{e}}_u$ reduces from 149.2% to 30.3%. However, substantial improvement in prediction precision $\mathbf{e}_u$, from 19.2% to 1.2% and reduction in maximum prediction error $\overline{\mathbf{e}}_u$ from 149.2% to 3.7%, is achieved with the inclusion of the history of the output time derivatives in the inverse operator $\hat{\mathbb{G}}_{d}^{-1}$ from Eq. (5.113). Therefore, inclusion of the output and its time derivatives in the observables, as predicted by the analysis, leads to increased precision in data-enabled inverse operators for the MIMO example system.

5.6 Chapter conclusion

This chapter investigated the type of data (observables) needed to learn inverse operators (that predict the input needed to track a desired output) with a desired precision. Analysis was presented to show that the data-enabled inverse operators can have high precision for linear systems, and under some conditions, for nonlinear systems, if a sufficiently-long time history of the output and sufficiently-precise estimates of the output's instantaneous time derivatives up to the relative degree are available. An algorithm based on these minimal amount of data was illustrated with simulation examples. In particular, results showed that the inclusion of the output and its time derivatives as observables, based on predictions by the analysis, leads to increased precision in data-enabled inverse operators for both SISO and MIMO examples.

Chapter 6

OUTPUT-SAMPLED MODEL PREDICTIVE PATH INTEGRAL CONTROL (O-MPPI) FOR INCREASED EFFICIENCY (MC3)

This chapter introduces the third Scenario **SC3**, trajectory regulation with dynamic obstacles in real-time. Sampling-based approaches such as model predictive path integral control (MPPI) [75] have become popular methods to solve optimization problems due to fast computations possible with graphic processing units and parallelized computing. However, in general, it is challenging to appropriately select a desirable input mean to meet constraints in the output space, especially in dynamic environments. Then, the proposed output-sampling-based MPPI (o-MPPI) approach (**MC3**) uses the inverse model $G^{-1}(\cdot)$ to map outputs to inputs rather than the traditional approach in MPPI of using the input-to-output forward model $f(\cdot)$ as illustrated in Fig. 6.1(top). The inputs obtained using the inverse model are then weighted to obtain the optimized control sequence, as in standard MPPI. This chapter is based on the work accepted by the International Conference on Robotics and Automation (ICRA) 2024 [4], and the supporting video can be found at https://youtu.be/snhlZj3l5CE.

6.1 Proposed framework

The proposed output-sampling-based MPPI (o-MPPI) approach is summarized as in Algorithm 3. Essentially, there are four steps: (i) selection of the cost function; (ii) sampling in the output space; (iii) inversion to find the associated inputs; and (iv) weighted selection of the input.

As in standard MPPI, a cost function is used to specify the desirability of a specific

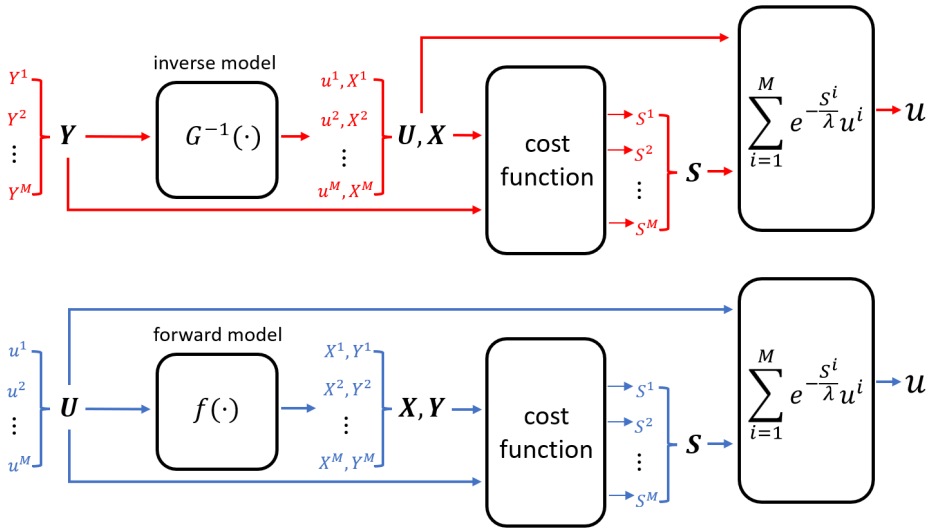

Figure 6.1: Comparison between the proposed output-MPPI (o-MPPI) in red (top) and the standard MPPI in blue (bottom). There are M rollouts and $f(\cdot)$ maps inputs to states and outputs in the standard MPPI and the inverse $G^{-1}(\cdot)$ is the inverse dynamics that maps output trajectories to inputs and the states in the proposed o-MPPI. S^m is the cost for the m^{th} rollout output-state-input (Y^m, X^m, u^m). $\lambda \in R^+$ is the temperature parameter used in the weighting to obtain the optimized input u.

input-state-output rollout. In particular, consider the optimization

$$\min_u J(u) = \phi(X_{N-1}) + \sum_{k=0}^{N-1} q(X_k) + \frac{1}{2} u_k^T R u_k \tag{6.1}$$

subject to system dynamics

$$X_{k+1} = f(X_k, u_k), Y_k = h(X_k), \tag{6.2}$$

where f represents the forward dynamics, h maps the state X_k to the output Y_k at time step k, and the output trajectory is given by $\mathbf{Y} \triangleq \begin{bmatrix} h(X_0) & h(X_1) & \dots & h(X_{N-1}) \end{bmatrix}$. In the above optimization, $q(\cdot)$ is the running cost, R is the weight matrix of the input energy cost, and the terminal cost is $\phi(\cdot)$ in Eq. (6.1). Additionally, there are constraints on the output trajectory to lie in an acceptable region, i.e., $\mathbf{Y} \in \mathcal{Y}$.

Algorithm 3 o-MPPI (red is the difference from standard MPPI)

Given: Number of rollouts & time steps M, N; Cost function; Temperature parameter λ; Inverse model G^{-1};
while Task is not done **do**
  $X_k \leftarrow$ state estimate
  for $m \rightarrow 0$ to $M-1$ in parallel **do**
    $Y_0^m \leftarrow h(X_k)$, $S^m \leftarrow 0$
    [**Sampling**] Sample the m^{th} trajectory rollout Y^m
    [**Inverse**] Compute the corresponding inverse input u^m and states X^m from the inverse model G^{-1}
    Calculate the trajectory cost S^m based on the cost function and the m^{th} rollout (u^m, X^m, Y^m)
  end for
  [**Weighting**] For $m = 1, 2, \ldots, M$, compute the normalized weights $\{w_m\}$ based on the trajectory costs $\{S^m\}$ and the selected temperature parameter λ
  Obtain the weighted average $u = \sum_{m=0}^{M} w_m u^m$
  Apply the first entry of u to the system
  $k \leftarrow k+1$
end while

The output sampling can be generated from any trajectory planning method, e.g., parametric ones like spline or bezier curves by specifying the waypoints as in Fig. 6.2, using optimization methods such as k^{th}-order constrained path optimization (KOMO) [114], via-point-based Stochastic Trajectory Optimization (VP-STO) [115], or from a data-based planner [116, 117, 118, 119]. For each sampled output, inverse maps are used to find the corresponding input that yields tracking of the output.

The optimized input is obtained as the weighted sum of all the input rollouts as in

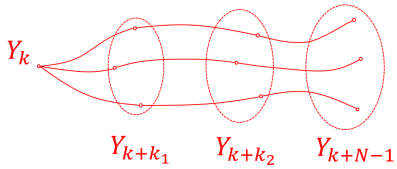

Figure 6.2: A generic way of sampling trajectory rollouts based on fitted smooth curves specified by waypoints that are selected from the regions of interest (dashed ellipsoids).

standard MPPI, i.e.,

$$\sum_{m=0}^{M} w_m u^m = u_{\text{mean}} + \sum_{m=0}^{M} w_m \epsilon^m, \tag{6.3}$$

where $u^m = u_{\text{mean}} + \epsilon^m$ and $\{w_m\}$ for $m = 1, 2, \ldots, M$ are normalized weights, i.e., $\sum_{m=1}^{M} w_m = 1$.

6.2 Application of o-MPPI to experimental setup

The system for evaluating the proposed o-MPPI is set up to mimic a dynamic autonomous driving scenario with moving obstacles.

6.2.1 System description

In the experiment, the goal is for a fast bot (green) to move as close as possible to its desired speed of 20 cm/s while maneuvering around slower bots that can be considered as moving obstacles at the same time. The ability for the fast bot to maintain its desired speed and overtake the slower bots, as illustrated in Fig. 6.3, are used to quantitatively compare the performance of the standard MPPI and the proposed o-MPPI. An example simulation run is shown in the left plot in Fig. 6.3. The slower bots are moving at constant speeds of 10 cm/s (blue) and 12 cm/s (red). All bots are driving in the counter-clockwise direction in ellipsoidal tracks with dimensions tabulated in Table 6.1 and illustrated in Fig. 6.3, similar to the shape of the track in [120].

Table 6.1: Turtlebot, and specifications of track shown in Fig. 6.3.

bot turning circle radius (cm)	10.5	lane width lw (cm)	30
max vel. $\bar{v}$ (cm/s)	22	straight lines sl (cm)	150
max angular vel. $\bar{\omega}$ (rad/s)	2.8	inner radius ir (cm)	40
collision area width cw (cm)	30	collision area length cl (cm)	63

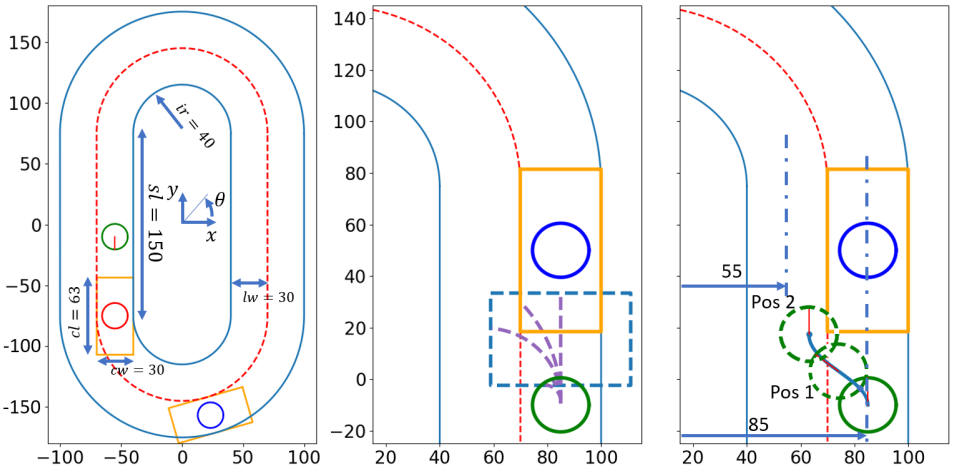

Figure 6.3: The left plot is the schematic drawing of the track. The middle plot illustrates an example region of interest (dashed blue rectangle) from which the output waypoint is sampled. The right plot shows an example output rollout where the solid circles indicate the current position and dashed circles indicate future positions in the rollout. The bright yellow rectangles indicate the collision regions.

6.2.2 Standard MPPI

6.2.2.1 Forward model F

The bot (TurtleBot3 Burger) state X at time step k is defined as $X_k \triangleq \begin{bmatrix} x_k & y_k & \theta_k & v_k & \omega_k \end{bmatrix}$. (x_k, y_k) is the position pair and θ_k is the orientation with respect to the Turtle Track coordinates, as in Fig. 6.3. v_k is the forward velocity and ω_k is the angular velocity. The input

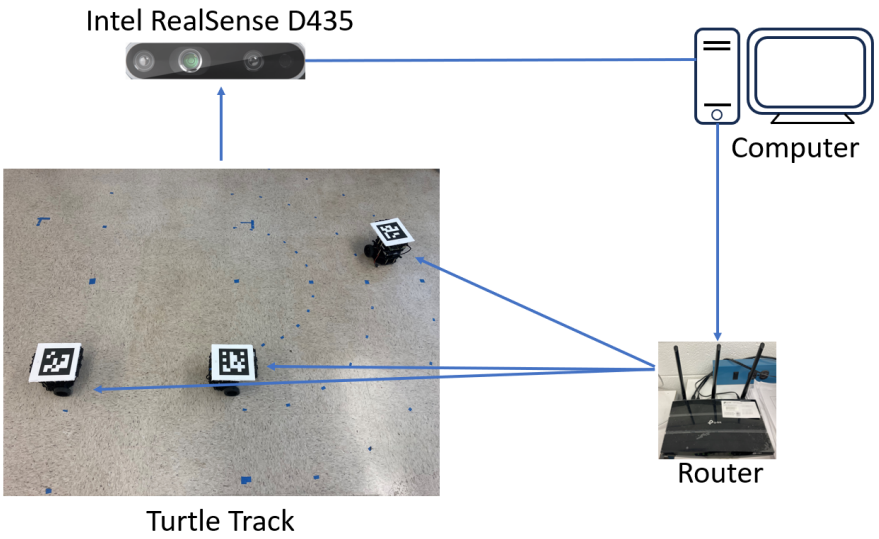

Figure 6.4: The positions of the bots (TurtleBot3 Burger) are estimated from images taken with an Intel RealSense D435 camera. The control algorithms are run on a computer and the control commands are sent to the robots via a wireless router. In the Turtle Track photo, the blue tapes are used to indicate the track boundaries.

u to the system is $u_k \triangleq \begin{bmatrix} v_k^{\text{des}} & \omega_k^{\text{des}} \end{bmatrix}$ where v_k^{des} is the desired forward velocity and ω_k^{des} is the desired angular velocity, and the system output is $Y_k \triangleq \begin{bmatrix} x_k & y_k \end{bmatrix}$.

The forward bot dynamics $\dot{X} = f(X, u)$ is approximated by the forward Euler discretization $X_{k+1} = \mathbf{F}(X_k, u_k)$,

$$x_{k+1} = x_k + v_k \cos(\theta_k) * \Delta t \tag{6.4}$$

$$y_{k+1} = y_k + v_k \sin(\theta_k) * \Delta t \tag{6.5}$$

$$\theta_{k+1} = \theta_k + \omega_k * \Delta t \tag{6.6}$$

$$v_{k+1} = \text{sat}_{\bar{v}}(v_k + \alpha(v_k^{\text{des}} - v_k) * \Delta t) \tag{6.7}$$

$$\omega_{k+1} = \text{sat}_{\bar{\omega}}(\omega_k + \alpha(\omega_k^{\text{des}} - \omega_k) * \Delta t), \tag{6.8}$$

where the sampling time is $\Delta t = 0.04$ s, $\text{sat}_\beta(x) = \beta \text{sign}(x)$ if $\text{abs}(x) > \beta$ otherwise $\text{sat}_\beta(x) = x$ is a saturation function used to bound with the magnitudes of the speed and angular velocity

with $\bar{v}$ and $\bar{\omega}$ defined in Table 6.1, and the settling-time parameter $\alpha = 4/0.35$ is estimated from the measured step response.

6.2.2.2 Cost function selection

The cost function J in Eq. (6.1) is designed to have only the running cost function $q(\cdot)$, with zero input energy cost (R=0), similar to the cost functions used in autonomous driving in [120, 121]. The running cost function is designed to have three components, i.e., $q(\cdot) \triangleq c_l(\cdot) + c_v(\cdot) + c_c(\cdot)$. The first cost $c_l(\cdot)$, designed for lane keeping, is the product of the squared values of the bot's distance to the center lines of the two lanes, as shown in the right plot in Fig. 6.3. Formally,

$$c_l(X_k) = w_0(r_k - 55)^2(r_k - 85)^2 + w_1\gamma_{OT}(x_k, y_k), \tag{6.9}$$

where $w_0 = 0.001, w_1 = 600$, $\gamma_{OT}(x_k, y_k) = 1$ if (x_k, y_k) is outside of the designed track shown in Fig. 6.3 and zero otherwise, $r_k = \sqrt{x_k^2 + z_k^2}$, and $z_k = 0$ if $\text{abs}(y_k) < 75.0$ and $z_k = y_k - 75.0 * \text{sign}(y_k)$ otherwise. The second cost $c_v(\cdot)$, designed for driving the bot at a speed close to the desired speed of 20 cm/s, is the squared of the speed difference between the current speed v_k and the desired one,

$$c_v(X_k) = w_1(v_k - 20)^2, \qquad w_1 = 0.4. \tag{6.10}$$

The third cost $c_c(\cdot)$, designed to avoid collisions with the slower bots, is set to a large value 500 if (x_k, y_k) is inside the (extended) collision region at time step k

$$c_c(X_k, \text{agent}_k) = \begin{cases} 500 & \text{proj}_f < (0.5cl + cr) \text{ and } \text{proj}_l < 0.5cw, \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

where cr is the turning circle radius, and cl, cw are the collision region length and width, respectively, as seen in Table 6.1. The computation of the projection distance proj_f and proj_l are described below

$$\text{proj}_f = \text{abs}\left(\mathbf{1}_f[\text{dis}_x, \text{dis}_y]^T\right), \text{proj}_l = \text{abs}\left(\mathbf{1}_l[\text{dis}_x, \text{dis}_y]^T\right)$$

where $\text{dis}_x = x_{\text{agent},k} - x_k, \text{dis}_y = y_{\text{agent},k} - y_k,$ and

$$\mathbf{1}_f = \left[\cos\left(\theta_{\text{agent},k}\right), \sin\left(\theta_{\text{agent},k}\right)\right], \mathbf{1}_l = \left[\cos\left(\theta_{\text{agent},k} - \frac{\pi}{2}\right), \sin\left(\theta_{\text{agent},k} - \frac{\pi}{2}\right)\right].$$

6.2.2.3 MPPI algorithm

The standard MPPI [120] algorithm is summarized in Algorithm 4 (blue parts indicate differences from the proposed o-MPPI) with the temperature parameter selected as $\lambda = 2.0$ and the covariance matrix is selected as $\Sigma_\epsilon = \text{diag}([4.0^2, 1.0^2])$ to enable sufficient exploration. In general, an additional cost can be added to the cost function for input deviation as $c(\cdot) = \sum_{k=0}^{N-1} \gamma u_k^T \Sigma_\epsilon^{-1} \epsilon_k$ [120]. However, the hyperparameter γ is set to zero to promote exploration in the dynamic environment with moving obstacles. The mean of the input distribution u_{mean} in Eq. (6.3) and the prediction horizon T are varied to illustrate their impact on MPPI effectiveness in the results and discussion Section 6.3.

6.2.2.4 Weighting

Given the trajectory costs S^m for all the rollouts $(0 \le m \le M-1)$ based on the cost function, the weight w^m associated with each rollout $(0 \le m \le M-1)$ is computed as $w^m = \frac{1}{\eta}(S^m - \beta)$, where β is the minimum cost of the rollouts $\beta = \min_m [S_m]$ and the normalizing factor η is $\eta = \sum_{m=0}^{M-1} \exp(-\frac{1}{\lambda}(S_m - \beta))$.

6.2.3 o-MPPI

The proposed o-MPPI (see Aglorithm 3) differs from the standard MPPI in the sampling of the output and the use of inverse to find the input – these are described below. The proposed o-MPPI uses the same temperature parameter λ, cost function and weighting to determine the input as the standard MPPI case in Sections 6.2.2.2 and 6.2.2.3.

Algorithm 4 Standard MPPI from [75]

Given: Number of rollouts & time steps M, N; Cost function; Temperature parameter λ; Forward model F;
Control hyperparameters: $\Sigma_\epsilon, \phi, q, \gamma \in [0, 1]$
$(u_0, u_1, ..u_{N-1})$ Initial control sequence
while task not completed **do**
  $X_k \leftarrow$ state estimate
  for $m \rightarrow 0$ to $M-1$ in parallel **do**
    $X_0^m \leftarrow X_k$, $S^m \leftarrow 0$
    [**Sampling**] Sample the m^{th} input perturbation rollout $\mathcal{E}^m = \{\epsilon_0^m, \epsilon_1^m, \ldots, \epsilon_{N-1}^m\}$
    [**Forward**] Compute the trajectory rollout y^m from the forward model F and the perturbed input u_ϵ^m computed from $\mathcal{E}^m$ as in Eq. (6.3)
    Calculate the trajectory cost S^m based on the cost function and the m^{th} rollout (u_m, X^m, y_m)
  end for
  [**Weighting**] For $m = 1, 2, \ldots, M$, compute the normalized weights $\{w_m\}$ based on the trajectory costs $\{S^m\}$ and the selected temperature parameter λ
  Obtain the weighted average $u = \sum_{m=0}^{M} w_m u^m$
  Apply the first entry of u to the system
  Warm start [75] and $u_{N-1} \leftarrow$ Initialize(u_{N-1})
  $k \leftarrow k + 1$
end while

6.2.3.1 Sampling the trajectory rollout

The generic way in Fig. 6.2 is used to sample the trajectory rollouts in this chapter with only two waypoints: the initial output and the final output. Specifically, a rectangle-shape region of interest for the endpoint $Y_{k+N-1} = \begin{bmatrix} x_{k+N-1} & y_{k+N-1} \end{bmatrix}$ is determined based on the allowed

magnitudes of the inputs u and the prediction horizon N. Then, the m^{th} sampled trajectory rollout is generated by fitting cubic splines between the current output and the m^{th} final output $Y^m_{k+N-1} = (x_e, y_e)$ sampled inside the region of interest, as seen in the right plot in Fig. 6.3. Specifically, the m^{th} rollout expression $t \in [0, T]$ (for the output x) is selected as a cubic spine as in [122]

$$x^m_d(t) = a_0 t + a_1 t + a_2 t^2 + a_3 t^3, \tag{6.12}$$

where the coefficients a_i for $i = 1, \ldots, 4$ can be determined from the boundary conditions

$$x(0) = x_k, \dot{x}(0) = v_k \cos(\theta_k), x(T) = x^m_e, \dot{x}(T) = v^m_e \sin(\theta^m_e),$$

where the endpoint orientation θ^m_e is designed to be aligned with the road direction and the endpoint forward speed v^m_e is the travel distance divided by the prediction horizon T, i.e., $v_e = \sqrt{(x_e - x_k)^2 + (y_e - y_k)^2}/T$. The output $y^m_d(t)$ is generated in a similar manner with the selected final waypoint output y^m_e.

6.2.3.2 Inverse model G^{-1}

Given a differentiable output trajectory $x_d(t), y_d(t)$ with $t \in [0, T]$ and starting state X_k, i.e., $x_d(0) = x_k$, $\dot{x}_d(0) = v_k \cos(\theta_k)$, $y_d(0) = y_k$, $\dot{y}_d(0) = v_k \sin(\theta_k)$, the inverse input can be obtained at time steps $k + j$ with $j = 0, 1, \ldots, N - 1$ from the forward dynamics in Eq. (6.7) to (6.8), as:

$$v^{\text{des}}_{k+j} = \frac{v_{p,k+j+1} - v_{p,k+j}}{\alpha \Delta t} + v_{p,k+j}, \quad \omega^{\text{des}}_{k+j} = \frac{\omega_{p,k+j+1} - \omega_{p,k+j}}{\alpha \Delta t} + \omega_{p,k+j} \tag{6.13}$$

where

$$v_{p,k+j} = \sqrt{\dot{x}^2_d((k + j - k)\Delta t) + \dot{y}^2_d((k + j - k)\Delta t)}, \tag{6.14}$$

$$\omega_{p,k+j} = \begin{cases} (\theta_{p,k+j} - \theta_{p,k+j-1})/\Delta t & j \geq 0 \\ \omega_k & j = 0 \end{cases} \tag{6.15}$$

$$\theta_{p,k+j} = \arctan\left(\frac{\dot{y}_d((k + j - k)\Delta t)}{\dot{x}_d((k + j - k)\Delta t)}\right), \tag{6.16}$$

$v_{p,k+j}, \omega_{p,s}, \theta_{d,s}$ are the planned speed, angular velocity, and the orientation at time step s, respectively. Saturation is not considered when computing the inverse input in Eq. (6.13). However, if the final optimal input is large (for either o-MPPI or standard MPPI), then it is saturated in the experimental system and bounded in simulations by the saturation function in Eqs. (6.7)- (6.8).

6.3 Results and discussion

The ability to avoid and successfully overtake moving obstacles in dynamic environments are comparatively evaluated below for the proposed o-MPPI and the standard MPPI. The evaluation starts with the study of a single dynamic obstacle, which is followed by the case with multiple obstacles.

6.3.1 Case with a single dynamic obstacle

For comparative evaluation with a single obstacle, three MPPI and one o-MPPI cases are simulated with the following conditions.

- **Case 1 o-MPPI:** The proposed o-MPPI with prediction horizon T = 2.0 s.

- **Case 2 Small-horizon standard MPPI:** Standard MPPI with prediction horizon T = 2.0 s, i.e., prediction steps N = 50 with sampling time period Δt = 0.04 s. The nominal speed is 15 cm/s, i.e., with initial control sequence (in Algorithm 4) $(u_0, u_1, ..u_{N-1}) \equiv [15 \text{ cm/s}, 0 \text{ rad/s}]$;

- **Case 3 Large-horizon standard MPPI:** Standard MPPI as in **Case 2** except with a larger prediction horizon T = 8.0 s and corresponding prediction steps N = 200;

- **Case 4 Slow-initial standard MPPI:** Standard MPPI as in **Case 2** except that the initial speed is lower at 10 cm/s i.e., with initial control sequence (in Algorithm 4) $(u_0, u_1, ..u_{N-1}) \equiv [10 \text{ cm/s}, 0 \text{ rad/s}]$.

The controlled bot (green in Fig. 6.5) is said to achieve a successful overtake of a moving obstacle (blue circle in Fig. 6.5) if: (i) the controlled bot is driving in the counter-clockwise direction all the time, (ii) the controlled bot does not run outside of the track or into the collision regions of the moving obstacle, and (iii) within a specified time, the position of the controlled bot is ahead of the slower constant-speed obstacle. as indicated by the red dashed lines in Fig. 6.6. For all these cases, the initial position of the moving obstacle is $(x, y) = (85\text{ cm}, 50\text{ cm})$ on one corner of the track as shown in Fig. 6.6 and it has a constant speed of 10 cm/s. The simulation is run till the moving obstacle reaches the other corner of the track as indicated in Fig. 6.6. The initial position of the controlled bot is behind the moving obstacle at $X_0 = \begin{bmatrix} x_0 & y_0 & \theta_0 & v_0 & \omega_0 \end{bmatrix}^T = \begin{bmatrix} 85\text{ cm} & -10\text{ cm} & \frac{\pi}{2}\text{ rad} & 15\text{ cm/s} & 0\text{ rad/s} \end{bmatrix}^T$. The initial trajectory rollouts for o-MPPI and standard MPPI for the four different cases are visualized in Fig. 6.5 for comparative evaluation. Additionally, 100 repeated simulations were run, and the success rates for the four cases with different numbers of rollouts M are tabulated in Table 6.2 and 6.3.

6.3.1.1 Less number of rollouts (efficient exploration)

The proposed o-MPPI method requires **20**-times less number of rollouts compared to the standard MPPI, as seen in Tables 6.3 and 6.2, In particular, to achieve 100% success rate of overtaking, the standard MPPI requires $M = \mathbf{1000}$ rollouts with a prediction horizon $T = 8.0$ s while the o-MPPI method needs $M = \mathbf{50}$ rollouts with a prediction horizon $T = 2.0$ s. Thus, the proposed o-MPPI achieves a more efficient exploration compared to standard MPPI.

6.3.1.2 Smaller prediction horizon

The o-MPPI method requires **4**-times smaller prediction horizon compared to the standard MPPI as seen in Table 6.3 and 6.2. In particular, to achieve 100% success rate in overtaking, the standard MPPI requires a prediction horizon $T = \mathbf{8.0}$ s while the inversion-based sampling method needs only $M = 50$ rollouts with a prediction horizon $T = \mathbf{2.0}$ s.

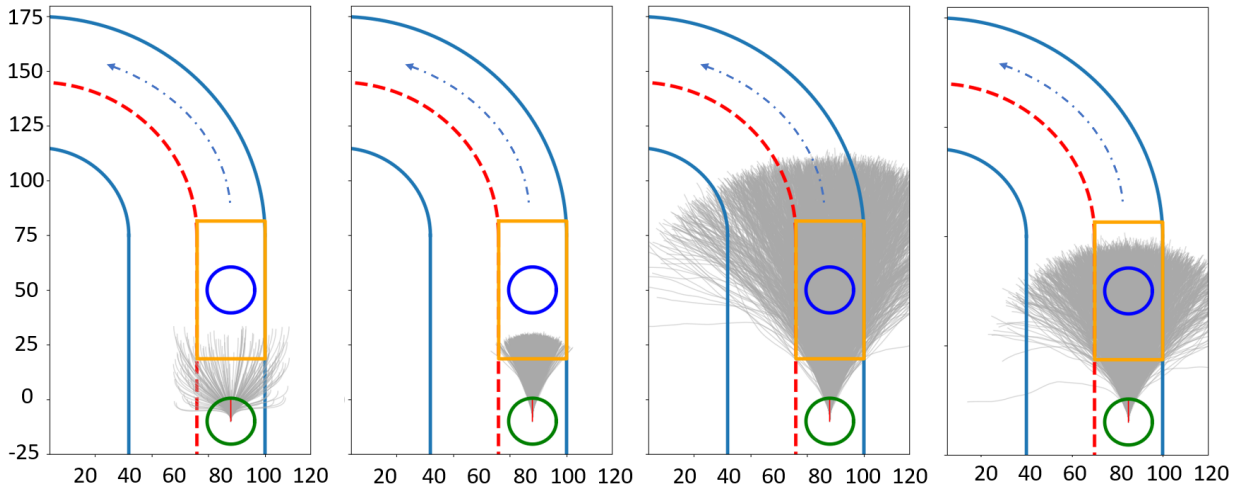

Figure 6.5: Initial rollouts for **Cases 1 to 4** (left to right). The green circle represents the controlled bot with the red line indicating its orientation. The blue circle denotes the constant-speed bot of speed 10 cm/s. The bright yellow rectangle depicts the collision region of the constant-speed bot. The cost function component c_c in Eq. (6.11) will be a large value if the controlled bot (x_k, y_k) runs inside the (extended) collision region. Grey lines demonstrate the trajectory rollouts of **Case 1 to 4** in Section 6.3.1.

When the standard MPPI method is restricted to a prediction horizon of 2.0 s, the controlled bot only achieves around 25% success rate, even with a large number of rollouts M. This is because o-MPPI has rollouts that run into the other lane as seen in the second left plot in Fig. 6.5. In contrast, rollouts of MPPI (rollout number $M = 2000$) with a prediction horizon $T = 2.0$ s) are mostly restricted to within the original lane. Note that the explored area of the standard MPPI increases substantially with a larger prediction horizon of $T = 8.0$ s as seen in Fig. 6.5.

In this sense, the proposed o-MPPI with a smaller prediction horizon requirement is beneficial since requiring a larger prediction horizon might not be feasible due to sensor-range limitations and it can also lead to increased computational load.

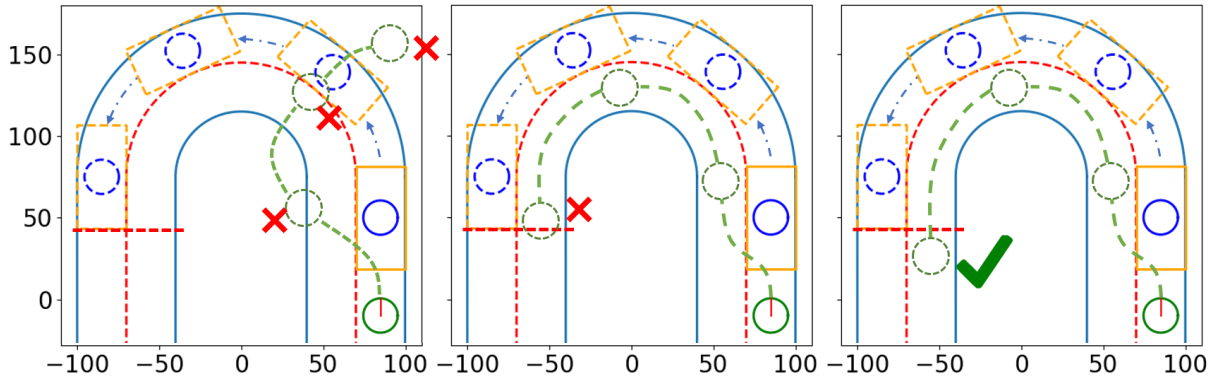

Figure 6.6: Successful versus unsuccessful overtake, as defined in Section 6.3.1. (Left plot) Unsuccessful overtake where the controlled bot runs outside the track, passes through the moving obstacle, or drives in the clockwise direction. (Middle plot) Unsuccessful overtake where the final position is not ahead of the moving obstacle's collision region within specified time. (Right plot) Successful overtake that avoids undesirable situations in the left and the middle plots.

Table 6.2: o-MPPI-controlled bot success rate (%) of overtakes with T = 2.0 s for **Case 1** in Section 6.3.1.

M	50	100	200
success rate	100	100	100

6.3.1.3 MPPI performance is susceptible to the slow initial

MPPI performance is susceptible to improper initial input sequences, even with sufficient prediction horizon T = 8.0 s and number of rollouts M = 2000. For **Case 4** with slow-initial speed of 10 cm/s example, the controlled bot has difficulty overtaking the moving obstacle since the explored area has less overlap with the overtake regions, as seen in Fig. 6.5. Therefore, MPPI performance is affected by the initialization strategy and is susceptible to dynamic environments.

Table 6.3: MPPI-controlled bot success rate (%) of overtakes for different prediction horizon T and number of rollouts M for **Cases 2 to 4** in Section 6.3.1

T	M	succ. rate	T	M	succ. rate
Case 2					
2.0	50	22	4.0	50	4
2.0	500	28	4.0	500	2
2.0	1000	15	4.0	1000	6
2.0	2000	22	4.0	2000	2
Case 3					
8.0	50	49	6.0	50	34
8.0	500	75	6.0	500	52
8.0	1000	100	6.0	1000	89
8.0	2000	100	6.0	2000	97
Case 4					
8.0	2000	1			

6.3.1.4 Trade-off in sampling time Δt selection with MPPI

Generally, a higher sampling rate for control is beneficial since this can lead to more precision. However, a higher rate Δt of control also results in a higher dimensional search space for input sampling, which requires more samples to be drawn for sufficient exploration, as illustrated in Fig. 6.7. Less exploration can also result in less success since the best rollout may not be explored when the environment changes too fast or uncertainties are large. In contrast, since the o-MPPI trajectory rollouts are sampled from the potential set of waypoints in the output space that only depends on the prediction horizon T, the control sampling rate Δt does not affect the explored region.

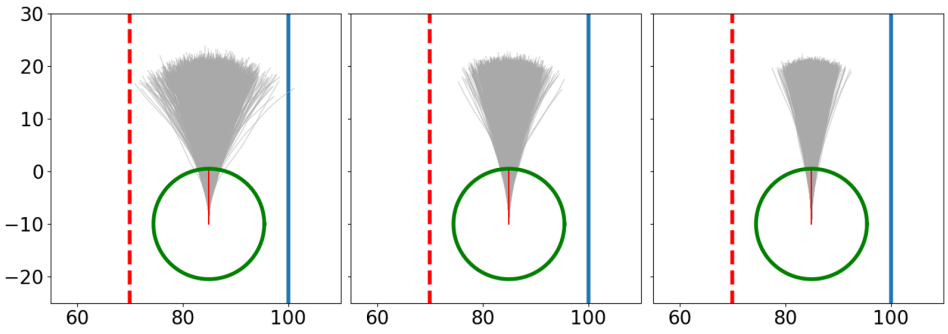

Figure 6.7: The exploration area (black shaded area) is smaller with an increasing sampling rate from 25 Hz (left), 50 Hz (middle) to 100 Hz (right). All sampling rates have the same prediction horizon of T = 2.0 s, the same initial state, the same nominal inputs and the same input distribution, and the same number of rollouts M = 2000.

6.3.2 Case with multiple dynamic obstacles

For comparative evaluation with multiple obstacles, two moving obstacles are considered as in the left plot of Fig. 6.3. The cost function is kept the same as in the previous Section 6.3.1 with a single dynamic obstacle.

The first obstacle has a constant speed of 10 cm/s (red circle) and tracks the inner lane with initial position (x, y) = $(23\text{ cm}, -156\text{ cm})$ and the second obstacle has speed 12 cm/s (blue circle) and tracks the outer lane with initial position (x, y) = $(-55\text{ cm}, -75\text{ cm})$. The initial position of the controlled bot (green) is behind the first moving obstacle and has the initial condition $X_0 = \begin{bmatrix} x_0 & y_0 & \theta_0 & v_0 & \omega_0 \end{bmatrix}^T = \begin{bmatrix} -55\text{ cm} & -10\text{ cm} & -\frac{\pi}{2}\text{ rad} & 15\text{ cm/s} & 0\text{ rad/s} \end{bmatrix}^T$ Comparative simulation and experiments of o-MPPI and standard MPPI were performed with settings as in **Case 1** (o-MPPI) and **Case 3** (Larger-horizon standard MPPI) proposed in Section 6.3.1.

6.3.2.1 Comparison of two obstacle avoidance

Only the o-MPPI-controlled bot managed to successfully maneuver around both moving obstacles after slowing down because both lanes were blocked by the two slower constant-speed bots as seen in the trace plots in Fig. 6.8. First, the o-MPPI-controlled bot switched to the inner lane since the inner-lane, constant-speed bot has a relatively faster speed at 12 cm/s compared to the outer-lane, constant-speed bot moving at 10 cm/s and second then the o-MPPI-controlled bot switched back to the outer lane once there is sufficient room to drive at a faster speed that is closer to the desired speed 20 cm/s. In contrast, the standard MPPI-controlled bot got stuck behind the constant-speed bot, as seen from the trace in the upper region of the track in Fig. 6.8. As shown with **Case 4** (slow-initial standard MPPI), with a smaller speed the nominal mean of the MPPI input distribution become smaller, and can lead to lower success rate for overtaking. It is noted that both the o-MPPI-controlled bot and the MPPI-controlled bot were able to successfully maneuver around a single obstacle when the controlled bot is moving at a higher speed as seen in the accompanying video at https://youtu.be/snhlZj3l5CE.

6.3.2.2 Oscillatory driving versus overtaking

Comparison of overlays of the experimental snapshots in Fig. 6.9 (left) shows that the MPPI-controlled bot can not only get stuck behind the slower moving obstacle bot (with speed 10 cm/s in the outer lane), but it also tends to have oscillatory driving behavior. This is because the oscillatory motion allows the bot to achieve a relatively faster speed closer to its desired speed in the cost function component c_v in Eq. (6.10). In contrast, overlays of the o-MPPI-controlled bot in Fig. 6.9 (right) show that it is able to maintain higher speed by successfully switching to the inner lane since the obstacle bot in the inner lane in the front has a higher speed 12 cm/s compared to the bot in the outer lane with speed 10 cm/s.

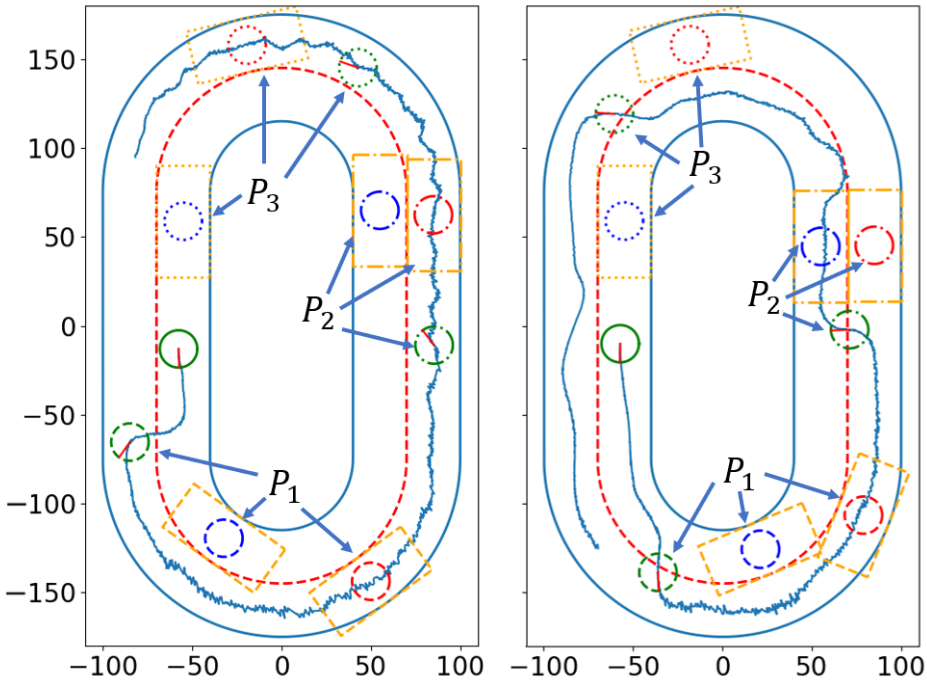

Figure 6.8: Experimental traces of the MPPI-controlled bot (left) and o-MPPI-controlled bot (right). The o-MPPI-controlled bot could maneuver multiple times (snapshots of positions are indicated as P_1 (dashed), P_2 (dash-dotted), and P_3 (dotted plots)) to overtake two constant-speed moving bots. However, the MPPI-controlled bot got stuck behind the constant-speed moving bot (blue) although it switched to the outer lane initially (at position P_1 (dashed plot)) to avoid a single moving bot. Note that the MPPI-controlled bot moved in a zig-zag manner, which is also seen in Fig. 6.9. The full experiments can be seen in the accompanying video at https://youtu.be/snhlZj3l5CE.

6.4 Chapter conclusion

This chapter proposed an output-sampling-based model predictive path integral control (o-MPPI) to improve the efficiency of standard MPPI. An advantage of the proposed output sampling is that it can leverage the substantial work on trajectory planning in robotics to meet constraints that are often posed in the output space, which in turn improves the

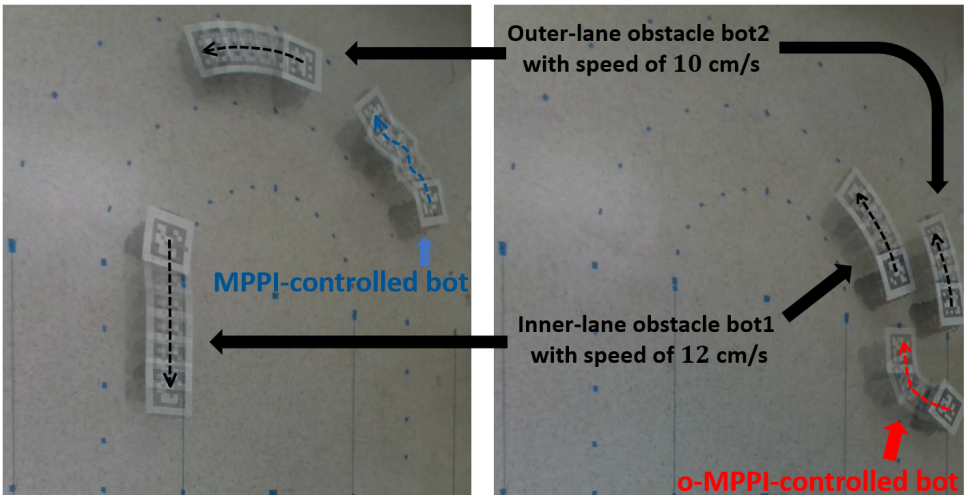

Figure 6.9: Overlays of experimental images for multiple dynamic obstacles: (i) (left) the standard MPPI with a controlled bot that remains stuck and (ii) (right) the proposed o-MPPI where the controlled bot successfully manages to switch lanes and eventually overtake the slower moving obstacle. The full experiment can be viewed in the accompanying video at https://youtu.be/snhlZj3l5CE.

efficiency of MPPI. Instead of forward models that map from input to output as in standard MPPI, the proposed o-MPPI uses inverse models to map from the sampled output to inputs. The improved efficiency of the o-MPPI was seen in both the simulation and the experimental results — o-MPPI required less number of rollouts and a shorter prediction horizon, compared to the standard MPPI.

Chapter 7

SUMMARY AND FUTURE WORK

7.1 Summary of the main contributions

The research timeline can be seen in Fig. 1.2. The main contributions proposed in each **SC** are summarized below:

1. Chapter 3 (based on the work [1]) i) proposed using the input-weighted complex Gaussian process for modeling the unknown interaction dynamics (the challenge in **SC1**), ii) achieved precision for SEA robots, iii) developed bounds on iteration gain design for multi-input-multi-output iterative learning control.

2. Chapter 4 & 5 (based on the work [2, 3]) i) identified needed observables (sufficient time history T of the output and time derivatives up to order r, relative degree of the system) to remove hidden-state dependence (the challenge in **SC2**) in multi-input-multi-output square systems, ii) proposed an Algorithm for finding precision inverse operator from data.

3. Chapter 6 (based on the work [4]) i) proposed new output-sampling-based MPPI (o-MPPI), ii) showed that o-MPPI is more efficient than standard MPPI in terms of (20-times) less number of rollouts, (4-times) shorter prediction horizon and robustness to the slow initial speed for increased efficiency (the challenge in **SC3**) using both simulation and experiment results.

7.2 Future work

The same type of dependence on the hidden states, as discussed in Chapter 4 & 5, affects the forward operator as well because the dependence of the input u and output y relation on the

hidden states η follows from Eq. (5.7). Therefore, the forward operator form can be proposed as in Eq. (5.84) and can be trained from the measured output and its time derivatives. It is noted that instantaneous time-derivatives also were selected as lifted observables in [31] to better linearlize the nonlinearties.

Neural-net-based inverse operator presented in Chapter 4 & 5 can be baked into model predictive control (MPC) by following the mechanisim how the neural-net-based forward model is used in MPC [123, 124, 125, 126, 127, 128, 129, 130, 131, 132].

For the output-sampling-based MPPI presented in Chapter 6, future work includes exploring the use of other trajectory-planning methods for output sampling, as well as the use of data-enabled deep-learning or Gaussian process models for the inverse opeartor developed in Chapter 5 in different scenarios, e.g., drones. A blending of the input sampling and output sampling can also be considered depending on the type of constraints. On the theory side, threoy can be developed for o-MPPI to establish the connection to the standard MPPI and the stochastic optimal control theory.

BIBLIOGRAPHY

[1] Leon Liangwu Yan, Nathan Banka, Parker Owan, Walter Tony Piaskowy, Joseph L Garbini, and Santosh Devasia. MIMO ILC using complex-kernel regression and application to precision SEA robots. *Automatica*, 127:109550, 2021.

[2] Leon Liangwu Yan and Santosh Devasia. Precision data-enabled koopman-type inverse operators for linear systems. *IFAC-PapersOnLine*, 55(37):181–186, 2022.

[3] Liangwu Yan and Santosh Devasia. What observables are needed for precision data-enabled learning of inverse operators? *Journal of Dynamic Systems, Measurement, and Control*, to appear, 2024.

[4] Leon Yan and Santosh Devasia. Output-sampled model predictive path integral control (o-MPPI) for increased efficiency. In *2024 International Conference on Robotics and Automation (ICRA)*, pages (accepted, to appear). IEEE, 2024.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[6] OpenAI. GPT-4 technical report, 2023.

[7] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227, 2018.

[8] Peter Schmid and Joern Sesterhenn. Dynamic mode decomposition of numerical and experimental data. *Bulletin of the American Physical Society*, 53, 2008.

[9] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[10] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[11] Vladimir Vovk. Kernel ridge regression. In *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pages 105–116. Springer, 2013.

[12] Nikolaus Kriegeskorte and Tal Golan. Neural network models and deep learning. *Current Biology*, 29(7):R231–R236, 2019.

[13] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[14] Geetika Malhotra, Seep Goel, and Smruti R Sarangi. Gputejas: A parallel simulator for gpu architectures. In *2014 21st International Conference on High Performance Computing (HiPC)*, pages 1–10. IEEE, 2014.

[15] Attila Kakay, Elmar Westphal, and Riccardo Hertel. Speedup of fem micromagnetic simulations with graphical processing units. *IEEE transactions on magnetics*, 46(6):2303–2306, 2010.

[16] Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. *Advances in neural information processing systems*, 32, 2019.

[17] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Ga3c: Gpu-based a3c for deep reinforcement learning. *CoRR abs/1611.06256*, 2016.

[18] Gill A. Pratt and Matthew M. Williamson. Series Elastic Actuators. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 399–406, 1995.

[19] David W Robinson. *Design and Analysis of Series Elasticity in Closed-loop Actuator Force Control*. Ph.d. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.

[20] Nicholas Paine, Sehoon Oh, and Luis Sentis. Design and control considerations for high-performance series elastic actuators. *IEEE/ASME Transactions on Mechatronics*, 19(3):1080–1091, 2014.

[21] L. Qui and E. J. Davison. Performance limitations of non-minimum phase systems in the servomechanism problem. *Automatica*, 29, March:337–349, 1993.

[22] M. W. Spong. Modeling and control of elastic joint robots. *ASME. J. Dyn. Sys., Meas., Control.*, 109(4):310–318., Dec 1987.

[23] B. C. Chiou and M. Shahinpoor. The effects of joint and link flexibilities on the dynamic stability of force-controlled manipulators. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 398–403 vol.1, May 1989.

[24] A. de Luca and P. Lucibello. A general algorithm for dynamic feedback linearization of robots with elastic joints. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 1, pages 504–510 vol.1, May 1998.

[25] A. de Luca, R. Farina, and P. Lucibello. On the control of robots with visco-elastic joints. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4297–4302, April 2005.

[26] Steven Daniel Eppinger. *Modeling robot dynamic performance for endpoint force control*. Phd thesis, Massachusetts Institute of Technology, 1988.

[27] Berk Altın and Kira Barton. Exponential stability of nonlinear differential repetitive processes with applications to iterative learning control. *Automatica*, 81:369–376, 2017.

[28] Xiang Li, Yun-Hui Liu, and Haoyong Yu. Iterative learning impedance control for rehabilitation robots driven by series elastic actuators. *Automatica*, 90:1–7, 2018.

[29] Jeremy G Stoddard, Georgios Birpoutsoukis, Johan Schoukens, and James S Welsh. Gaussian process regression for the estimation of generalized frequency response functions. *Automatica*, 106:161–167, 2019.

[30] Ian Abraham, Gerardo De La Torre, and Todd D Murphey. Model-based control using koopman operators. In *2017 Robotics: Science and Systems, RSS 2017*. MIT Press Journals, 2017.

[31] Giorgos Mamakoukas, Maria L Castano, Xiaobo Tan, and Todd D Murphey. Derivative-based koopman operators for real-time control of robotic systems. *IEEE Transactions on Robotics*, 2021.

[32] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[33] Farshid Asadi, Alaa Olleak, Jingang Yi, and Yuebin Guo. Gaussian process (GP)-based learning control of selective laser melting process. In *2021 American Control Conference (ACC)*, pages 508–513. IEEE, 2021.

[34] Stephen Piche, James D Keeler, Greg Martin, Gene Boe, Doug Johnson, and Mark Gerules. Neural network based model predictive control. In *Advances in Neural Information Processing Systems*, pages 1029–1035, 2000.

[35] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[36] Juš Kocijan, Roderick Murray-Smith, Carl Edward Rasmussen, and Agathe Girard. Gaussian process model based predictive control. In *Proceedings of the 2004 American control conference*, volume 3, pages 2214–2219. IEEE, 2004.

[37] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *J. of Robotic Systems*, 1(2):123–140, March 1984.

[38] S. Mishra and M.Tomizuka. Segmented iterative learning control for precision positioning of waferstages. In *2007 IEEE/ASME international conference on advanced intelligent mechatronics AIM*, pages 1–6, Sept 2007.

[39] B. Paden, D. Chen, R. Ledesma, and E. Bayo. Exponentially stable tracking control for multi-joint flexible manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, 115(1):53–59, 1993.

[40] Yongqiang Ye and Danwei Wang. Clean system inversion learning control law. *Automatica*, 41(9):1549–1556, 2005.

[41] Szuchi Tien, Qingze Zou, and Santosh Devasia. Iterative control of dynamics-coupling-caused errors in piezoscanners during high-speed afm operation. *IEEE Transactions on Control Systems Technology*, 13(6):921–931, 2005.

[42] J. Ghosh and B. Paden. Nonlinear repetitive control. *IEEE Transactions on Automatic Control*, 45(5):949–954, 2000.

[43] H.-S. Ahn, Y. Q. Chen, and K. L. Moore. Iterative learning control: Brief survey and categorization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1099–1121, 2007.

[44] Kyong-Soo Kim and Qingze Zou. A modeling-free inversion-based iterative feedforward control for precision output tracking of linear time-invariant systems. *IEEE/ASME Transactions on Mechatronics*, 18(6):1767–1777, 2012.

[45] Andreas Deutschmann, Pavel Malevich, Andrius Baltuška, and Andreas Kugi. Modeling and iterative pulse-shape control of optical chirped pulse amplifiers. *Automatica*, 98:150–158, 2018.

[46] Robin de Rozario and Tom Oomen. Data-driven iterative inversion-based control: Achieving robustness through nonlinear learning. *Automatica*, 107:342–352, 2019.

[47] Santosh Devasia. Iterative machine learning for output tracking. *IEEE Transactions on Control Systems Technology*, 27(2):516–526, 2017.

[48] Yan Yan, Haiming Wang, and Qingze Zou. A decoupled inversion-based iterative control approach to multi-axis precision positioning: 3d nanopositioning example. *Automatica*, 48(1):167–176, 2012.

[49] Nathan Banka, W Tony Piaskowy, Joseph Garbini, and Santosh Devasia. Iterative machine learning for precision trajectory tracking with series elastic actuators. In *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*, pages 234–239. IEEE, 2018.

[50] Robin De Rozario, Juliana Langen, and Tom Oomen. Multivariable learning using frequency response data: a robust iterative inversion-based control approach with application. In *American Control Conference (ACC)*, pages 2215–2220. IEEE, 2019.

[51] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

[52] Rafael Boloix Tortosa, Juan José Murillo Fuentes, Francisco Javier Payán Somet, and Fernando Pérez-Cruz. Complex gaussian processes for regression. *IEEE transactions on neural networks and learning systems*, 29(11):5499–5511, 2018.

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

[54] John Lataire and Tianshi Chen. Transfer function and transient estimation by gaussian process regression in the frequency domain. *Automatica*, 72:217–229, 2016.

[55] Gianluigi Pillonetto, Francesco Dinuzzo, Tianshi Chen, Giuseppe De Nicolao, and Lennart Ljung. Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3):657–682, 2014.

[56] Jean Tarbouriech, Shubhanshu Shekhar, Matteo Pirotta, Mohammad Ghavamzadeh, and Alessandro Lazaric. Active model estimation in markov decision processes. In *Conference on Uncertainty in Artificial Intelligence*, pages 1019–1028. PMLR, 2020.

[57] Hyung-Jin Yoon, Donghwan Lee, and Naira Hovakimyan. Hidden markov model estimation-based Q-learning for partially observable markov decision process. In *2019 American Control Conference (ACC)*, pages 2366–2371. IEEE, 2019.

[58] Jennifer Pohle, Roland Langrock, Floris M van Beest, and Niels Martin Schmidt. Selecting the number of states in hidden markov models: pragmatic solutions illustrated using animal movement. *Journal of Agricultural, Biological and Environmental Statistics*, 22(3):270–293, 2017.

[59] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

[60] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[61] Lennart Ljung et al. Theory for the user. *System Identification*, 1987.

[62] Hong Thom Pham, Bo-Suk Yang, et al. A hybrid of nonlinear autoregressive model with exogenous input and autoregressive moving average model for long-term machine state forecasting. *Expert Systems with Applications*, 37(4):3310–3317, 2010.

[63] Mason Kamb, Eurika Kaiser, Steven L Brunton, and J Nathan Kutz. Time-delay observables for koopman: Theory and applications. *SIAM Journal on Applied Dynamical Systems*, 19(2):886–917, 2020.

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

[65] Leontine Aarnoudse, Wataru Ohnishi, Maurice Poot, Paul Tacx, Nard Strijbosch, and Tom Oomen. Control-relevant neural networks for intelligent motion feedforward. In *2021 IEEE International Conference on Mechatronics (ICM)*, pages 1–6. IEEE, 2021.

[66] Jayati Ghosh and Brad Paden. Iterative learning control for nonlinear nonminimum phase plants. *J. Dyn. Sys., Meas., Control*, 123(1):21–30, 2001.

[67] Benjamin T Fine, Sandipan Mishra, and Masayoshi Tomizuka. Model inverse based iterative learning control using finite impulse response approximations. In *2009 American Control Conference*, pages 931–936. IEEE, 2009.

[68] Kuo-Tai Teng and Tsu-Chin Tsao. A comparison of inversion based iterative learning control algorithms. In *2015 American Control Conference (ACC)*, pages 3564–3569. IEEE, 2015.

[69] Isaac A Spiegel, Nard Strijbosch, Tom Oomen, and Kira Barton. Iterative learning control with discrete-time nonlinear nonminimum phase models via stable inversion. *International Journal of Robust and Nonlinear Control*, 31(16):7985–8006, 2021.

[70] S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. *IEEE Transactions on Automatic Control*, 41(7):930–943, 1996.

[71] Jan C Willems, Paolo Rapisarda, Ivan Markovsky, and Bart LM De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, 2005.

[72] Qingze Zou and Santosh Devasia. Preview-based stable-inversion for output tracking of linear systems. *ASME J. Dyn. Syst. Meas. Control,*, 1999.

[73] Qingze Zou and Santosh Devasia. Precision preview-based stable-inversion for nonlinear nonminimum-phase systems: The vtol example. *Automatica*, 43(1):117–127, 2007.

[74] Max van Meer, Maurice Poot, Jim Portegies, and Tom Oomen. Gaussian process based feedforward control for nonlinear systems with flexible tasks: With application to a printer with friction. *IFAC-PapersOnLine*, 55(37):241–246, 2022.

[75] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[76] Ihab S Mohamed, Guillaume Allibert, and Philippe Martinet. Model predictive path integral control framework for partially observable navigation: A quadrotor case study. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 196–203. IEEE, 2020.

[77] Ji Yin, Zhiyuan Zhang, Evangelos Theodorou, and Panagiotis Tsiotras. Trajectory distribution control for model predictive path integral control using covariance steering. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1478–1484. IEEE, 2022.

[78] Chuyuan Tao, Hunmin Kim, and Naira Hovakimyan. RRT-guided model predictive path integral method. *arXiv preprint arXiv:2301.13143*, 2023.

[79] Taekyung Kim, Gyuhyun Park, Kiho Kwak, Jihwan Bae, and Wonsuk Lee. Smooth model predictive path integral control without smoothing. *IEEE Robotics and Automation Letters*, 7(4):10406–10413, 2022.

[80] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning*, pages 750–759. PMLR, 2022.

[81] Ihab S Mohamed, Kai Yin, and Lantao Liu. Autonomous navigation of agvs in unknown cluttered environments: log-mppi control strategy. *IEEE Robotics and Automation Letters*, 7(4):10240–10247, 2022.

[82] Dylan M Asmar, Ransalu Senanayake, Shawn Manuel, and Mykel J Kochenderfer. Model predictive optimized path integral strategies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3182–3188. IEEE, 2023.

[83] Raphael Kusumoto, Luigi Palmieri, Markus Spies, Akos Csiszar, and Kai O Arras. Informed information theoretic model predictive control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2047–2053. IEEE, 2019.

[84] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574. IEEE, 2011.

[85] Liang Ma, Jianru Xue, Kuniaki Kawabata, Jihua Zhu, Chao Ma, and Nanning Zheng. Efficient sampling-based motion planning for on-road autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1961–1976, 2015.

[86] Santosh Devasia, Degang Chen, and Brad Paden. Nonlinear inversion-based output tracking. *IEEE Transactions on Automatic Control*, 41(7):930–942, 1996.

[87] Franziska Meier, Daniel Kappler, Nathan Ratliff, and Stefan Schaal. Towards robust online inverse dynamics learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4034–4039. IEEE, 2016.

[88] Athanasios S Polydoros, Lazaros Nalpantidis, and Volker Krüger. Real-time deep learning of robotic manipulator inverse dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3442–3448. IEEE, 2015.

[89] Diego Romeres, Mattia Zorzi, Raffaello Camoriano, Silvio Traversaro, and Alessandro Chiuso. Derivative-free online learning of inverse dynamics models. *IEEE Transactions on Control Systems Technology*, 28(3):816–830, 2019.

[90] S. Devasia. Should model-based inverse inputs be used as feedforward under plant uncertainty? *IEEE Trans. on Automatic Control*, 47(11):1865–1871, Nov, 2002.

[91] Esmaeil Naderi and Khashayar Khorasani. Inversion-based output tracking and unknown input reconstruction of square discrete-time linear systems. *Automatica*, 95:44–53, 2018.

[92] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[93] Nathan Banka and Santosh Devasia. Application of iterative machine learning for output tracking with magnetic soft actuators. *IEEE/ASME Transactions on Mechatronics*, 23(5):2186–2195, 2018.

[94] Peter J Schreier and Louis L Scharf. *Statistical signal processing of complex-valued data: the theory of improper and noncircular signals*. Cambridge university press, 2010.

[95] Riccardo Marino and Patrizio Tomei. *Nonlinear control design*. Prentice-Hall International, 1995.

[96] Charles A Desoer and M Vidyasagar. *Feedback Systems: Input-output Properties*, volume 55. SIAM, 1975.

[97] S. Kolavennu, S. Palanki, and J.C. Cockburn. Nonlinear control of nonsquare multivariable systems. *Chemical Engineering Science*, 56(6):2103–2110, 2001.

[98] Hassan K Khalil. Nonlinear systems third edition. *Patience Hall*, 115, 2002.

[99] Alberto Isidori. Nonlinear control systems. communications and control engineering. *Springer. 3rd edition.*, 1995.

[100] Jorge Gonçalves and Sean Warnick. Necessary and sufficient conditions for dynamical structure reconstruction of LTI networks. *IEEE Transactions on Automatic Control*, 53(7):1670–1674, 2008.

[101] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Generalizing koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17(1):909–930, 2018.

[102] Souransu Nandi, Victor Migeon, Tarunraj Singh, and Puneet Singla. Polynomial chaos-based controller design for uncertain linear systems with state and control constraints. *Journal of Dynamic Sys., Meas., and Control*, 140(7):071009, 2018.

[103] Emre Sariyildiz, Rahim Mutlu, and Chuanlin Zhang. Active disturbance rejection based robust trajectory tracking controller design in state space. *Journal of Dynamic Systems, Measurement, and Control*, 141(6), 2019.

[104] Ana Gabriela Gallardo Hernández, Leonid Fridman, Arie Levant, Yuri Shtessel, Ron Leder, Cristina Revilla Monsalve, and Sergio Islas Andrade. High-order sliding-mode control for blood glucose: Practical relative degree approach. *Control Engineering Practice*, 21(5):747–758, 2013.

[105] Gianmario Rinaldi and Antonella Ferrara. Automatic identification of the relative degree of nonlinear systems: Application to sliding mode control design and experimental assessment. *Control Engineering Practice*, 94:104207, 2020.

[106] Peng Guo. Nonlinear predictive functional control based on hopfield network and its application in CSTR. In *2006 International Conference on Machine Learning and Cybernetics*, pages 3036–3039. IEEE, 2006.

[107] Miao He, Xiaomin Wu, Guifang Shao, Yuhua Wen, and Tundong Liu. A semiparametric model-based friction compensation method for multijoint industrial robot. *Journal of Dynamic Systems, Measurement, and Control*, 144(3), 2022.

[108] Davide Astolfi, Francesco Castellani, and Francesco Natili. Wind turbine multivariate power modeling techniques for control and monitoring purposes. *Journal of Dynamic Systems, Measurement, and Control*, 143(3), 2021.

[109] H Hatze. The use of optimally regularized fourier series for estimating higher-order derivatives of noisy biomechanical data. *Journal of biomechanics*, 14(1):13–18, 1981.

[110] Karsten Ahnert and Markus Abel. Numerical differentiation of experimental data: local versus global methods. *Computer Physics Communications*, 177(10):764–774, 2007.

[111] Arie Levant. Higher-order sliding modes, differentiation and output-feedback control. *International journal of Control*, 76(9-10):924–941, 2003.

[112] John H Lagergren, John T Nardini, G Michael Lavigne, Erica M Rutter, and Kevin B Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A*, 476(2234):20190800, 2020.

[113] Floris Van Breugel, J Nathan Kutz, and Bingni W Brunton. Numerical differentiation of noisy data: A unifying multi-objective optimization framework. *IEEE Access*, 8:196865–196877, 2020.

[114] Marc Toussaint. A tutorial on newton methods for constrained trajectory optimization and relations to slam, gaussian process smoothing, optimal control, and probabilistic inference. *Geometric and numerical foundations of movements*, pages 361–392, 2017.

[115] Julius Jankowski, Lara Brudermüller, Nick Hawes, and Sylvain Calinon. Vp-sto: Via-point-based stochastic trajectory optimization for reactive robot behavior. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10125–10131. IEEE, 2023.

[116] Oktay Arslan and Panagiotis Tsiotras. Machine learning guided exploration for sampling-based motion planning algorithms. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2646–2652. IEEE, 2015.

[117] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019.

[118] Jacob J Johnson, Linjun Li, Fei Liu, Ahmed H Qureshi, and Michael C Yip. Dynamically constrained motion planning networks for non-holonomic robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6937–6943. IEEE, 2020.

[119] Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q-H Meng. Neural RRT*: Learning-based optimal path planning. *IEEE Transactions on Automation Science and Engineering*, 17(4):1748–1758, 2020.

[120] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[121] Ji Yin, Zhiyuan Zhang, and Panagiotis Tsiotras. Risk-aware model predictive path integral control using conditional value-at-risk. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7937–7943. IEEE, 2023.

[122] Zhimin Zhang, John Tomlinson, and Clyde Martin. Splines and linear control theory. *Acta Applicandae Mathematica*, 49:1–34, 1997.

[123] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.

[124] Katrine Seel, Esten I Grøtli, Signe Moe, Jan T Gravdahl, and Kristin Y Pettersen. Neural network-based model predictive control with input-to-state stability. In *2021 American Control Conference (ACC)*, pages 3556–3563. IEEE, 2021.

[125] Yi Ming Ren, Mohammed S Alhajeri, Junwei Luo, Scarlett Chen, Fahim Abdullah, Zhe Wu, and Panagiotis D Christofides. A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering*, page 107956, 2022.

[126] Stephen Piche, James Keeler, Greg Martin, Gene Boe, Doug Johnson, and Mark Gerules. Neural network based model predictive control. *Advances in neural information processing systems*, 12, 1999.

[127] David C Gordon, Alexander Winkler, Julian Bedei, Patrick Schaber, Jakob Andert, and Charles R Koch. Introducing a deep neural network-based model predictive control framework for rapid controller implementation. *arXiv preprint arXiv:2310.08392*, 2023.

[128] Seong Hyeon Hong, Jackson Cornelius, Yi Wang, and Kapil Pant. Optimized artificial neural network model and compensator in model predictive control for anomaly mitigation. *Journal of Dynamic Systems, Measurement, and Control*, 143(5):051005, 2021.

[129] Bailun Jiang, Boyang Li, Weifeng Zhou, Li-Yu Lo, Chih-Keng Chen, and Chih-Yung Wen. Neural network based model predictive control for a quadrotor UAV. *Aerospace*, 9(8):460, 2022.

[130] Daming Wang, Zheng John Shen, Xin Yin, Sai Tang, Xifei Liu, Chao Zhang, Jun Wang, Jose Rodriguez, and Margarita Norambuena. Model predictive control using artificial neural network for power converters. *IEEE Transactions on Industrial Electronics*, 69(4):3689–3699, 2021.

[131] Roja Eini and Sherif Abdelwahed. A neural network-based model predictive control approach for buildings comfort management. In *2020 IEEE International Smart Cities Conference (ISC2)*, pages 1–7. IEEE, 2020.

[132] Mariam Elnour, Yassine Himeur, Fodil Fadli, Hamdi Mohammedsherif, Nader Meskin, Ahmad M Ahmad, Ioan Petri, Yacine Rezgui, and Andrei Hodorog. Neural network-based model predictive control system for optimizing building automation and management systems of sports facilities. *Applied Energy*, 318:119153, 2022.

[133] Santosh Devasia et al. Output-sampled model predictive path integral control (o-mppi) for increased efficiency. *arXiv preprint arXiv:2309.13201*, 2023.