NORTHWESTERN UNIVERSITY

# Real-Time Safe Control for Model-Based and Data-Driven Robotics

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Mechanical Engineering

By

Giorgos Mamakoukas

EVANSTON, ILLINOIS

June 2021

## Abstract

Real-Time Safe Control for Model-Based and Data-Driven Robotics

Giorgos Mamakoukas

Robot dynamics are typically highly nonlinear and their control is a challenging research topic without a single overarching algorithm that demonstrates best performance. In this thesis, I derive a nonlinear control algorithm that uses an analytical, closed-form solution to compute feedback with controllability-based guarantees for convergence. The proposed algorithm, based on needle variations, is computationally more efficient than alternative top performing methods, while also *a priori* guaranteeing convergence for dynamics that are controllable with first-order Lie brackets. The performance of the proposed algorithm is demonstrated on various systems and tasks, including a differential drive robot avoiding moving obstacles while converging to the target and a 3D underactuated robotic fish model tracking a target in the presence of fluid drift. The results highlight the ability of the feedback scheme to reject disturbances and run in real time, rendering it an attractive candidate nonlinear controller.

As a second contribution, this thesis improves system identification methods via Koopman Operators. First, it introduces a generalizable methodology for data-driven identification of nonlinear dynamics using Koopman models that bound the model error in terms of the prediction horizon and the magnitude of the derivatives of the system states. The approach relies on synthesizing Koopman basis functions using the derivatives of general nonlinear dynamics that need not be known and can be computed numerically in real time. The error bounds are verified in different scenarios, including data-driven models learned from unknown dynamics of a pendulum with highly noisy measurements. The efficacy of the data-driven modeling approach is also validated with simulation and experimental results on the control of a tail-actuated robotic fish and is shown to outperform a well-tuned model-free PID (Proportional Integral Derivative) controller. When updated online, the data-driven model leads to significantly improved control performance in the presence of unmodeled fluid disturbance.

Second, to improve learning of unknown dynamics with little data, this thesis leverages side information (i.e., general knowledge we have about the properties of a system, besides training data) and presents a new algorithm that imposes stability on the learned representation. The proposed algorithm, which is provably more memory efficient than top competing methods and achieves orders of magnitude lower modeling error, makes the learned model robust to the amount of training data and improves both prediction accuracy and control performance. The benefits of stability-constrained data-driven models are demonstrated in simulation and experiment, including the tasks of stabilizing a quadrotor in free-fall and learning the dynamics of a pusher-slider system. Conditions under which the learned dynamics of a controlled system can lead to an online certificate for stabilizing controllers, via control Lyapunov function analysis, are also discussed.

## Acknowledgements

This thesis is the result of years of efforts. Along the way, there were a number of people who helped me, directly or indirectly, and I would like to acknowledge their contributions. First of all, to my adviser Todd Murphey, you have helped me understand that research itself is nonlinear (pun intended). You inspired me to think bold, strive to set high expectations, develop as an independent researcher, and become a better writer. To my committee member Randy Freeman, thank you for cultivating my interest in control theory through your courses and our personal conversations, as well as for your ability to identify the exact (often numerical) cause of 'impossible' results. To my committee member, Malcolm MacIver, thank you for your high attention to detail when reviewing our research papers.

I would also like to thank my external research collaborators for contributing to these results and making my research experience a great journey. To Xiaobo Tan, thank you for always salvaging meaningful results from our failures, and trying to re-frame them towards a positive contribution. It has helped me approach failure with a more welcoming mindset. To Maria Castaño, thank you for your hard work, patience, and willingness to stay up late to complete our experiments during my visits, even at times when we had to put out fires. I also want to thank my colleagues for sharing the same passion for research, offering me feedback on my papers and presentations, and being great friends. Special thanks to Emmanouil, Anastasia, Ian, Katie, and Gerardo, with whom I have worked more closely.

My family has been a indispensable part of this journey and supported me throughout the many bumps of this road. To my mother, thank you for all your wonderful support and for mitigating the time difference by staying up late to chat. To my father, thank you for making me laugh at the frustrating moments of research and life and instilling an optimistic outlook in me. To my siblings, close relatives, and friends, thank you for being there for me. To Nikos, a dear friend, thank you for being a great mentor for both academic and personal matters. Special thanks to Gerhard, a remarkable person, who has been nothing sort of a family in the states for me. I will always be grateful for all your help.

Last, I want to dedicate this thesis to my lovely grandmother, who passed away before I could finish it. I know how much you were anticipating my graduation and complaining that I always said ‘one more year to go’. I will always remember you.

# Table of Contents

# List of Tables

## List of Figures

CHAPTER 1

# Introduction

Dynamics of robotic systems are often unknown, high-dimensional, and highly nonlinear, making real-time control challenging [**5**]. Further, systems are underactuated either by design—in order to reduce actuator weight, expenses, or energy consumption—or as a result of technical failures. Such challenges can destabilize optimization schemes that either do not run in real time or rely heavily on knowing the exact model and environmental parameters. As a result, to be successful, feedback policies need to exploit the nonlinearities of the dynamics, be general enough for a broad class of systems, and be able to learn or adapt online to unmodeled changes [**6**], as well as balance model accuracy and computational efficiency. In this thesis, I present real-time algorithms that improve both system identification and control of robotic systems.

To control systems with known dynamics, one can draw from many schemes, including linear quadratic regulator (LQR) [**7**], linear model predictive control (LMPC) [**8**], nonlinear model predictive control (NMPC) [**9**], feedback linearization [**10**], differential dynamic programming (DDP) [**11**], and variants of the above [**12**, **13**]. Additional nonlinear control techniques include steering methods using sinusoid controls [**14**], sequential actions of Lie bracket sequences [**15**], backstepping [**16**, **17**], perturbation methods [**18**], sliding mode control (SMC) [**19**–**21**], intelligent control [**22**, **23**], and hybrid control [**24**]. The aforementioned methods have limitations. LQR and LMPC remain accurate only near the localization points; NMPC and DP are typically computationally expensive; backstepping is generally ineffective in the presence of control limits; perturbation methods assume a future of control decisions that do not take the disturbance history into account; SMC methods suffer from chattering, which results in high energy consumption and instability risks by virtue of exciting unmodeled high-frequency dynamics [**25**], intelligent control methods are subject to data uncertainties [**26**], while other methods (e.g. feedback linearization, steering methods) are often case-specific and will not hold for the level of generality encountered in robotics.

Underactuated systems that are subject to nonintegrable differential constraints, termed nonholonomic constraints, are particularly challenging in robotics [**27**]. Examples include wheeled agents that are not allowed to skid (e.g., unicycle, differential drive, tricycle). These systems are studied in terms of their controllability, which reveals all possible effects that combined control inputs can have. As a result, controllability analytically answers the existence of control solutions that can move a robot between arbitrary states in finite time and is particularly useful for underactuated systems that are subject to velocity, but not displacement, constraints.

A popular approach in controlling nonholonomic systems is piecewise constant motion planning [**28**, **29**]. Lafferriere and Sussmann [**29**, **30**] extend the original dynamics with *fictitious* action variables in the direction of the nested Lie brackets to determine a control for the extended system. They first compute the time the system must flow along each vector field, in a sequential manner, to accomplish a given motion of the extended system. Then, using the Campbell-Baker-Hausdorff-Dynkin (CBHD) formula [**31**–**33**], they recover the solution in terms of the original inputs of the system. On the other hand, piecewise constant motion planning is model-specific, since the process changes for different number of inputs. In addition, solutions involve a sequence of individual actions that generate the Lie bracket motion and the actuation sequence grows increasingly larger for higher order brackets. Compensating for the third-order error in the CBHD formula involves two second-order Lie brackets and twenty successive individual inputs, each of infinitesimal duration [**34**]. The sequence is described in detail by [**29**]. In practice, such actuation becomes challenging as the number of switches grows. The theoretically infinitesimal duration of each input may be hard to reproduce in hardware, while, in the face of uncertainty and time-evolving trajectories, actuation consisting of a large sequence of controls (e.g., of twenty actions) is likely to change once feedback is received.

Another popular approach is steering using sinusoids [**14**, **15**, **35**–**38**]. This method applies sinusoidal control inputs of integrally related frequencies. States are sequentially brought into the desired configuration in stages, while the rest of the states remain invariant over a single cycle. This approach has been validated in generating motion of an underactuated robot fish [**39**]. Steering using sinusoids suffers from the complicated sequence of actions that grows as a function of the inputs involved. Moreover, besides also being model-specific, the method addresses each state separately, meaning each state gets controlled by its own periodic motion, requiring $N$ periods for an $N$-dimensional system, leading to slow convergence. Further, solutions focus on the final states (at the end of each cycle) and not their time evolution, hence they may temporarily increase the running cost (consider the car example of Fig. 7 in [**14**]). As with the method of piecewise constant motion planning, when tracking a moving target, these factors also compromise the performance of this approach.

Other trajectory generation techniques for controllable systems involve differential flatness [**40**–**44**] and kinematic reduction [**45**–**47**]. Control based on differential flatness uses outputs and their derivatives to determine control laws. However, as discussed in [**48**], there is not an automatic procedure to discover whether flat outputs exist. Further, differential flatness does not apply to all controllable systems and motion planning is further complicated when control limits or obstacles are present [**45**].

Needle variation methods constitute a distinct approach to optimal control. Contrary to optimal control algorithms that try to minimize a local approximation of the cost function by iteratively searching in high-dimensional spaces, a computationally expensive process, needle variation methods aim to *reduce*, not minimize, the objective. They avoid the approximation of the value function, and instead exploit the time-evolving sensitivity of the objective to infinitesimal switched dynamics to optimally perturb the default trajectory. As a result, needle variation methods are less computationally expensive than methods such as iLQR and other algorithms that expand the cost function to second order [**49**, **50**]. For example, while first-order needle variation controls compute the evolution of the state and first-order costate equations via two $n \times 1$ differential equations in each iteration, the iLQR algorithm solves the Riccati equations to calculate a descent direction and, together with the simulation of the state, it computes in total three $n \times 1$ and one $n \times n$ differential equations.

Needle variation methods exist globally, demonstrate a large region of attraction [**2**, **51**], and have a less complicated representation on Lie Groups [**51**]. On the other hand, needle variation methods based on the mode insertion gradient have so far been implemented only as single-action feedback. As a result, control responses are discontinuous without filtering realizable by hardware. Additionally, single-action controllers exhibit poor performance near equilibrium and often switch to iLQR policies to ensure stability [**2**, **3**].

Sequential Action Control (SAC) [**2**] is a nonlinear controller based on needle variation theory. SAC uses an analytical closed-form solution to synthesize control actions that best improve the stated objective, contrary to other optimization schemes that iteratively calculate optimal control actions to *minimize* a cost function [**52**, **53**]. As a consequence, SAC obtain orders of magnitude improvements in computational speed over controllers such as iLQG [**13**], while achieving comparable performance with less control effort [**2**]. These traits render SAC well-suited for online tasks of high-dimensional and highly-nonlinear robots. On the other hand, SAC is challenging to implement in hardware because its feedback is discontinuous, it performs poorly near equilibrium, and can fail to find solutions even for single tasks (i.e., diagonal displacement of vehicle-like systems).

In the first part of this thesis, I address the shortcomings of SAC and leverage needle variation methods to create a computationally efficient nonlinear controller that has controllability-based convergence guarantees for known dynamics. First, in Chapter 3, I demonstrate empirically that SAC is able to reject unknown drift intensities and directions [**2**, **54**, **55**] and produces comparable trajectories to alternative trajectory optimization methods using less control effort and resulting in better objective cost. In Chapter 4, I establish a mathematical relationship between the solution of gradient descent and SAC. By proving that the mode insertion gradient provides a descent direction for the entire time horizon, I construct a continuous feedback synthesis scheme out of the single-action solution, referred to as Needle Variation Controller (NVC). The proposed algorithm is a generalization of all first-order control solutions and, under certain parameter choices, matches the solution of conjugate gradient descent. NVC relates SAC to gradient descent and, by showing that the entire curves of SAC solutions can be implemented for control, I allow the discontinuous, hard-to-implement SAC solutions to be applied as smooth feedback that is more actuator-friendly, and without losing the convergence guarantees. Further, by relating SAC to gradient descent, I bridge the gap from sequential action control to traditional control policies and show the potential for other feedback policies to be implemented in a single-action fashion. Using NVC, I successfully control a bipedal walker and show that the proposed controller readily adapts to hybrid systems. Without using gait specific parameters, NVC performs gait initialization, establishes a quasi-periodic gait cycle, and is robust to force disturbances.

Last, in Chapter 5, I augment SAC with second-order needle variations, termed mode insertion hessian (MIH), to overcome control singularities and improve the convergence rate. I relate the MIH expression to controllability analysis by revealing its underlying Lie bracket structure and present a second-order Sequential Action Controller that guarantees control solutions for the entire class of systems that are controllable with first-order Lie brackets. For classically studied systems, such as the differential drive cart, this amounts to being able to guarantee that the control approach is *globally* certain to provide descent at every state. As a consequence, provided that the objective is convex with respect to the state (in the unconstrained sense), second-order SAC provably guarantees that the robot reaches the target in a collision-free manner in the presence of (moving) obstacles without relying on predefined trajectories. I demonstrate the efficacy of the algorithm on various systems, including a differential drive vehicle and an underactuated dynamic model of an underwater vehicle. These studies conclude Part 1 of the thesis.

Robot dynamics are often unknown or stochastic (e.g., Sphero SPRK [**56**], soft robotics [**57**, **58**]) and environments that are complex or changing (such as sand [**59**–**61**] or water [**62**–**65**]) are hard to model accurately. In the face of such uncertainty, robotic applications often fail due to poor prediction and control. For this reason, system identification methods are used to develop or adapt a model from data [**66**–**71**].

Recent data-driven efforts in robotics have focused on Koopman operators [**72**]. Koopman operators are linear embeddings of nonlinear systems that evolve functions of the states without loss of accuracy [**73**–**76**]. In general, linear representations are often desirable because they admit closed-form solutions, simplify modeling, and are general enough to be useful in many applications (e.g. Kalman filters). Further, there are well-established tools for the analysis (e.g. investigating properties of a system, such as stability and dissipativity), prediction, estimation, and control of linear systems [**77**]. Further, linear dynamical systems can be learned in a self-supervised manner and naturally arise in many areas of machine learning and time series modeling with several active research applications, such as dynamic texture classification [**4**, **78**] and video action recognition [**79**, **80**].

In robotics, Koopman operators have gained attention for the purposes of both system identification [**81**] and real-time nonlinear control [**76**], as they can help address both the difficulty with nonlinearity and the need to incorporate data in the model [**81**–**83**]. The linear representation allows one to control the nonlinear system using tools from linear optimal control [**73**, **84**], which is often easier and faster to implement than nonlinear methods, thus enabling online feedback for high-dimensional nonlinear systems. Beyond the computational speed and the reduction in feedback complexity, the linear representation-based control could lead to better performance compared to a controller that is based on the original nonlinear system [**85**, **86**]. However, with few exceptions [**85**, **87**, **88**], Koopman operators are typically infinite-dimensional and studies seek finite-dimensional approximations to Koopman operators that still capture the dynamics with high fidelity using methods such as the Dynamic Mode Decomposition (DMD) [**89**], extended DMD (EDMD) [**82**, **90**], Hankel-DMD [**91**], or closed-form solutions [**92**, **93**].

In the trade-off between the dimensionality and the modeling accuracy of the linear representation, these studies face the challenge of finding the minimum number and choice of basis functions for the desired accuracy [**94**]. There has not been a systematic way to select basis functions for approximate Koopman models of general nonlinear systems; rather, most efforts rely on trial-and-error [**57**,**95**–**99**] and machine learning tools [**94**,**100**], or are system-specific [**81**]. Furthermore, there is no available method to analyze the predictive accuracy of the finite-dimensional Koopman operators for general nonlinear systems. In light of this, Chapter 8 presents a generalizable methodology for choosing Koopman basis functions and analyzes the model accuracy with error bounds [**101**]. The proposed method constructs the basis functions for the Koopman operator using higher-order derivatives of the nonlinear dynamics, which need not be known; only the derivatives of the tracked states must be available. The error bounds, which depend on the prediction time horizon and the magnitude of the derivatives, can be used to determine the basis functions for the desired level of model accuracy. This is the first work that selects basis functions using a systematic methodology and provides an error bound on the accuracy of a Koopman representation for general nonlinear dynamics. The proposed data-driven modeling approach is validated with simulation and experimental results on the control of a tail-actuated robotic fish. Updating the dynamical model in real time significantly improves the performance of Koopman-LQR in the presence of unknown fluid disturbance.

When learning representations from data, it is important to generate models that generalize beyond the training data and remain accurate for long predictions. To do so, data-driven models need to have the same properties (e.g. symmetry) as the underlying dynamics. However, when learning representations from data, side information and knowledge of the modeled dynamics have typically not been leveraged to improve the models. Koopman-based efforts, in particular, have relied on black-box optimization tools that return best-fitting models without regard for their properties. As a result, research efforts have overlooked whether the properties of the learned model are consistent with those of the original system. Stability is an especially important feature of linear representations. It describes the long-term behavior of a system and is critical both for numerical computations to converge and to accurately represent the true properties of many physical systems. However, it is often overlooked and the learned linear representation may be unstable even when the underlying system is stable [**102**]. For example, work in [**90**] makes assumptions on the stability of the underlying nonlinear dynamics—i.e., the system has a single attractor that is (asymptotically) stable—but does not enforce similar constraints on the learned model. As a result, stable nonlinear dynamics are sometimes represented by Koopman operators that are unstable, due to noise, poor quality (e.g., sparse or highly-correlated) measurements, or even limitations of the learning schemes used [**103**–**105**]. Needless to say, when the stability properties of the underlying system and the learned model do not match, the Koopman-based evolution of the states diverges exponentially from the true solution.

Chapter 9 presents an algorithm, called SOC, for learning stable LDSs for prediction and control. SOC is *provably* more memory efficient than competing alternatives, with an $\mathcal{O}(n^2)$ space complexity—$n$ being the state dimension—compared to $\mathcal{O}(n^4)$. In addition, SOC returns a stable LDS even after one single iteration, which can be crucial in online applications and time-sensitive tasks where obtaining a stable state-transition matrix as early in the optimization process as possible becomes of central importance. The performance of SOC is demonstrated on learning dynamic textures from videos, as well as learning and controlling (in simulation and experiment) the Franka Emika Panda robotic arm [**106**]. When compared to the current top-performing models, a constraint generation (CG) [**4**] and a weighted least squares (WLS) [**107**] approach, SOC achieves an orders-of-magnitude lower reconstruction error, robustness even in low-resource settings, and better control performance. Notably, SOC is the first that tests the control performance of stable LDS; CG has been formulated but not evaluated for control tasks and it is not straightforward that WLS can be implemented for such applications.

Chapter 10 uses the SOC algorithm for the data-driven identification of stable Koopman operators (DISKO) for the purposes of predictions that remain numerically stable and accurate over long time horizons as well as improving control performance, especially in the low-data limit. Specifically, it derives the prediction error induced by Koopman models over an arbitrary number of time steps (a result that is used to show the need for stable operators); it provides conditions for choosing Koopman basis functions that are consistent with the stability properties of the underlying nonlinear system and which can improve data-driven learning; and it presents a method to construct candidate control-Lyapunov functions for nonlinear dynamics, which are used to verify stabilizing controllers. The benefits of DISKO are demonstrated with simulation and experimental results on various systems, including a quadrotor in free-fall and a pusher-slider system.

# Part 1

# Model-Based Control with Controllability-Based Convergence Guarantees

CHAPTER 2

# Background

This chapter reviews Sequential Action Control (SAC), which is the pillar upon which Part 1 of this thesis is developed. SAC is a model-based controller for control-affine nonlinear systems that is based on the theory of needle variations. Chapter 3 demonstrates empirically the ability of SAC for disturbance rejection and Chapter 4 relates SAC to Gradient Descent. Last, Chapters 5 and 6 augment SAC with second-order needle variations and provide explicit controllability-based guarantees for convergence and collision avoidance.

## 2.1. Needle Variation Control Methods

Consider a system with state $x : \mathbb{R} \mapsto \mathbb{R}^N$ and control $u : \mathbb{R} \mapsto \mathbb{R}^{M \times 1}$ with control-affine dynamics of the form

$$\dot{x}(t) = f(x(t), u(t)) = g(x(t)) + h(x(t))u(t), \tag{2.1}$$

where $g(x(t))$ is the drift vector field. Needle variation methods consider a time period $[t_o, t_f]$ and switched control modes described by

$$\dot{x}(t) = \begin{cases} f_1, & t_o \leq t < \tau - \frac{\lambda}{2} \\ f_2, & \tau - \frac{\lambda}{2} \leq t < \tau + \frac{\lambda}{2} \\ f_1, & \tau + \frac{\lambda}{2} \leq t \leq t_f, \end{cases} \tag{2.2}$$

where $f_1$ and $f_2$ are dynamics associated with *default* and *inserted* control $v(t)$ and $u(t)$, respectively, and defined as

$$f_1 \triangleq f(x(t), v(t))$$
$$f_2 \triangleq f(x(t), u(\tau)).$$

Parameters $\tau$ and $\lambda$ are the switching time between the two modes and the (infinitesimal) duration of the inserted dynamics $f_2$. Note that the default control $v(t)$ is the actuation for the nominal trajectory—$v(t)$ could itself be the result of a different controller—which is then improved by the insertion of a new control vector $u(t)$ creating a switched mode $f_2$. In addition, while the default control $v(t)$ of the switched mode sequence in (2) may be time-dependent, the dynamics $f_2$ have control $u(\tau)$ that has a fixed value over $[\tau - \frac{\lambda}{2}, \tau + \frac{\lambda}{2}]$. Dynamics of the form (2.2) appear in optimal control of hybrid systems to optimize the time scheduling of *a priori* known modes [**108**], but single-action feedback schemes use them to obtain a new control mode that will optimally perturb the trajectory of any type of system with a needle action.

Needle variation methods consider objectives that have typically been control-independent of the form[1]

$$J(s(t)) = \int_{t_o}^{t_f} \ell(s(t))\,\mathrm{d}t + m(s(t_f)). \tag{2.3}$$

and use the first-order sensitivity of the cost function to infinitesimal applications of inserted control (called the mode insertion gradient (MIG) in the hybrid systems literature [**110**, **111**]). For simplicity, the arguments are dropped as necessary in the following analysis. The mode insertion gradient for control-independent costs, derived in [**108**], is

$$\frac{dJ}{d\lambda_+} = \rho^T(f_2 - f_1), \tag{2.4}$$

where the elements of $\rho : \mathbb{R} \mapsto \mathbb{R}^N$ are the first-order adjoint states (costates), which are calculated from the default trajectory via

$$\begin{aligned} \dot{\rho} &= -D_s\ell^T - D_s f_1^T \rho \\ \text{subject to: } \rho(t_f) \;&=\; D_s m(s(t_f))^T. \end{aligned} \tag{2.5}$$

The subscript $\lambda_+$ indicates that derivative is considered after evaluating the limit $\lambda \to 0$. Note that, in practice, the applied control will have finite duration. Due to continuity, however, the first-order sensitivity of the cost to inserted control will be similar to the

[1]While the objective for needle variation controls has typically not included a control term, doing so is straightforward and yields similar performance. Work in [**2**, **109**] has considered objectives with control terms, and one can recompute the mode insertion gradient and mode insertion Hessian assuming the objective depends on $u$ [**109**] without impacting any of the rest of the approach.

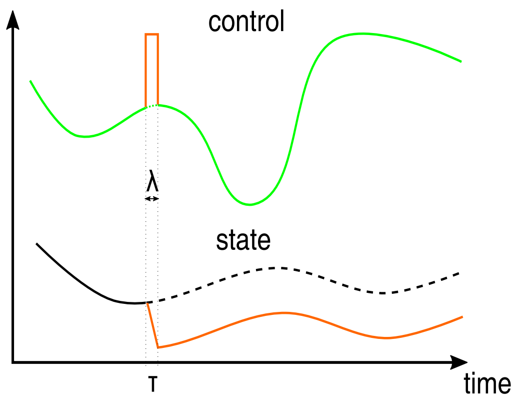


Figure 2.1. A fixed-value perturbation in the nominal control, introduced at time $\tau$ and with duration $\lambda$, and the associated variation in the state. In the limit $\lambda \to 0$, the control perturbation becomes a needle variation.

MIG value (e.g., remain negative, if the MIG (2.4) is negative) in a neighborhood around $\lambda = 0$. For more details, refer to [**2**].

## 2.2. Sequential Action Control (SAC)

SAC, introduced in [**2**], is a needle-variation based closed-loop control algorithm. It considers two modes, $v(t)$ and $u(\tau)$, associated with default and optimal dynamics $f_1$ and $f_2$, respectively and, over the course of each horizon $[t_o, t_f]$, it injects optimal dynamics $f_2$ for infinitesimal duration. That is, the algorithm switches from the default mode $f_1$ to the optimal action mode $f_2$ and back to $f_1$.

More specifically, during each cycle SAC forward simulates the dynamics of the system along a user-specified time horizon $T$. Then, it uses the MIG to compute a closed-form analytical expression of an optimal action that minimizes the MIG (not the objective function) with respect to an infinitesimal application of $f_2$, that is

$$\begin{aligned} u^*(t) =&\min_{u(t)} \ \frac{1}{2}(\frac{dJ}{d\lambda^+}(t) - \alpha_d)^2 + \frac{1}{2}\|u(t)\|_R^2 \\ =&(\Lambda + R^T)^{-1}[\Lambda u_1(t) + \alpha_d h(t)^T \rho(t)], \end{aligned} \tag{2.6}$$

where $\Lambda \triangleq h(t)^T \rho(t) \rho(t)^T h(t) \succeq 0$ and $\alpha_d \in \mathbb{R}^-$ expresses the (desired) first-order cost sensitivity (MIG) to the injected dynamics. Typically, $\alpha_d = -\gamma J$ where $\gamma \in \mathbb{R}^+$. Although $\alpha_d$ changes across iterations, it remains constant throughout $[t_o, t_f]$ of any individual control update.

The optimal control curve $u^*(t)$ in (2.6) returns the optimal control value as a function of time and enables SAC to compute the most effective time to act, i.e., one that minimizes the MIG while penalizing also control effort and the cost of waiting:

$$\tau = \min_t \ \|u(t)\| + \frac{dJ}{d\lambda_i^+}(t) + (t - t_o)^\beta. \tag{2.7}$$

Last, SAC considers finite controls that could improve the objective more than infinitesimal actions. Starting with an initial finite duration $\lambda$ centered at $\tau$, SAC iteratively reduces the duration via a backtracking line search until the objective improvement is above a specified value. SAC controls exist and are unique, as they are solutions to Tikhonov regularization problems [**2**]. A visual overview of SAC is presented in Fig. 2.2. The algorithm successively, every $t_s$ seconds, performs the set of computations presented in Algorithm I.

The results in [**2**] show that SAC is promising for on-line optimization problems. Additionally, SAC is computationally very efficient, as it computes an analytical solution

[2] $t_{curr} = i \times t_s$

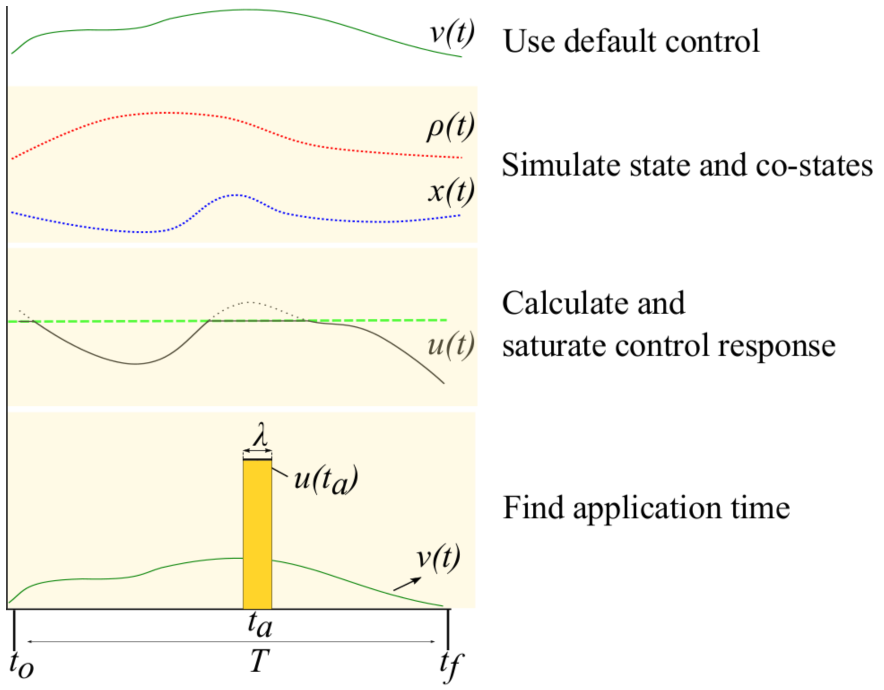


Figure 2.2. The algorithmic steps of SAC. Using default control, the state and costate variables are forward simulated in time. The optimal control response is computed from a closed-form analytical expression and saturated. In the last step, the application time of a single inserted action is chosen to correspond to a negative MIG.

**Algorithm I** Sequential Action Control

```
1: Simulate dynamics $f_1$ for $t \in (t_{curr}, t_{curr} + T)^2$
2: Compute initial tracking cost $J_{init,i}$ from equation (2.3)
3: Analytically compute optimal control curve $u^*(t)$
4: Search for optimal time $\tau_i$ to enact infinitesimal control $u(\tau_i)$ from equation (2.7)
5: Saturate control
6: Perform line search to specify control duration $\lambda_i$, centered at $\tau_i : (\tau_i - \frac{\lambda_i}{2}, \tau_i + \frac{\lambda_i}{2})$
```

to a non-linear optimal control problem. As a result, it avoids the large computational cost involved in solving the $\frac{n \times n + n}{2}$ Riccati differential equations used by open-loop optimal control approaches for $\mathbb{R}^n$-state systems. Further, it readily imposes control constraints, and can avoid local solutions at which SQP algorithms stop prematurely (see [**2**]).

The computational cost for systems with multidimensional state and control space renders some optimization methods too slow to incorporate feedback and perform in real time. The speed of the algorithm and its ability to scale better with respect to state and control dimensions are the reasons to consider SAC appropriate for real-time applications.

CHAPTER 3

# Empirical Disturbance Rejection of SAC for Unmodeled Fluid Flow

This chapter uses Sequential Action Control (SAC) for fast trajectory-tracking tasks in the presence of fluid drift. Through the benchmark example of the dynamic car, it is shown that SAC outperforms a traditional offline projection - based optimization technique in terms of control effort and objective cost. Motivated by recent work on effort-efficient, sight-independent weakly electric fish, this chapter also shows that SAC successfully provides control-optimal dynamics to perform short-range underwater maneuvers. Simulation results highlight SAC's empirical robustness to different drift intensities and added mass properties, even when the effective fluid drift is not included in the controller's model.

## 3.1. Simulation Results

In this chapter, I investigate applications of SAC on systems with drift. The goal is to illustrate the algorithm's ability to track trajectories:

– in non-dynamic/fluid environments

– underwater, in the presence of drift

– using dynamics with added mass in fluid.

The systems I consider are the 2D model of the dynamic car and an underwater model with the same dynamics and added mass parameters from the electric fish (dynamic fish-robot).

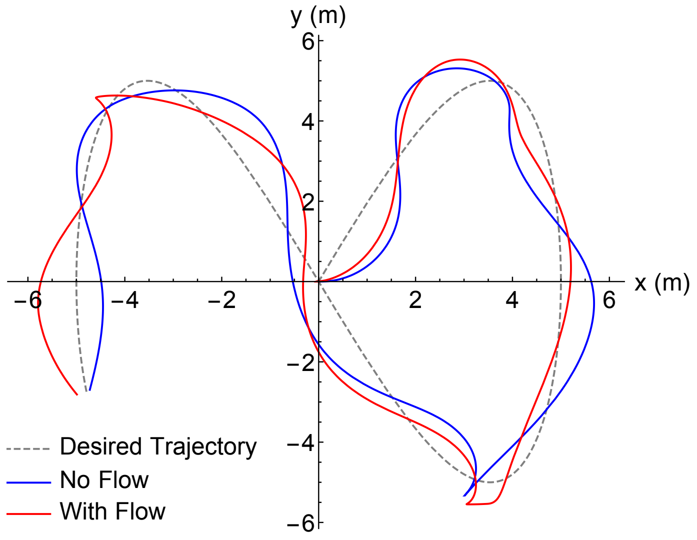


Figure 3.1. A parametric plot of SAC-computed trajectories on tracking the desired trajectory (dotted line) using the dynamic car dynamics at a control sequencing frequency of 20 Hz. The performance of the control is tested against no drift and drift of -1 m/s $\hat{x}$. Although the time horizon used for the simulations is extremely short (T = 1 s), the performance of SAC remains largely unaffected by the presence of flow.

These examples showcase SAC's general ability to track trajectories in environments with fluid flow and, more specifically, control the kinematics of the actual systems. The kinematic model of the car tests the general trajectory - tracking ability of SAC in fluid environments. Application on the dynamic fish-robot serves as a stepping stone for ultimately testing SAC on a robotic fish system [**112**, **113**].

Both examples present the same trajectory - tracking task with and without fluid drift. The results of the two cases are compared to show the effect of fluid environments on SAC. Underwater dynamics are simplified to constant fluid velocity drift in the system's state. To model the effect of drift, different intensities of fluid flow are used in both systems.

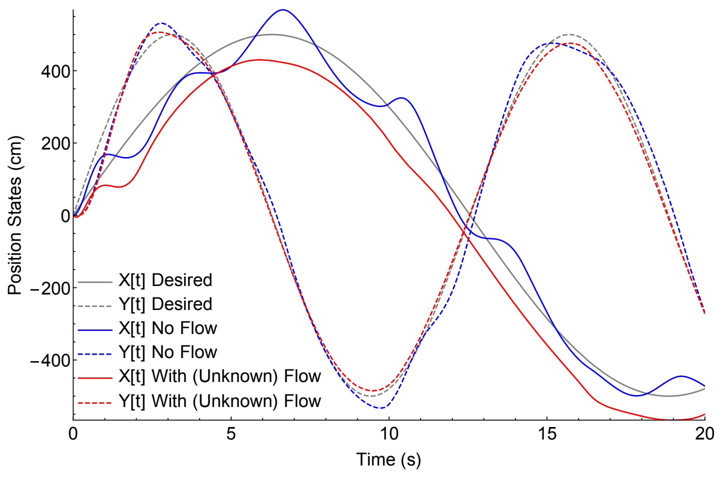


Figure 3.2. The SAC algorithm tests the dynamic car system on tracking the reference (gray) signal in two ways. On the first (blue) run, SAC is tested in a non-fluid environment and on the second run (red) in a fluid environment with -1 m/s $\hat{x}$ flow. The simulation uses non-fluid dynamics (absence of drift effects) in both cases–that is, the controller does not know there is fluid drift on the second run. Dotted lines show the x-coordinates and solid ones the y- ones. The reference signal is marked with gray, the neutral-environment test with blue and the underwater one with red.

### 3.1.1. Dynamic Car

The dynamic car is a well studied example often used in the literature to measure tracking performance of optimization algorithms [**114**, **115**]. In this subsection, SAC is tested on the 2D underactuated model of the dynamic car with state $s = (x, y, v, \theta, \dot{\theta})^T$, where $v$ is the forward velocity of the car in the body frame, and control input $u = (u_D, u_T)^T$ –

drive and turn, respectively. The dynamics are modeled as

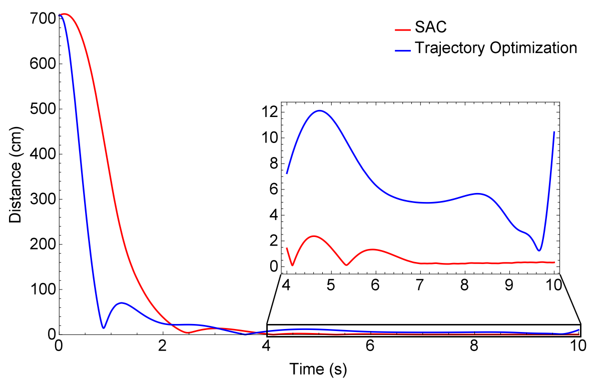
$$f(s,u) = \begin{pmatrix} v \cdot cos\theta + \dot{x}_w \\ v \cdot sin\theta + \dot{y}_w \\ u_D - \eta_1 \cdot v \\ \dot{\theta} \\ u_T - \eta_2 \cdot \dot{\theta} \end{pmatrix}, \tag{3.1}$$

where $\dot{x}_w$ and $\dot{y}_w$ represents the fluid drift in the x- and y- world frame axes and $\eta_1 =$ 0.01 1/s and $\eta_2 = 0.03$ 1/s represent linear and rotational damping coefficients. In this example, SAC is applied to track the following desired trajectory:

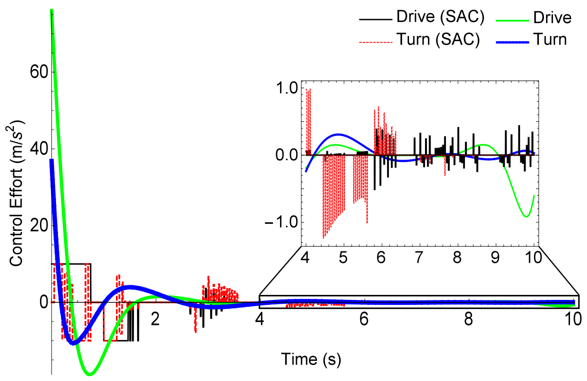
$$s_d(t) = (5 \cdot sin\frac{t}{4},\ 5 \cdot sin\frac{t}{2},\ 0, \frac{\pi}{2},\ 0)^T. \tag{3.2}$$

The parameters used in the simulations are $t_s$=0.05 s, $T$=1 s,$\dot{x}_w$ = -1.0 m/s, $\dot{y}_w = 0$, $Q$=**Diag**[100,100,1/1000,0,1/100], $P_1$=**Diag**[0, 0, 0, 0, 0], $R$ = **Diag**$[10^{-8}, 10^{-8}]$. The system starts from initial conditions $s_0 = (0,0,0,0,0)^T$ and the control remains constrained within the following limits: ($u_D \in [-10, +10]$ m/s$^2$, $u_T \in [-30, +30]$ rad/s$^2$).

The results of the simulation are presented in Fig. 3.1. The dynamic car stays within a couple centimeters of the desired trajectory, both in the absence and presence of fluid drift. In this benchmark example, fluid drift does not have a significant effect on tracking performance, despite the short time horizon used. Even though fluid dynamics have been simplified with constant velocity drift, SAC stays largely unaffected in its performance and is able to turn and drive the car to follow this changing track. The algorithm is also



(a) Tracking error performance of SAC and Trajectory Optimization on the task of reaching a stationary nearby target.



(b) Controls produced by SAC and Trajectory Optimization on the task of reaching a stationary nearby target.

Figure 3.3. SAC and the projection-based trajectory optimization scheme are tested on reaching a nearby target at (x,y) = (5 m, 5 m), starting from $s_0 = [0, 0, 0, 0, 0]$ and using the dynamic car dynamics in the presence of a -1.0 m/s $\hat{x}$ drift. As shown in the left figure, SAC satisfies saturation limits and exhibits better station keeping performance by remaining closer to the target (zoomed image). SAC outperforms Trajectory Optimization also in terms of the control efforts, which are measured by integrating control actions over application time: $\int_{t_0}^{t_f} u(t)dt$. Throughout the ten seconds of simulation, SAC uses 26.71 m/s$^2\cdot$ s and Trajectory Optimization uses 41.35 m/s$^2\cdot$ s. After the first two seconds, the integrated errors are 26.3 cm and 65.6 cm and the controls used are 3.63 m/s$^2\cdot$ s and 4.41 m/s$^2\cdot$ s for SAC and Trajectory Optimization, respectively. Control saturations used in SAC keep controls below 10 m/s$^2$, better resembling experimental actuation constraints.

compared to a projection-based trajectory optimization method on the task of reaching a nearby location. As Fig. 3.3 shows, SAC is able to decrease the objective (distance to the target) with less control and better final error, despite the saturation limits applied on control.

The ability of SAC to perform well with fluid drift inspired further work. Specifically, the authors tested whether SAC could apply optimal corrective control, without knowledge of the existing drift effects. In simulation, SAC is tested in the context of two dynamics, the non-fluid $f_{nf}$ and the fluid $f_{real}$. Dynamics $f_{nf}$ are the dynamics of the car $f$ in the absence of drift ($\dot{x} = \dot{y} = 0$), whereas dynamics $f_{real}$ are the actual dynamics of the environment ($\dot{x} = -1.0\ m/s, \dot{y} = 0$). In this way, SAC performs its computations, and applies control using $f_{nf}$. The actual progression of the system states, however, occurs using $f_{real}$ and the control that is computed for $f_{nf}$. The results presented in Fig. 3.2 show that SAC performance does not significantly deteriorate "underwater" without knowledge of the true fluid dynamics. The controller tracks the desired y-state accurately, while the x-state is only slightly off throughout the simulation. As the figure shows, the car oscillates back and forth over the reference target in the non-fluid test (blue curves) due to control overshoot. This effect arises because of the very short time horizon that does not allow the controller to know the desired trajectory well in advance. A second simulation with twice the time horizon duration ($T = 2$ s) showed reductions in the oscillations around these reference x-state. Yet the results of Fig. 3.2 describe a more realistic scenario, in which the controller has limited information about the controller's future motion and becomes more reactive than predictive.

### 3.1.2. Dynamic Fish-Robot

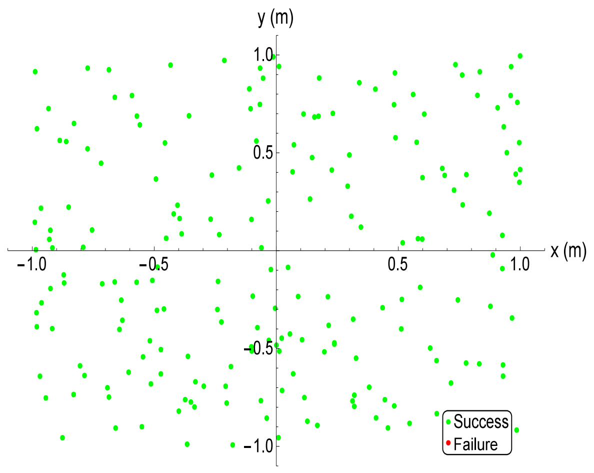
Through the benchmark example of the dynamic car, SAC is shown to be capable of trajectory - tracking in the presence of fluid drift, whether or not it has any knowledge of the actual fluid dynamics. Its computational speed and empirical disturbance rejection (evident in the results presented) are reasons to believe SAC can appropriately control underwater vehicles in real-time. Fish in general, and the weakly electric fish more specifically, are promising candidate system for underwater dynamical models in cluttered, dirty environments [**1**, **112**, **116**–**118**]. For this reason, this is the second system on which SAC is tested.

The weakly electric fish black ghost knifefish *Apteronotus albifrons* lives in dirty, turbulent waters and provides science with an example of optimal underwater motion [**1**, **117**]. Its ability to navigate has already been studied and shown to be optimal in prey-tracking scenarios [**1**], providing a potentially useful design model for future AUVs.

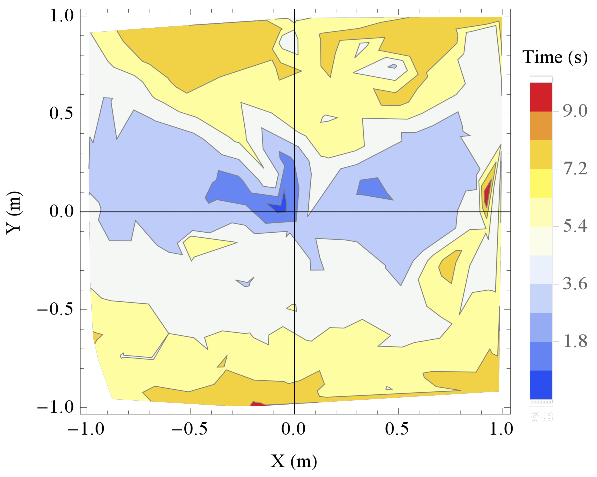
Its ability to navigate in turbid water sensing by way of a self-generated electric field drew the attention of the scientific world which saw a model of how to extend underwater expeditions to low-visibility areas. As described in [**119**] and [**120**], the weakly electric fish use active electrosense to map its surroundings and catch its prey. It continually generates an oscillating electric field and, through thousands of electric sensors placed throughout its body, can sense the presence of objects around it. Objects that do not share the same conductivity as water and perturb the self-generated electric field of the fish cause voltage changes at the sensors that are processed by the animal.

The dynamics of the underactuated electric fish are derived from Euler-Lagrange (EL) equations equivalent to Kirchhoff's equations [**1**]. With state $s = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})^T$ expressed in the world frame, and control input $u = (u_D, u_T)^T$ – drive and turn, respectively – in the body frame, the model incorporates damping and control input in the body - frame and fluid drift in the world frame. Specifically, the fish is modeled as a rigid body with a generalized inertia matrix $I = \mathbf{Diag}[m_1, m_2, m_3, j_1, j_2, j_3]$ and body-frame velocity $V_b$ given in terms of $G_{wb}(x, y, \theta)$ (the transformation from the world to the body - frame) and $R(\theta)$ (the rotation matrix):[1]

[1]The $^{\vee}$ operation on a 4x4 matrix is defined as $G^{\vee} = (G_{14}, G_{24}, G_{34}, G_{32}, G_{13}, G_{21})$, with $G_{ij}$ defining the element of G in the i-th row, j-th column; $\dot{p} = (\dot{x}, \dot{y}, \dot{z})^T$.



(a) A success/failure map of nearby targets for the dynamic fish-robot. All targets are successfully reached within ten seconds of simulation time.

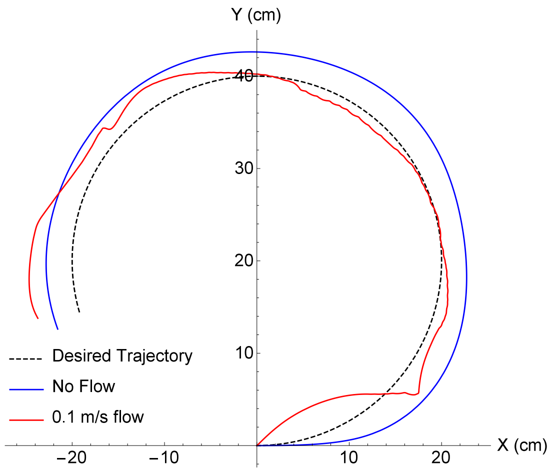


(b) Target locations are color-coded based on the simulation time it takes the dynamic robot-fish to reach them. Targets lying ahead ($+\hat{x}$) or behind (-$\hat{x}$) the system are reached the fastest. The asymmetry around the y-axis origin is due to the +0.1 m/s $\hat{y}$ drift.

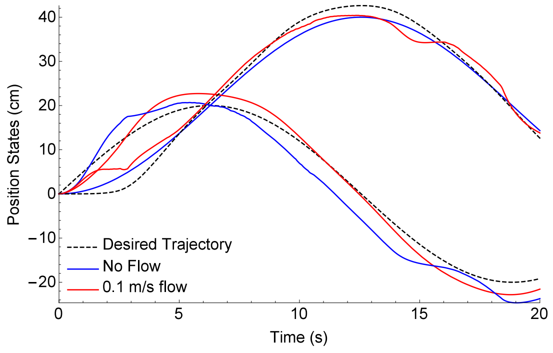
Figure 3.4. A map of target locations posed to the dynamic robot-fish system in the presence of +0.1 m/s $\hat{y}$ drift. Two hundred targets are randomly generated from a sample space of (x, y) = (1 m, 1 m) using Monte Carlo sampling. Success is defined by whether the robot-fish, always starting from an initial state of $s_0 = [0, 0, 0, 0, 0, 0]^T$, is at the end of the simulation (10 seconds) within 2 cm of the target, equal to half the longest dimension of the electric fish [**1**]. The only concern of the task is to approach the nearby targets and so zero weight is applied on the orientation $\theta$ of the system.

$$V_b = (G_{wb}^{-1} \cdot \dot{G}_{wb})^{\vee} = \begin{pmatrix} R^T \dot{p} \\ \omega \end{pmatrix},$$

$$R(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad G_{wb}(x, y, \theta) = \begin{pmatrix} & & & x \\ & R & & y \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$



(a) A parametric plot of SAC-produced trajectories.



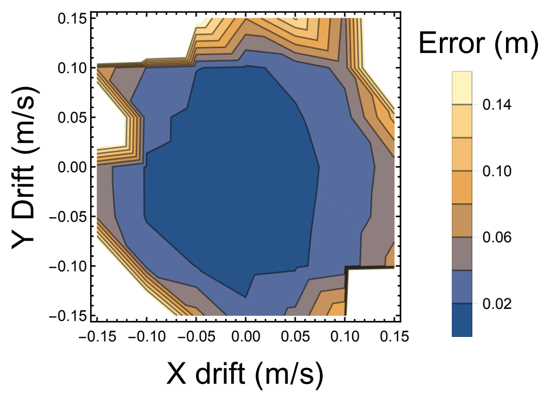
(b) Tracking error in the x-y states.

Figure 3.5. SAC is applied on the dynamic fish-robot model to track the desired trajectory at a control sequencing frequency of 20 Hz. The performance of the controller is tested against no drift and drift of +0.1 m/s $\hat{y}$. The computed trajectories are plotted against the reference signal. Although the time horizon used for the simulations is extremely short (T = 1 s), the performance of SAC remains largely unaffected by the presence of flow.

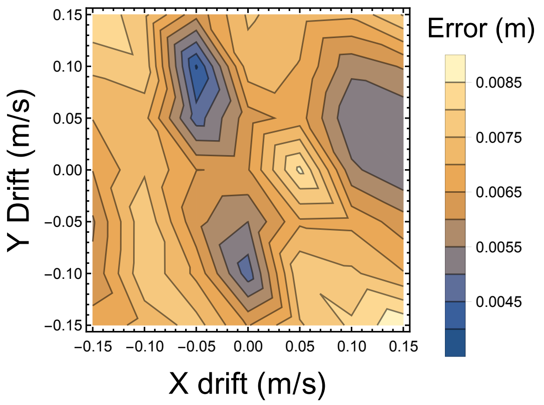
The generalized inertia matrix $I$ uses information about the physical limitations of motion and restorative forces in underwater environment. Parameters $m_1, m_2, m_3$ describe the added mass matrix (due to the volume of fluid accelerated by translations of the fish) and parameters $j_1, j_2, j_3$ refer to the added moment of inertia matrix (due to the volume of fluid accelerated by rotations). The values used are $(m_1, m_2, m_3) = (6.04, 17.31, 8.39)$ g, $(j_1, j_2, j_3) = (1.57, 27.78, 54.11)$ g cm$^2$ found in [**1**].

Assuming the body lies on a 2D plane, its potential energy (PE) is constant and its KE $= \frac{1}{2} V_b^T I V_b$ (invariant across transformations). The Lagrangian of the system is L $\triangleq$ KE - PE, the EL differential equations are:

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt}\frac{\partial L}{\partial \dot{s}_i} = F_{ext}, \quad \text{for } i = 1, 2, 3, \tag{3.3a}$$



(a) Tracking error throughout simulation. The white region corresponds to large errors, not visible in the bar legend.



(b) Steady-state tracking error, as measured after the first five seconds of simulation.

Figure 3.6. A contour plot on the effect of fluid drift intensity on trajectory - tracking performance for the dynamic system. The maps plot performance error as a function of fluid drift intensity (in both the x- and y- direction) and are generated from interpolating data for drift ∈ (-0.15, 0.15) m/s sampled in steps of 0.05 m/s. Performance error is calculated as the integrated distance (in m) away from the desired trajectory throughout the simulation period (20 seconds). The majority of the error occurs in the first five seconds, until the controller catches up with the target. The right figure shows the error between 5-20 seconds. The desired trajectory is provided in 3.4 and has a total arc length of 1.52 m over the simulation period.

where

$$F_{ext} = R(\theta) \cdot \begin{pmatrix} u_D + D_1 \\ D_2 \\ u_T + D_3 \end{pmatrix}, \tag{3.3b}$$

where $D_1, D_2, D_3$ are the damping forces. The rotation matrix in equation (3.3) is used to convert the forces of damping and control in the body-frame. Solving equation (3.3) yields $\ddot{x}, \ddot{y}, \ddot{\theta}$. Water drift $\dot{x}_w, \dot{y}_w$ is added to the measured state - velocities $\dot{x}, \dot{y}$.

First, SAC is tested on reaching nearby targets in the presence of +0.1 m/s $\hat{y}$ drift. The results, presented in Fig. 3.4, show that SAC reaches all randomized targets. These results highlight the maneuverability of the dynamic robot-fish dynamics, which SAC can efficiently handle. The parameters used in the simulation are $t_s = 0.05$ s, $T = 1$ s, $Q = \textbf{Diag}[1000,\ 1000,\ 0,\ 0.01,\ 0.01,\ 0.01]$, $P_1 = \textbf{Diag}[10000,\ 10000,\ 0,\ 1,\ 1,\ 1]$, $R = \textbf{Diag}[10^{-3}, 10^{-6}]$. SAC is further applied on tracking the following desired trajectory for 20 seconds, with and without flow:

$$s_d(t) = (0.2 \cdot \cos[\frac{t}{4} - \frac{\pi}{2}], 0.2 + 0.2 \cdot \sin[\frac{t}{4} - \frac{\pi}{2}], 0, 0, 0, 0)^T.$$

Starting from $s_0 = (0,\ 0,\ 0,\ 0,\ 0,\ 0)^T$, SAC uses $t_s = 0.05$ s, $T = 1$ s, $Q = \textbf{Diag}[1000,$ $1000,\ 0,\ 0.01,\ 0.01,\ 0.01]$, $P_1 = \textbf{Diag}[10000,\ 10000,\ 0,\ 1,\ 1,\ 1]$, $R = \textbf{Diag}[10^{-3}, 10^{-6}]$, and saturation constraints of 1 N on both control inputs. Simulation results are presented in Fig. 3.5. Not only does SAC successfully track the desired trajectory using the electric fish dynamics, but it also does reasonably well in the presence of fluid drift. The resulting underwater motion (with drift) is less smooth compared to the motion without drift, but is not worse in terms of the objective which is tracking the reference signal. Last, SAC is tested on the following trajectory - tracking task for a set of different fluid flow direction and intensities:

$$s_d(t) = (0.2 \cdot \sin\frac{t}{4},\ 0.2 \cdot \sin\frac{t}{2},\ 0,\ 0,\ 0,\ 0)^T. \tag{3.4}$$

Results are presented in Fig. 3.6 and show that SAC's ability to track the desired trajectory is empirically robust to different fluid intensities. While these simulation results cannot guarantee SAC's success tracking trajectories underwater in general, they suggest that

SAC can be used for underwater tracking using the model of the knifefish and provides a promising basis for further exploration.

## 3.2. Discussion

This chapter presents the ability of SAC to perform trajectory-tracking tasks in the presence of fluid drift. Because underwater environments are difficult to model accurately, several optimization schemes involve approximations in their models. Due to being offline or having a high computational cost, such schemes do not seem good candidates for real-time problems. On the other hand, the application results of this chapter show SAC to be a control-efficient solution that empirically rejects fluid drift disturbance, without sacrificing tracking performance. Hence, SAC seems a reasonable alternative for underwater trajectory-tracking. On the other hand, Sequential Action Control generates discontinuous feedback that may be challenging (or infeasible) for actuators. In Chapter 4, I show how one can use the solution of Sequential Action Control to generate smooth feedback that can provably decrease the cost function.

CHAPTER 4

# Relationship of SAC to Gradient Descent and Smooth Control Synthesis

This chapter proves that the mode insertion gradient (MIG), traditionally used in hybrid systems for optimal mode scheduling, provides a descent direction over the entire time horizon. This result allows us to construct first-order needle-variation based controllers for general nonlinear systems that synthesize continuous control responses from the MIG, maintaining the benefits of needle variation methods, such as high computational efficiency and a large region of attraction. The proposed algorithm of needle-variation based continuous (NVC) feedback is a generalization of all possible first-order algorithms and, for specific parameters, it is shown to match the conjugate gradient descent algorithm. As a second result of this chapter, the proposed control solution guarantees descent even when subject to arbitrary scaling (that can differ both across different inputs and over time) as long as it maintains the sign of each input. The benefits of NVC are showcased in simulations of a three-link biped that demonstrates successful gait initiation, walking, and disturbance tolerance.

## 4.1. Descent Direction of Needle-Variation-Based Continuous Control

In hybrid systems literature, the mode insertion gradient (MIG) is the first-order sensitivity of the objective to an infinitesimal perturbation of the default control. Feedback controllers that use needle variation methods, such as Sequential Action Control [**2**], construct control responses that correspond to a negative MIG. The resulting solution is itself a function of time and returns the optimal infinitesimal input perturbation at an application time. In this chapter, I show that the mode insertion gradient, and by extension the solutions of first-order single-action controllers [**2**,**3**], is a descent direction for the control over the entire time horizon. As a result, one can apply the control curve and avoid the sparse implementation of single-action feedback schemes. This is an alternative implementation that updates all the control values over the time horizon, similar to linear quadratic methods, but with less computation. The analysis of this chapter rests on definitions and assumptions that are defined next.

### 4.1.1. Assumptions

**Definition 1.** *A trajectory that is the local minimizer is given by a pair* $(x^*, u^*)$ *if and only if the pair satisfies the optimality conditions posed by Pontryagin's Maximum principle:*

$$\frac{\partial H}{\partial \rho} = \frac{d}{dt}s(t) \quad \frac{\partial H}{\partial s} = -\frac{d}{dt}\rho(t) \quad \frac{\partial H}{\partial u} = 0, \tag{4.1}$$

*where* $H$*, the Hamiltonian function, is defined as*

$$H \triangleq L(s(t), u(t), t) + \rho^T(t) f(s(t), u(t), t).$$

**Assumption 1.** *The vector elements of dynamics* $f_1$ *and* $f_2$ *are real, bounded,* $\mathcal{C}^1$ *in* $x$*, and* $\mathcal{C}^0$ *in* $u$ *and* $t$*.*

**Assumption 2.** *The incremental cost $\ell(\cdot)$ and the terminal cost $m(\cdot)$ are real, bounded, and $\mathcal{C}^1$ in $x$.*

**Assumption 3.** *Default and inserted controls $v$ and $u$ are real, bounded, and $\mathcal{C}^0$ in $t$.*

There are two variants of single-action policies, SAC and iterative SAC (iSAC), with slightly different control policies. In the analysis that follows, I examine the properties of both algorithms.

### 4.1.2. Feedback Policy of Sequential Action Control

Let $v(t)$ be the default control and $w(t)$ be the control perturbation vector, such that

$$u(t) = v(t) + w(t). \tag{4.2}$$

For ease of comparison, I also express the SAC solution in terms of the nominal vector and a perturbation,

$$\begin{aligned} u(t) =& \underset{u}{\operatorname{argmin}} \ \frac{1}{2}(\frac{dJ}{d\lambda_+} - \alpha_d)^2 + \frac{1}{2}\|u\|_R^2 \\ =& v(t) - (\Lambda + R)^{-1}(Rv(t) - \alpha_d h^T \rho(t)), \end{aligned}$$

such that

$$w(t) = -(\Lambda + R)^{-1}(Rv(t) - \alpha_d h^T \rho(t)) \tag{4.3}$$

### 4.1.3. Feedback Policy of Iterative Sequential Action Control

To show that the control solution of Sequential Action Control is a descent direction over the entire time horizon, I use the slightly modified control solution of iterative Sequential Action Control (iSAC) [**3**], a variation of SAC with conditions on stability.

Work in [**3**] uses $\alpha_d$ as the desired sensitivity and computes controls given by

$$\begin{aligned} u(t) &= \underset{u}{\operatorname{argmin}} \frac{1}{2}(\frac{dJ}{d\lambda_+} - \alpha_d)^2 + \frac{1}{2}\|u - v\|_R^2 \\ &= v(t) + \alpha_d(\Lambda + R)^{-1}h^T(t)\rho(t), \end{aligned} \tag{4.4}$$

where $\Lambda \triangleq h(t)^T\rho(t)\rho(t)^T h(t) \succeq 0$ as in (2.6). The first term in (4.4) drives the MIG to a desired negative sensitivity ($\alpha_d$), while the second term penalizes large deviations from the nominal control. Given (4.2), the control update becomes

$$w(t) = \alpha_d(\Lambda + R)^{-1}h^T(t)\rho(t). \tag{4.5}$$

Note that the only difference between the control solution of iSAC (4.4) and SAC (2.6) is that iSAC penalizes the deviation from the nominal control $\||u(t) - v(t)\|_R^2$, where SAC penalizes the magnitude of the applied control $\|u(t)\|_R^2$. The two solutions become identical if the nominal control is zero.

### 4.1.4. Proofs of Descent for MIG-based Continuous Feedback

This subsection proves that the control solution of iSAC is a descent direction over the entire time horizon. Further, it proves that the descent direction remains a direction when scaled in arbitrary ways.

**Proposition 1.** *Consider systems with state $s$ and control $u$ and an objective given by* (2.3)*. Then, the control policy given by* (4.5) *is a descent direction for all $t \in [t_o, t_f]$. Further, if there exists $t \in [t_o, t_f]$ for which the MIG is negative, then the feedback policy in* (4.5) *will decrease the cost. Moreover, this feedback policy converges to the local minimizer trajectory, as stated in Definition 1.*

Proof. Using a first-order Taylor expansion, I write

$$J(v(t) + w(t)) \approx J(v(t)) + \frac{\partial J}{\partial u(t)}\bigg|_{v(t)} \cdot w(t). \tag{4.6}$$

For objectives of the form in (2.3), I use the Gâteaux derivative to calculate the gradient of the cost with respect to the control

$$\frac{d}{d\epsilon} J(v(t) + \epsilon w(t))\bigg|_{\epsilon=0} = \int_{t_o}^{t_f} \rho(t)^T h(t) \cdot w(t) \, \mathrm{d}t. \tag{4.7}$$

From (4.6) and (4.7), the first-order change in cost can be approximated with

$$\Delta J \approx \int_{t_o}^{t_f} \rho^T(t) h(t) \cdot w(t) \mathrm{d}t. \tag{4.8}$$

Equivalently, using the expression of the MIG (2.4) for control-affine dynamics (8.21),

$$\begin{aligned}
\frac{dJ}{d\lambda_+} =& \rho^T(f_2 - f_1) \\
=& \rho^T(g + hu - g - hv) \\
=& \rho^T h(u - v) \\
=& \rho^T hw,
\end{aligned}$$

such that, using (4.2), I can write

$$\Delta J \approx \int_{t_o}^{t_f} \frac{dJ}{d\lambda_+} \mathrm{d}t,$$

which, with the update of (4.5) and given that $\alpha_d < 0$, becomes

$$\Delta J \approx \int_{t_o}^{t_f} \alpha_d \|\rho^T(t)h(t)\|^2_{(\Lambda(t)+R)^{-1}} \mathrm{d}t \leq 0.$$

The equality is true when

$$\begin{aligned} \Delta J = 0 &\Leftrightarrow \rho^T(t)h(t) = 0 \; \forall \; t \in [t_o, t_f] \\ &\Leftrightarrow \frac{dJ}{d\lambda_+} = 0 \; \forall \; t \in [t_o, t_f]. \end{aligned}$$

Therefore, if there exists $t \in [t_o, t_f]$ for which $dJ/d\lambda_+ < 0$, then $\Delta J < 0$. The change in cost, to first order, will be zero if and only if $\rho^T(t)h(t) = 0$ for all $t \in [t_o, t_f]$, which is the condition for a minimum according to Pontryagin's Maximum Principle (for objectives of the form in (2.3)). □

Note that the SAC policy (4.3) is not a guaranteed descent direction over the entire horizon for control-independent cost functions (2.3). On the other hand, it is straightforward to show that the SAC policy is a descent direction over the entire horizon for control-dependent cost functions. The difference between SAC and iSAC and their relation to Gradient Descent is examined in Section 4.2.

Next, I prove that one can scale the descent direction in an arbitrary way, differently across the time-horizon, and still obtain a descent direction, provided that the direction of the update is maintained.

**Proposition 2.** *Consider systems with state $x$ and control $u$ and an objective metric given by* (2.3)*. Let $\Gamma(t) \succ 0$ be a diagonal matrix. The control update $w(t)$ given by* (4.5) *remains a descent direction for the entire trajectory even when scaled to $w_s(t) = \Gamma(t)w(t)$.*

Proof. Consider the update policy in (4.5) with scaled control $\bar{u}(t)$. Let $w_s(t)$ indicate the perturbation after scaling, such that $\bar{u}(t) = v(t) + w_s(t)$, where $w_s(t) = \Gamma(t)w(t)$, where the diagonal elements of $\Gamma(t)$, $\gamma_i(t) \geq 0$, can be different from each other.

From Proposition 1, the cost change, approximated to first-order, is then

$$\Delta J \approx \int_{t_o}^{t_f} \alpha_d \|\rho^T(t)h(t)\|^2_{\Gamma(t)(\Lambda(t)+R)^{-1}} \mathrm{d}t. \tag{4.9}$$

Given that $\alpha_d < 0$, $\Gamma(t) \succ 0$, and $\Lambda(t) + R \succ 0$, (4.9) is negative if there exists time $t$ in $[t_o, t_f]$ such that $\rho^T(t)h(t) \neq 0 \iff w_s(t) \neq 0$. Hence,

$$\Delta J < 0$$

$$\iff \exists\, t \in [t_o, t_f] \text{ such that } w_s(t) = \Gamma(t)w(t) \neq 0.$$

Therefore, given the update (4.5), the cost—approximated to first-order—is guaranteed to decrease provided that there exists $t \in [t_o, t_f]$ such that $\bar{u}(t) \neq v(t)$. □

I note that Proposition 2 holds even when control inputs are scaled differently in time. Fig. 4.1 shows valid cases of control distortion that remain a descent direction.

Compared to other first-order continuous policies and to the best of the authors' knowledge, the scalability results of the control inputs shown in 2 guarantee descent over a broader set of cases of control distortion. For example, work in [**121**] considers only particular cases of the scalability results shown in Figure 4.1, that of control clipping. Their

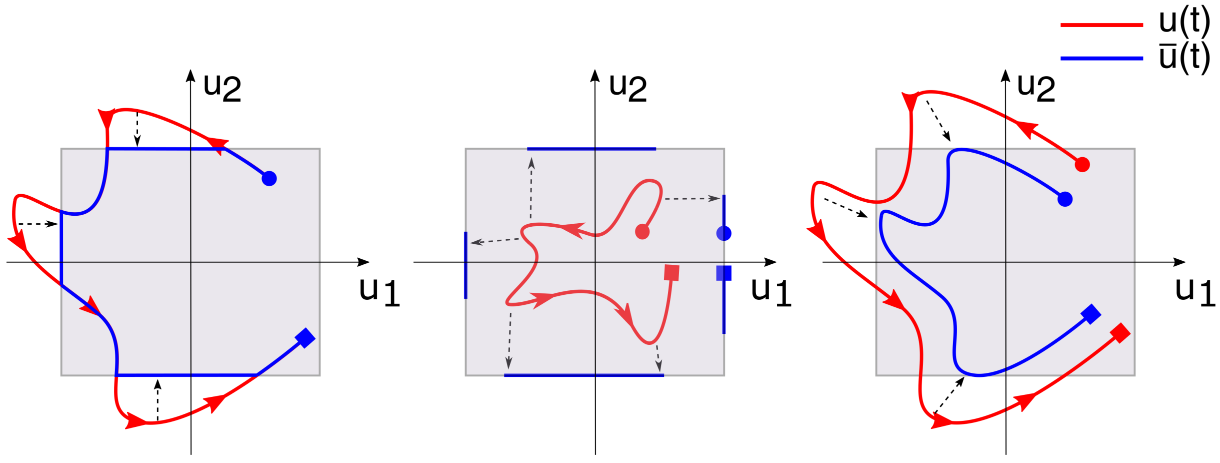


Figure 4.1. Cases of control scaling that remain valid descent directions for the proposed needle-variation controller. The left figure shows control clipping, where values are saturated at a specific threshold; the middle figure shows arbitrary stretching to the saturation limits; the right figure shows proportional scaling that maintains the same direction of the applied control. Control curves are a function of time and arbitrarily shown for a 2-input system for easier visualization. Simulation results in this chapter use control clipping.

method suggests dividing, based on a guess, the time horizon into regions of saturated and unsaturated control inputs and updating actuation only for the latter regions. Work in [**122**] provides a control clipping and proofs that the modified control update remains a descent direction, however it also considers only a special case of the results shown here. In particular, I am not aware of a method that considers the arbitrary scaling of the inputs to the saturation limits, which would be a desired traits for applications capable only of discrete actuation.

## 4.2. Connection to Gradient Descent

Steepest (or gradient) descent algorithm [**123**] is arguably the simplest and most popular first-order trajectory optimization algorithm for general nonlinear optimal control problems. The update policy is given by

$$w(t) = -Rv(t) - h^T(t)\rho(t). \tag{4.10}$$

Conjugate gradient descent [**124**], a first-order technique that provably outperforms steepest descent in terms of convergence at comparable computational cost, is only the conjugate variant of steepest descent. The only difference is the conjugate update, which modifies the control update in (4.10) to ensure successive controls are conjugate with respect to one another. In that sense, all algorithms can have their conjugate variant. For the purposes of comparison, I relate the proposed algorithm to the gradient descent policy (4.10); if the solutions match, then the conjugate variants of both methods will also be identical. To that extent, by showing that NVC can, under certain parameter choices, match steepest descent, the reader should also assume that NVC can also, under the same choices and by using a conjugate update, also match the conjugate gradient descent algorithm.

Next, I compare both variants of single-action control, SAC (4.3) and iSAC (4.4), to gradient descent.

### 4.2.1. Relationship Between Gradient Descent and iSAC

In general, the solution of iSAC (4.5) is not equivalent to the gradient descent.

$$\begin{aligned} \alpha_d(\Lambda + R)^{-1}(h^T\rho) = &- Rv - h^T\rho \\ \alpha_d h^T\rho = &- (\Lambda + R)Rv - (\Lambda + R)h^T\rho. \end{aligned}$$

To show the above relationship is not always true, consider $h^T\rho = 0$ such that

$$0 = -RRv,$$

which is only satisfied if $v = 0$.

### 4.2.2. Relationship Between Gradient Descent and SAC

On the other hand, for specific $\alpha_d$ and $R$ parameters, the SAC update policy (4.3) matches the solution of the gradient descent.

**Proposition 3.** *Consider control affine dynamics* (8.21). *If* $R = I$ *and* $\alpha_d = -(\rho^T hRu_o + \rho^T hh^T\rho + 1)$*, then the update policies of* (4.3) *and* (4.10) *match.*

Proof. Setting the control updates of the proposed algorithm and the Gradient Descent, shown in (4.3) and (4.10) respectively, equal to each other,

$$\begin{aligned}(\Lambda + R)^{-1}(Rv - \alpha_d h^T\rho) =& Rv + h^T\rho \\ Rv - \alpha_d h^T\rho =& (\Lambda + R)(Rv + h^T\rho),\end{aligned}$$

where $\Lambda = h^T\rho\rho^T h \succeq 0$ and $\Lambda + R \succ 0$. I consider two cases: 1) $h^T\rho = 0$ and 2) $h^T\rho \neq 0$.

**Case 1.** $h^T\rho = 0$.

If $h^T\rho = 0$, then also $\Lambda = 0$ and the above equation becomes

$$\begin{aligned} Rv =& R(Rv) \\ (I - R)Rv =& 0 \\ R = I&. \end{aligned}$$

Note that, given $R \succ 0$, the solutions also match if $v = 0$, but that is not always the case, especially when one iterates on previous control solutions.

**Case 2.** $h^T\rho \neq 0$.

Using $R = I$, then

$$\begin{aligned} v - \alpha_d \underbrace{h^T\rho}_{\boldsymbol{p}} &= (\Lambda + I)(v + h^T\rho) \\ 0 &= \boldsymbol{p}\boldsymbol{p}^T v + v + \boldsymbol{p}\boldsymbol{p}^T\boldsymbol{p} + \boldsymbol{p} - v + \alpha_d\boldsymbol{p} \\ 0 &= \boldsymbol{p}(\boldsymbol{p}^T v + \boldsymbol{p}^T\boldsymbol{p} + 1 + \alpha_d) \\ \alpha_d &= -(\boldsymbol{p}^T v + \boldsymbol{p}^T\boldsymbol{p} + 1) \\ \alpha_d &= -(\rho^T h v + \rho^T h h^T \rho + 1) \qquad (4.11) \end{aligned}$$

Alternatively, for $v = 0$, then

$$\begin{aligned} -\alpha_d \boldsymbol{p} =& (\Lambda + R)(h^T \rho) \\ =& (\boldsymbol{p}\boldsymbol{p}^T + R)\boldsymbol{p} \\ =& \boldsymbol{p}^T \boldsymbol{p}\boldsymbol{p} + R\boldsymbol{p}, \end{aligned}$$

which, from inspection, has a solution $\alpha_d = -\rho^T h h^T \rho - \gamma \mathcal{I}$ for $R = \gamma I$, where $\gamma \in \mathbb{R}^+$. $\square$

It is worth noting that $\alpha_d$, as given by (4.11), is not necessarily negative. Similarly, it can easily be shown that the SAC solution (4.3) does not always generate a descent direction, that is a negative $\Delta J$, as shown in (4.8).

Next, I prove that for $\alpha_d$ given by (4.11), the policies shown in (4.5) and (4.10) always match. Although this may seem trivial, a solution may not always exist. For example, consider two vectors $x$ and $z$. Even though a scalar $c = \frac{z^T x}{\|z\|}$ algebraically satisfies the equation $x = cz$, such a solution is infeasible if the vectors are not parallel to each other.

**Proposition 4.** *Consider control affine dynamics* (8.21)*. If $R = I$ and $\alpha_d = -(\rho^T h R v h + \rho^T h h^T \rho + 1)$, then the control policies of* (4.5) *and* (4.10) *match.*

Proof. Setting the update policies of the proposed algorithm and Gradient Descent equal to each other,

$$\begin{aligned} v - \alpha_d \boldsymbol{p} =& \Lambda v + v + \Lambda \boldsymbol{p} + \boldsymbol{p} \\ 0 =& \Lambda(v + \boldsymbol{p}) + \boldsymbol{p} + \alpha_d \boldsymbol{p}, \end{aligned} \tag{4.12}$$

where

$$\Lambda = \boldsymbol{p}\boldsymbol{p}^T = \begin{bmatrix} p_1^2 & \cdots & p_1p_n \\ & \ddots & \\ \vdots & p_ip_j & \vdots \\ & & \ddots \\ p_np_1 & \cdots & p_n^2 \end{bmatrix},$$

such that

$$\begin{bmatrix} p_1^2 & \cdots & p_1p_n \\ \ddots & & \\ \vdots & p_ip_j & \vdots \\ & & \ddots \\ p_np_1 & \cdots & p_n^2 \end{bmatrix} \begin{bmatrix} v_1+p_1 \\ v_2+p_2 \\ \vdots \\ \\ v_n+p_n \end{bmatrix} + \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ \\ p_n \end{bmatrix} + \alpha_d \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ \\ p_n \end{bmatrix} = 0.$$

In the $k^{th}$ row (where $k \in [1, n]$),

$$p_k \sum_{i=1}^{n} p_i(v_i+p_i)+p_k+\alpha_d p_k = 0$$

$$\alpha_d p_k = -p_k(\sum_{i=1}^{n} p_i(v_i+p_i)+1).$$

If $p_k = 0$, the above equation is always satisfied (for all $k \in [1, n]$), regardless of $\alpha_d$. If $p_k \neq 0$, dividing both sides by $p_k$ gives

$$\alpha_d = -\sum_{i=1}^{n} p_i(v_i+p_i)-1, \tag{4.13}$$

which is independent of $k$ and, thus, the same equation holds for all $k \in [1, n]$. Thus, $n$ equations collapse to 1, the same as the number of unknowns ($\alpha_d$). Therefore, (4.12) has always a solution, shown in (4.13), and the policies of iSAC and Gradient Descent match each other. $\square$

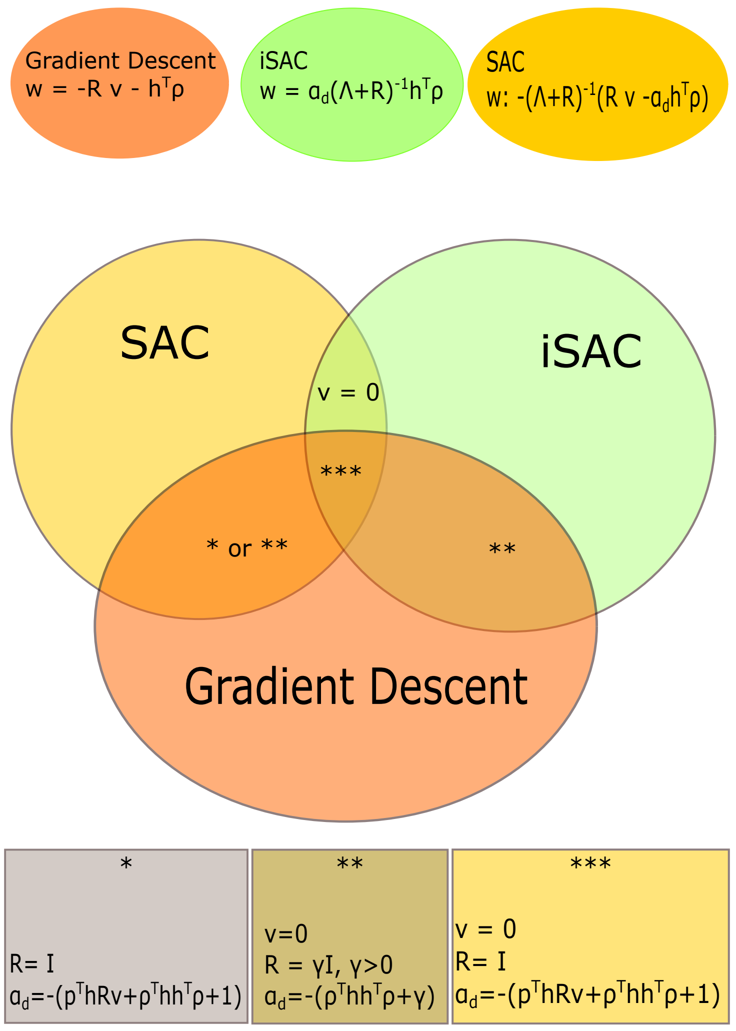

Gradient Descent
$w = -R\,v - h^T\rho$
iSAC
$w = \alpha_d(\Lambda+R)^{-1}h^T\rho$
SAC
$w: -(\Lambda+R)^{-1}(R\,v - \alpha_d h^T\rho)$
SAC
iSAC
$v = 0$
***
* or **
**
Gradient Descent
*
R= I
$\alpha_d=-(\rho^T hRv+\rho^T hh^T\rho+1)$
**
v=0
$R = \gamma I$, $\gamma>0$
$\alpha_d=-(\rho^T hh^T\rho+\gamma)$
***
v = 0
R= I
$\alpha_d=-(\rho^T hRv+\rho^T hh^T\rho+1)$

Figure 4.2. Visual relationship between the control solutions of SAC, iSAC, and Gradient Descent.

## 4.3. Example Systems

In this section, I use the benchmark cart pendulum system to test NVC and compare it both to trajectory optimization and single-action algorithms. I also use a 3-link biped to demonstrate the performance of NVC on a more challenging system and its empirical robustness against disturbances. The algorithmic steps are shown in Algorithm I and the parameters used in the simulations are reported in Table 4.1.

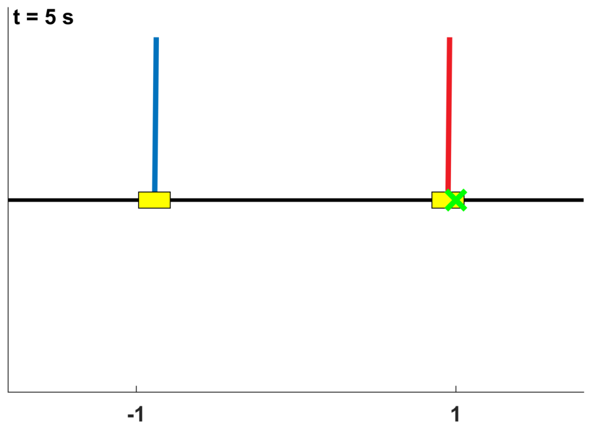
### 4.3.1. Cart Pendulum

To compare performance across algorithms I use the cart pendulum system, which has been a popular testbed for conventional controllers [**24, 125–127**]. Dynamics are used as in [**128**] and a state vector $x = [x_c, \dot{x}_c, \theta, \dot{\theta}]$ is utilized.

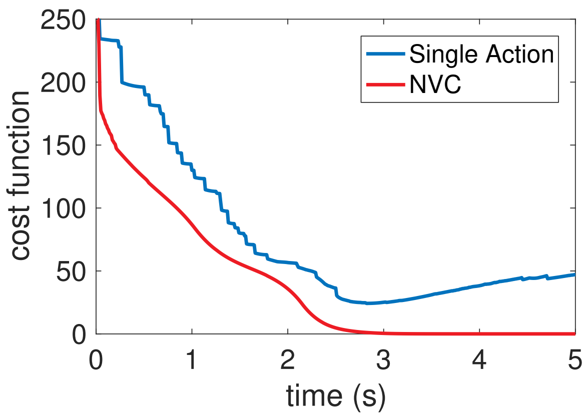
**Algorithm I** Needle Variation Controller

```
Initialize $k \in (0, 1)$.
 1: Simulate $x(t)$ and $\rho(t)$ in $[t_o, t_o + T]$ from $f_1$
 2: Compute $J(v(t))$
 3: Compute $w(t)$ given by (4.5)
 4: Compute $J(u(t))$ and $\Delta J$
 5: $\beta = 0$
 6: while $\Delta J > \Delta J_{min}$ do
 7:     $k = k^\beta$
 8:     Update $u(t) = v(t) + kw(t)$
 9:     Saturate $u(t)$
10:     Compute $J(u(t))$
11:     $\beta = \beta + 1$
12: end while
```

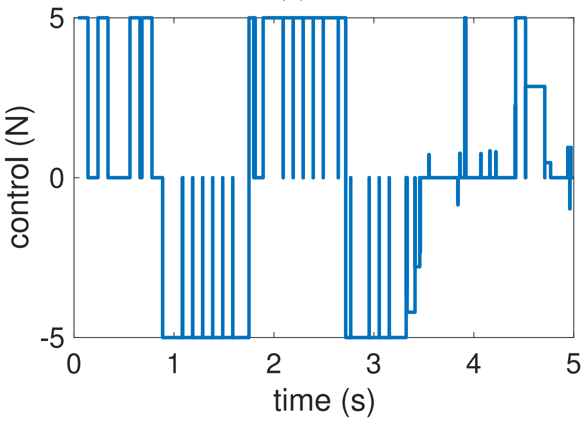
Fig. 4.3 illustrates a comparison in performance between single-action and continuous-action needle variation controllers. Saturation limits of 5 N are imposed. These limits are the same as in [**2**], where they are unable to assure convergence for all four states, and



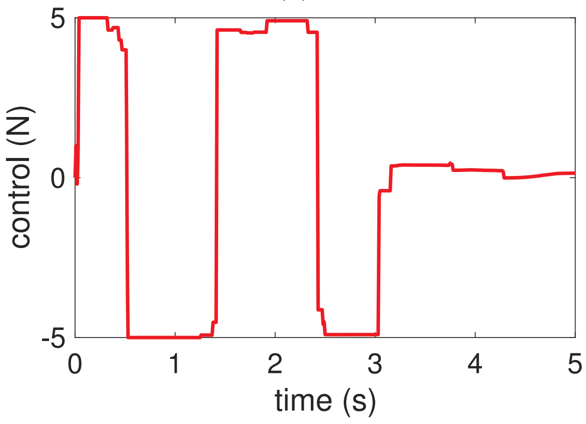
(a)



(b)



(c)

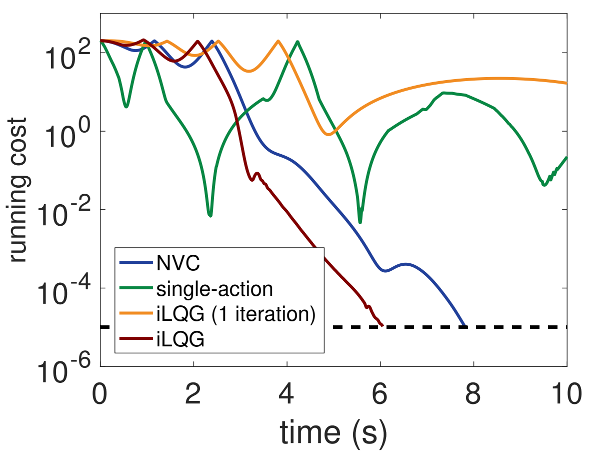


(d)

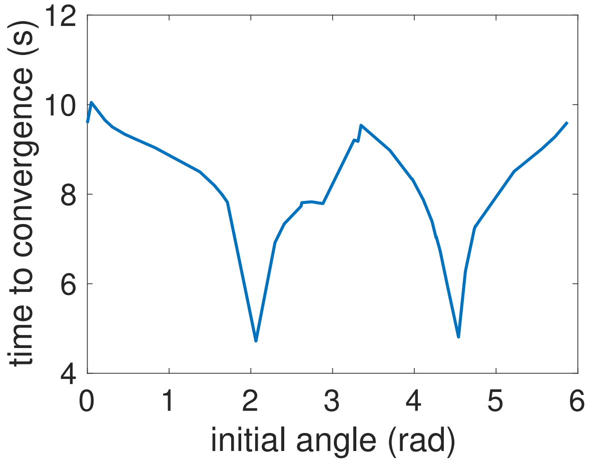
Figure 4.3. Performance of a single-action needle variation controllers and NVC on the inversion task of a cart pendulum. NVC successfully inverts the pendulum with saturation limits of 5 N, while single-action controls are reported to require at least 15 N for the four-state system [**2**]. Given the time, NVC can invert the pendulum with as low as 1N saturation limits. Note that the NVC solution appears similar to a low-pass filtered single-action solution.

four times lower than those used in [**3**] to invert the pendulum with single-action control. I further compare NVC to the iLQG controller with results shown in Fig. 4.4a.

I demonstrate the robustness of NVC through a Monte Carlo simulation over the initial pendulum angle $\theta_0$ as well as over the relative state weights in $Q$ and $R$ matrices that define the quadratic objective. Fig. 4.4b demonstrates that NVC converges in all 50 trials



(a)



(b)

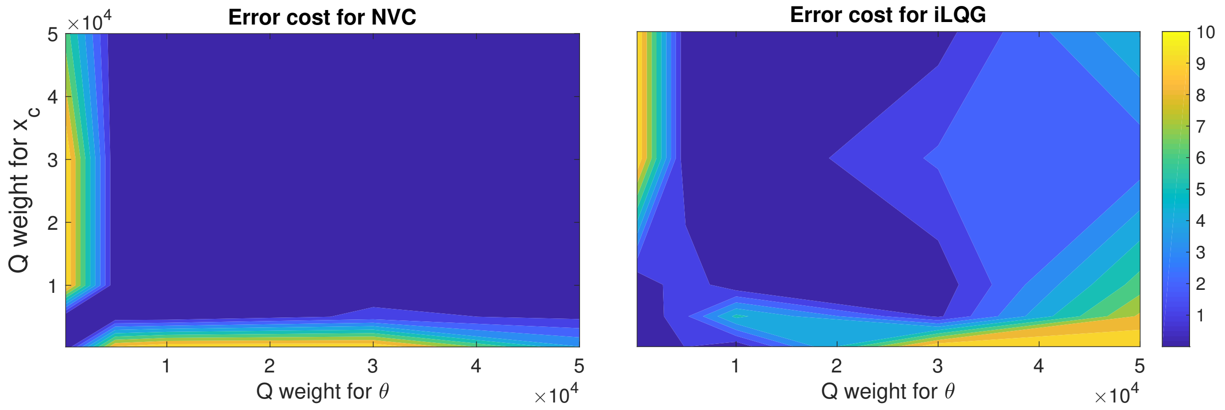
Figure 4.4. Fig. 4.4a compares NVC, single-action feedback, and two implementations of iLQG—one limited to one forward and backward pass per iteration and one with unlimited passes per iteration. The latter provides the optimizer, while the former can be thought of as providing a good enough solution. Fig. 4.4b shows a Monte Carlo simulation over the initial angle $\theta_0$. NVC successfully leads to inversion in 50 out of 50 trials with convergence times ranging from 4.7 to 10.1 seconds. For the same set of parameters and range of initial conditions, single-action policies do not converge for any of the trials [**3**].

randomly sampled over $[0, 2\pi]$. Fig. 4.5 shows that NVC is robust to changes in the $Q$ and $R$ matrices within almost two orders of magnitude, while iLQG[1] does not converge as reliably.

### 4.3.2. 3-link Biped

I use a 3-link model that consists of a torso and two identical legs without knees or ankles; all links are rigid. The behavior of the system is described by the swing phase dynamics, derived using the method of Lagrange [**130**] and the impact update on the states [**131**,**132**]. To handle impulses that occur at impact, I modify the adjoint variable using reset maps, similar to [**2**]. Specifically, I simulate the adjoint variables using the same differential

[1]I implement iLQG using the code that is available here and is described in [**129**].



(a) (b)

Figure 4.5. Comparison of NVC and iLQG performance over a range of cost function formulations. Terminal cost at the end of a 15-second simulation was interpolated over the range of Q sampled. For all simulations Q was defined as a diagonal matrix with weights = 0 for all states except $\theta$ and $x_c$, for which weights are shown on the x- and y-axis, respectively. Note the robustness of NVC for a range of task definitions.

Table 4.1. Parameter values for single-action control, iLQG, and NVC

| | Cart Pendulum | Biped |
|---|---|---|
| $x$ | $[x_c, \dot{x}_c, \theta, \dot{\theta}]$ | $[\theta_s, \theta_m, \theta_t, x_{hip}, y_{hip}, \dot{\theta}_s, \dot{\theta}_m, \dot{\theta}_t, \dot{x}_{hip}, \dot{y}_{hip}]$ |
| $Q$ | $[2, 0, 3, 0] \cdot 10^4$ | $[10, 350, 350, 0, 0, 0, 0, 0, 0, 0]$ |
| $R$ | $[0.2]$ | $[0.1, 0.1]$ |
| $T$ | 2.2 s | 1.0 s |
| $t_f$ | 15 s | 10 s |
| $t_s$ | 0.005 s | 0.01 s |
| $u_{max}$ | $\pm[20]$ | $\pm[30, 30]$ |

equation as (2.5) and utilize a reset map at time of impact to account for state jumps, as outlined in Equation (40) in [**2**].

Parameters of the dynamic model can be found in [**132**]. The biped states are described by $x = [\theta_s, \theta_m, \theta_t, x_{hip}, y_{hip}, \dot{\theta}_s, \dot{\theta}_m, \dot{\theta}_t, \dot{x}_{hip}, \dot{y}_{hip}]$, where $\theta_s, \theta_m$, and $\theta_{hip}$ are the angles of the support leg, moving leg, and torso from the vertical, respectively; $x_{hip}$ and $y_{hip}$ are the Cartesian coordinates of the hip. As legs alternate between support and swing phase, $\theta_s$ and $\theta_m$ swap upon impact. I illustrate the performance of NVC for this biped system in Fig. 4.6. The target states are given by $x_d = [\pi/8, \pi/8, \pi/18, 0, 0, 1, 1, 0, 0, 0]$ and provide the system with a command of driving the swing leg forward and maintaining a slightly forward-leaning posture of the torso that is commonly observed in everyday walking.

As such, NVC enables the generation of a dynamic gait rather than tracking pre-computed reference trajectories [**133**–**136**]. As seen in Fig. 4.6, NVC creates a quasi-periodic gait without gait-specific constraints, such as touch-down or take-off angles used by other controllers [**137**, **138**]. What is more, I do not initiate walking from favorable configurations, contrary to other approaches [**139**, **140**] that either ignore gait initiation or treat it separately. Lastly, I use the full dynamic model of the biped [**141**], while other online controllers take advantage of simplified dynamics or linear model approximations [**142**–**144**].

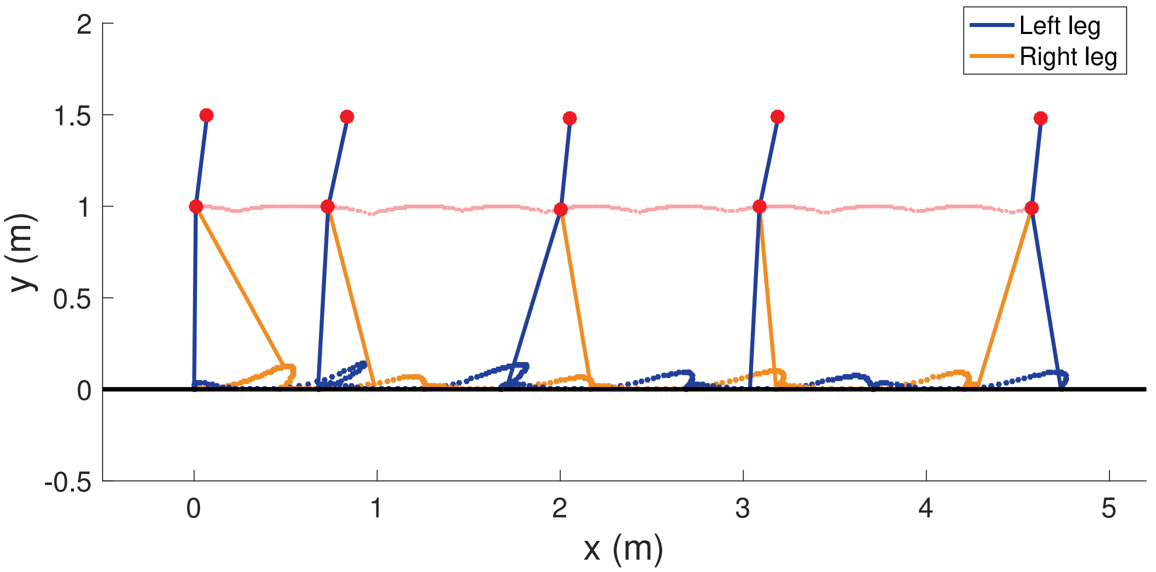
Fig. 4.7 demonstrates the algorithm's robustness against disturbances, which are introduced in the form of discontinuities in the angular velocity of the torso. Fig. 4.8 introduces forces on the torso for 0.02 s at various points throughout a gait cycle. The force disturbances that the biped can recover from range between $-32$ N and 18 N depending on the biped's position in the cycle. This range, when scaled by the size of the biped considered here (35 kg compared to $57^+$ kg), corresponds to the tolerable range reported in literature [**145**, **146**] for different controllers for kneed bipeds.

## 4.4. Discussion

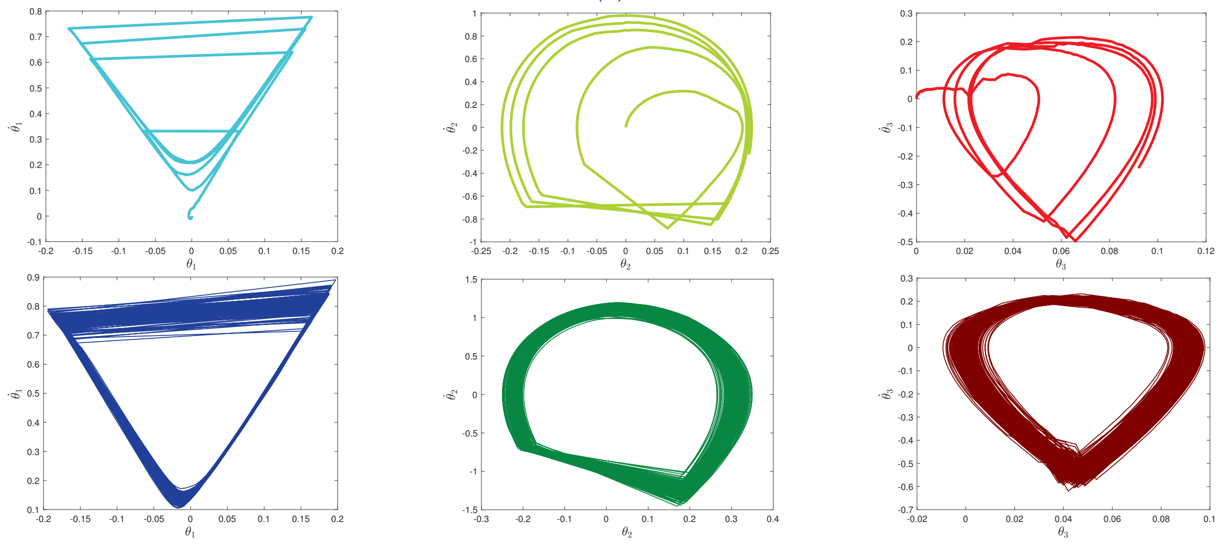
This study presents the Needle Variation Controller (NVC), a nonlinear feedback method. The algorithm synthesizes its control response in ways similar to the needle variation controllers shown in [**2**] and [**147**], but differs from these single-action policies in that it utilizes the entire time horizon of control values at each iteration. The resulting feedback of the proposed scheme guarantees descent, even in the presence of actuation limits. NVC maintains the desirable traits of existing needle variation methods, namely a large region of attraction, a closed-form expression, and computational efficiency.

Compared to popular feedback schemes, such as iLQG or DDP, the proposed method uses only first-order information and is computationally faster, which makes it suitable for online applications. Further, as the example of the biped model shows, the proposed scheme lends itself to hybrid system applications without additional overhead associated with switched dynamics and impacts. The controller design is not problem-specific, which makes NVC an attractive controller for versatile robotic applications.

Finally, the robustness of the controller can become especially meaningful when dealing with human-in-the-loop approaches, where safety is of utmost importance. It is also critical in applications such as rehabilitation equipment, exoskeletons, and other assistive devices that require low failure rates in the presence of unanticipated system inputs or changes in the environment.



(a)



(b)

Figure 4.6. Forward locomotion of a planar biped using a continuous-action needle variation controller. Note that the NVC-controlled biped is able to perform gait initialization and establish a quasi-periodic gait pattern. Fig. 4.6a shows the walking pattern of the biped. Fig. 4.6b shows phase plots of the legs and torso angles; the upper plots correspond to the gait initiation (first five seconds); the lower plots correspond to the simulation from $t = 10$ s until the end of the five-minute walk. The imperfect gait cycle can allude to a certain level of robustness with respect to deviations from a nominal gait phase.

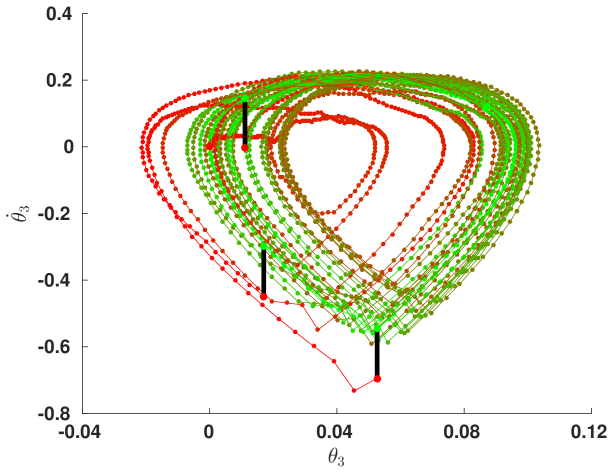


Figure 4.7. Gait recovery from disturbances on the torso angular velocity. The figure shows the phase plane related with the torso angle given disturbances of −0.15 rad/s applied every 8 seconds over a 30-second simulation. The introduced disturbance discontinuities are marked with a solid black line. The trajectory in-between disturbances changes linearly with time from red to green color. As a result, the green trajectory is the converging gait cycle and the red indicates deviations present immediately after the disturbances.

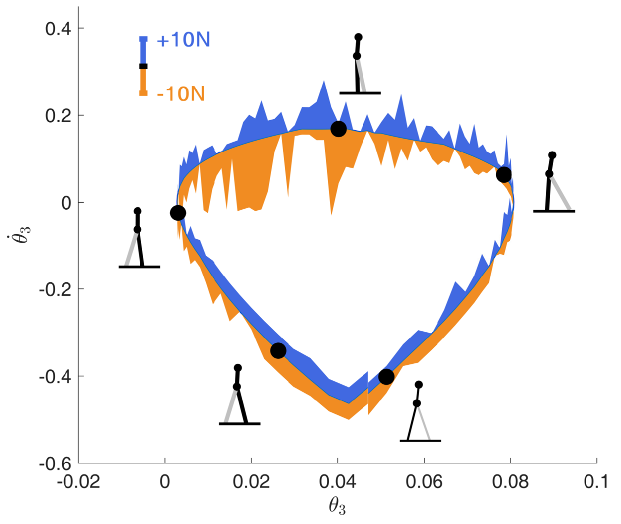


Figure 4.8. Gait recovery from external forces applied on the torso for 0.02 s at various times of a full gait cycle. The shaded regions indicate the maximum magnitude of positive (blue) and negative (orange) forces tolerated by the biped without losing balance. The legend bar in the figure indicates the magnitude scale of the forces.

CHAPTER 5

# Second-Order SAC with Controllability-Based Convergence Guarantees

This chapter derives nonlinear feedback control synthesis for general control affine systems using second-order actions—the second-order needle variations of optimal control—as the basis for choosing each control response to the current state. A second result of the chapter is that the method provably exploits the nonlinear controllability of a system by virtue of an explicit dependence of the second-order needle variation on the Lie bracket between vector fields. As a result, each control decision necessarily decreases the objective when the system is nonlinearly controllable using first-order Lie brackets. Simulation results using a differential drive cart, an underactuated kinematic vehicle in three dimensions, and an underactuated dynamic model of an underwater vehicle demonstrate that the method finds control solutions when the first-order analysis is singular. Lastly, the underactuated dynamic underwater vehicle model demonstrates convergence even in the presence of a velocity field.

## 5.1. Dependence of Needle Variation Controls on Nonlinear Controllability

In this section, I relate the controllability of systems to first- and second-order needle variation actions. After presenting the MIH expression, I relate the MIH to the Lie bracket terms between vector fields. Using this connection, I tie the descent property of needle variation actions to the controllability of a system and prove that second-order needle variation controls can produce control solutions for a wider set of the configuration state space than first-order needle variation methods. As a result, I am able to constructively compute, via an analytic solution, control formulas that are guaranteed to provide descent, provided that the system is controllable with first-order Lie brackets. Generalization to higher-order Lie brackets appears to have the same structure, but that analysis is postponed to future work.

### 5.1.1. Second-Order Mode Insertion Gradient

Needle variation methods in optimal control have served as the basic tool in proving the Pontryagin's Maximum Principle [**148**–**150**]. Using piecewise dynamics, they introduce infinitesimal perturbations in control that change the default trajectory and objective (see Fig. 2.1). Such dynamics are typically used in optimal control of hybrid systems to optimize the schedule of a-priori known modes [**108**, **151**].

Here, instead, I consider dynamics of a single switch to obtain a new control mode $u$ at every time step that will optimally perturb the trajectory [**2**]. The feedback algorithm presented in [**2**], however, only considers the first-order sensitivity of the cost function to a needle action and, as a result, often fails to provide solutions for controllable underactuated systems. By augmenting the algorithm with higher order information (via the MIH), I am able to provide solutions in cases when the first-order needle variation algorithm in [**2**] is singular.

Consider control-affine dynamics (8.21), a time period $[t_o, t_f]$ and control modes described by (2.2).

The derivation of the mode insertion Hessian is similar to [**152**] and is presented in the Appendix. For dynamics that do not depend on the control duration, the mode insertion Hessian (MIH)[1] is given by

$$\frac{d^2J}{d\lambda_+^2} = (f_2-f_1)^T\Omega(f_2-f_1)+\rho^T(D_xf_2\cdot f_2+D_xf_1\cdot f_1-2D_xf_1\cdot f_2)-D_xl_1\cdot(f_2-f_1), \tag{5.1}$$

where $\Omega : \mathbb{R} \mapsto \mathbb{R}^{N\times N}$ is the second-order adjoint state, which is calculated from the default trajectory and is given by

$$\begin{aligned} \dot{\Omega} &= -D_xf_1{}^T\Omega-\Omega D_xf_1-D_x^2l_1-\sum_{i=1}^{N}\rho_iD_x^2f_1^i \\ \text{subject to: } \Omega(t_f) &= D_x^2m(x(t_f))^T. \end{aligned} \tag{5.2}$$

The superscript $i$ in the dynamics $f_1$ refers to the $i^{th}$ element of the vector.

### 5.1.2. Dependence of Second Order Needle Variations on Lie Bracket Structure

The Lie bracket of two vectors $f(x)$, and $g(x)$ is

$$[f,g](x) = \frac{\partial g}{\partial x}f(x)-\frac{\partial f}{\partial x}g(x),$$

[1]In this work, I consider the second-order sensitivity with respect to an action centered at one single application time $\tau$. It is also possible to consider the second-order sensitivity with respect to two application times $\tau_i$ and $\tau_j$ in the same iteration. Assuming that the entire control curve is a descent direction over the time horizon for second-order needle variation solutions, as I have proved is the case for first-order needle variation methods, multiple second-order needle actions at different application times would still decrease the objective. On the other hand, searching for two application times would slow down the algorithm and was not preferred in this work.

which generates a control vector that points in the direction of the net infinitesimal change in state $x$ created by infinitesimal noncommutative flow $\phi_\epsilon^f \circ \phi_\epsilon^g \circ \phi_\epsilon^{-f} \circ \phi_\epsilon^{-g} \circ x_0$, where $\phi_\epsilon^f$ is the flow along a vector field $f$ for time $\epsilon$ [**15**, **153**]. Lie brackets are most commonly used for their connection to controllability [**154**, **155**], but here they will show up in the expression describing the second-order needle variation.

I relate second-order needle variation actions to Lie brackets in order to connect the existence of descent-providing controls to the nonlinear controllability of a system. Let $h_i : \mathbb{R} \mapsto \mathbb{R}^{N\times 1}$ denote the column control vectors that make up $h : \mathbb{R} \mapsto \mathbb{R}^{N\times M}$ in (8.21) and $u_i \in \mathbb{R}$ be the individual control inputs. Then, I can express dynamics as

$$f = g + \sum_i^M h_i u_i.$$

and, for default control $v = 0$, I can re-write the MIH as

$$\frac{d^2 J}{d\lambda_+^2} = \Big(\sum_{i=1}^M h_i u_i\Big)^T \Omega \sum_{j=1}^M h_j u_j + \rho^T \Big(\sum_{i=1}^M (D_x h_i u_i) \cdot g - D_x g \cdot (h_i u_i) + \sum_{i=1}^M D_x h_i u_i \sum_{i=1}^M h_i u_i\Big)$$
$$- D_x l_1 \sum_{i=1}^M h_i u_i.$$

Splitting the sum expression into diagonal ($i = j$) and off-diagonal ($i \neq j$) elements, and by adding and subtracting $2 \sum_i^M \sum_{j=1}^{i-1} (D_x h_i u_i)(h_j u_j)$, I can write

$$\sum_{i=1}^M D_x h_i u_i \sum_{i=1}^M h_i u_i = \sum_i^M \sum_{j=1}^{i-1} [h_i, h_j] u_i u_j + 2 \sum_i^M \sum_{j=1}^{i-1} (D_x h_i u_i)(h_j u_j) + \sum_{i=j=1}^M (D_x h_i u_i)(h_i u_i).$$

Then, I can express the MIH as

$$
\begin{aligned}
\frac{d^2 J}{d\lambda_+^2} =& \sum_{i=1}^{M}\sum_{j=1}^{M} u_i u_j h_i^T \Omega h_j \\
&+\rho^T\Big(\sum_{i=2}^{M}\sum_{j=1}^{i-1}[h_i,h_j]u_i u_j + 2\sum_{i=2}^{M}\sum_{j=1}^{i-1}(D_x h_i)h_j u_i u_j + \sum_{i=1}^{M}(D_x h_i)h_i u_i u_i + \sum_{i=1}^{M}[g,h_i]u_i\Big) \\
&- D_x l\Big(\sum_{i=1}^{M} h_i u_i\Big).
\end{aligned}
$$

The expression contains Lie bracket terms of the control vectors that appear in the system dynamics, indicating that second-order needle variations incorporate higher-order nonlinearities. By associating the MIH to Lie brackets, I next prove that second-order needle variation actions can guarantee decrease of the objective for systems that are controllable with first-order Lie brackets.

### 5.1.3. Existence of Control Solutions with First- and Second-Order Mode Insertion Gradients

In this subsection, I prove that the first two orders of the mode insertion gradient can be used to guarantee controls that reduce objectives of the form (2.3) for systems that are controllable with first-order Lie brackets. The analysis is applicable to optimization problems that satisfy the following assumptions.

**Assumption 4.** *The vector elements of dynamics $f_1$ and $f_2$ are real, bounded, $\mathcal{C}^2$ in $x$, and $\mathcal{C}^0$ in $u$ and $t$.*

**Assumption 5.** *The incremental cost $l_1(x)$ is real, bounded, and $\mathcal{C}^2$ in $x$. The terminal cost $m(x(t_f))$ is real and twice differentiable with respect to $x(t_f)$.*

**Assumption 6.** *Default and inserted controls $v$ and $u$ are real, bounded, and $\mathcal{C}^0$ in $t$.*

Under Assumptions 4-6, the MIG and MIH expressions exist and are unique. Then, as I show next, there are control actions that can improve any objective as long as there exists $t \in [t_o, t_f]$ for which $x(t) \neq x^*(t)$.

**Definition 2.** *A trajectory $x^*$ described by a pair $(x^*, u^*)$ is the global minimizer of the objective function $J(x^*(t))$ for which $J(x^*(t)) \leq J(x(t)) \; \forall \; x(t)$.*

Given Definition 1, a trajectory $x^*$ described by a pair $(x^*, u^*)$ is the global minimizer of the cost function in the unconstrained sense (not subject to the dynamics of the system) and satisfies $D_x J(x^*(t)) = 0$ throughout the time horizon considered.

**Assumption 7.** *The pair $(x^*, u^*)$ describes the only trajectory $x^*$ for which the unconstrained derivative of the objective is equal to zero (i.e., $D_x J(x^*(t)) = 0 \; \forall \; t \in [t_o, t_f]$).*

Assumption 4 is necessary to prove that the first-order adjoint is non-zero, which is a requirement for the controllability results shown in this work. It assumes that the objective function in the unconstrained sense does not have a maximizer or saddle point and has only one minimizer $x^*$ described by $(x^*, u^*)$ that indicates the target trajectory or location. It is an assumption that, among other choices, can be easily satisfied with a quadratic cost function that even includes penalty functions associated with physical obstacles.

**Proposition 5.** *Consider a pair $(x, v)$ that describes the state and default control of* (8.21). *If $(x, v) \neq (x^*, v^*)$, then the first-order adjoint $\rho$ is a non-zero vector.*

Proof. Using (2.3), and by Assumption 4,

$$\begin{aligned}
x \neq x^* &\Rightarrow D_x J(x(t)) \neq 0 \\
&\Rightarrow \int_{t_o}^{t_f} D_x l_1(x(t))\mathrm{d}t + D_x m(x(t_f)) \neq 0 \\
&\Rightarrow \int_{t_o}^{t_f} D_x l_1(x(t))\mathrm{d}t \neq 0 \text{ OR } D_x m(x(t_f)) \neq 0 \\
&\Rightarrow D_x l_1(x(t)) \neq 0 \text{ OR } D_x m(x(t_f)) \neq 0 \\
&\Rightarrow \dot{\rho} \neq 0 \text{ OR } \rho(t_f) \neq 0.
\end{aligned}$$

Therefore, if $x \neq x^*$, then $\exists\, t \in [t_o, t_f]$ such that $\rho \neq 0$. $\square$

**Proposition 6.** *Consider dynamics given by* (8.21) *and a trajectory described by state and control* $(x, v)$*. Then, there are always control solutions* $u \in \mathbb{R}^M$ *such that* $\frac{dJ}{d\lambda_+} \leq 0$ *for some* $t \in [t_o, t_f]$.

Proof. Using dynamics of the form in (8.21), the expression of the mode insertion gradient can be written as

$$\frac{dJ}{d\lambda_+} = \rho^T(f_2 - f_1) = \rho^T\Big(h(u - v)\Big).$$

Given controls $u$ and $v$ that generate a positive mode insertion gradient, there always exist control $u'$ such that the mode insertion gradient is negative, i.e. $u' - v = -(u - v)$. The mode insertion gradient is zero for all $u \in \mathbb{R}^M$ if the costate vector is orthogonal to each control vector $h_i$ or if the costate vector is zero everywhere.[2] $\square$

[2]If the control vectors span the state space $\mathbb{R}^N$, the costate vector $\rho \in \mathbb{R}^N$ cannot be orthogonal to each of them. Therefore, for first-order controllable (fully actuated) systems, there always exist controls for which the cost can be reduced to first order.

**Proposition 7.** *Consider dynamics given by* (8.21) *and a pair of state and control* $(x, v) \neq (x^*, v^*)$ *for which* $\frac{dJ}{d\lambda_+} = 0 \; \forall \; u \in \mathbb{R}^M$ *and* $\forall \; t \in [t_o, t_f]$. *Then, the first-order adjoint* $\rho$ *is orthogonal (under the Euclidean inner product) to all control vectors* $h_i$.

Proof. I rewrite (2.4) as

$$\frac{dJ}{d\lambda_+} = 0 \Rightarrow \rho^T \sum_i^M h_i(u_i - v_i) = 0$$
$$\Rightarrow \sum_i^M k_i w_i = 0 \; \forall \; w_i,$$

where $w_i = (u_i - v_i)$ and $k_i = \rho^T h_i \in \mathbb{R}$. The linear combination of the elements of $k$ is zero for any $w_i$, which means $k$ must be the zero vector. By Proposition 5, $\rho \neq 0$ for a trajectory described by a pair of state and control $(x, u) \neq (x^*, u^*)$ and, as a result, $\rho^T h_i = 0 \; \forall \; i \in [1, M]$. □

**Proposition 8.** *Consider dynamics given by* (8.21) *and a pair of state and control* $(x, v) \neq (x^*, v^*)$ *for which* $\frac{dJ}{d\lambda_+} = 0 \; \forall \; u \in \mathbb{R}^M$ *and* $\forall \; t \in [t_o, t_f]$. *Further assume that the control vectors* $h_i$ *and the Lie Bracket terms* $[h_i, h_j]$ *and* $[g, h_i]$*—where* $i, j \in [1, M]$*—span the state space* $\mathbb{R}^N$. *Then, there exist* $i$ *and* $j$ *such that either* $\rho^T[h_i, h_j] \neq 0$ *or* $\rho^T[g, h_i] \neq 0$.

Proof. Let $S = \{h_i, [h_i, h_j], [g, h_i]\} \; \forall \; i, j \in [1, M]$ be a set of vectors that span the state space $\mathbb{R}^N$ ($\text{span}\{S\} = \mathbb{R}^N$). Then, any vector in $\mathbb{R}^N$ can be written as a linear combination of the vectors in $S$. The first-order adjoint is an $N$-dimensional vector, which is non-zero for a trajectory described by a pair of state and control $(x, u) \neq (x^*, u^*)$ by

Proposition 5. Therefore, it can be expressed as

$$\rho \;=\; c_1 h_1 + \cdots + c_M h_M + \sum_{i=2}^{M} \sum_{j=1}^{i-1} c'_{i,j} [h_i, h_j] + \sum_{i=1}^{M} c''_i [g, h_i], \tag{5.3}$$

where $c_i, c'_i, c''_i \in \mathbb{R}$. Left-multiplying (5.3) by $\rho^T$ yields

$$\rho^T \rho = \sum_{i=1}^{M} c_i \rho^T h_i + \sum_{i=2}^{M} \sum_{j=1}^{i-1} c'_{i,j} \rho^T [h_i, h_j] + \sum_{i=1}^{M} c''_i \rho^T [g, h_i].$$

Given that $\frac{dJ}{d\lambda_+} = 0$, and by Proposition 7, $\rho$ is orthogonal to all control vectors $h_i$ (which also implies that the control vectors $h_i$ do not span $\mathbb{R}^N$), the above equation simplifies to

$$\rho^T \rho = \sum_{i=2}^{M} \sum_{j=1}^{i-1} c'_{i,j} \rho^T [h_i, h_j] + \sum_{i=1}^{M} c''_i \rho^T [g, h_i].$$

Because $\rho^T \rho \;\neq\; 0$, there exists $i, j \in [1, M]$ and a Lie bracket term $[h_i, h_j]$, or $[g, h_i]$ that is not orthogonal to the costate $\rho$. That is,

$$\exists\; i, j \in [1, M] \text{ such that } \rho^T [h_i, h_j] \neq 0 \text{ OR } \rho^T [g, h_j].$$

$\square$

First-order needle variation methods are singular when the mode insertion gradient is zero. When that is true, the next result—that is the main piece required for the main theoretical result of this section in Theorem 1—demonstrates that the second-order mode insertion gradient is guaranteed to be negative for systems that are controllable with first-order Lie Brackets, which in turn implies that a control solution can be found with second-order needle variation methods.

**Proposition 9.** *Consider dynamics given by* (8.21) *and a trajectory described by state and control* $(x, v) \neq (x^*, v^*)$ *for which* $\frac{dJ}{d\lambda_+} = 0$ *for all* $u \in \mathbb{R}^M$ *and* $t \in [t_o, t_f]$. *If the control vectors* $h_i$ *and the Lie brackets* $[h_i, h_j]$ *and* $[g, h_i]$ *span the state space (*$\mathbb{R}^N$*), then there always exist control solutions* $u \in \mathbb{R}^M$ *such that* $\frac{d^2 J}{d\lambda_+^2} < 0$.

Proof. See Appendix. □

**Theorem 1.** *Consider dynamics given by* (8.21) *and a trajectory described by state and control* $(x, v) \neq (x^*, v^*)$. *If the control vectors* $h_i$ *and the Lie brackets* $[h_i, h_j]$ *and* $[g, h_i]$ *span the state space* ($\mathbb{R}^N$), *then there always exists a control vector* $u \in \mathbb{R}^M$ *and a duration* $\lambda$ *such that the cost function* (2.3) *can be reduced.*

Proof. The local change of the cost function (2.3) due to inserted control $u$ of duration $\lambda$ can be approximated with a Taylor series expansion

$$J(\lambda) - J(0) \approx \lambda \frac{dJ}{d\lambda_+} + \frac{\lambda^2}{2} \frac{d^2 J}{d\lambda_+^2}.$$

By Propositions 6 and 9, either 1) $\frac{dJ}{d\lambda_+} < 0$ or 2) $\frac{dJ}{d\lambda_+} = 0$ and $\frac{d^2 J}{d\lambda_+^2} < 0$. Therefore, there always exist controls that reduce the cost function (2.3) to first or second order. □

### 5.2. Control Synthesis Based on Second-Order Needle Variations

In this section, I present an analytical solution of first- and second-order needle variation controls that reduce the cost function (2.3) to second order. I then describe the algorithmic steps of the feedback scheme used in the simulation results in Section 5.3.

### 5.2.1. Analytical Solution for Second Order Actions

For underactuated systems, there are states at which $\rho$ is orthogonal to the control vectors $h_i$ (see Proposition 7). At these states, control calculations based only on first-order sensitivities fail, while controls based on second-order information still have the potential to decrease the objective provided that the control vectors and their Lie brackets span the state space (see Theorem 1). I use this property to compute an analytical synthesis method that expands the set of states for which individual actions that guarantee descent of an objective function can be computed.

Consider the Taylor series expansion of the cost around control duration $\lambda$. Given the expressions of the first- and second-order mode insertion gradients, I can write the cost function (2.3) as a Taylor series expansion around the infinitesimal duration $\lambda$ of inserted control $u$:

$$J(\lambda) \approx J(0)+\lambda\frac{dJ}{d\lambda_+}+\frac{\lambda^2}{2}\frac{d^2J}{d\lambda_+^2}. \tag{5.4}$$

The first- and second-order mode insertion gradients used in the expression are functions of the inserted control $u$ in (8.21). Equation (5.4) is quadratic in $u$ and, for a fixed $\lambda$, has a unique solution which is used to update the control actions. Controls that minimize the Taylor expansion of the cost will have the form

$$u^*(t) = \underset{u}{\operatorname{argmin}}\; J(0)+\lambda\frac{dJ}{d\lambda_+}+\frac{\lambda^2}{2}\frac{d^2J}{d\lambda_+^2}+\frac{1}{2}\|u\|_R^2, \tag{5.5}$$

where the MIH has both linear and quadratic terms in $u$. I compute the minimizer of (5.5) to be

$$u^*(t) = [\frac{\lambda^2}{2}\Gamma + R]^{-1}[\frac{\lambda^2}{2}\Delta + \lambda(-h^T\rho)], \tag{5.6}$$

where $\Delta : \mathbb{R} \mapsto \mathbb{R}^{M\times 1}$ and $\Gamma : \mathbb{R} \mapsto \mathbb{R}^{M\times M}$ are respectively the first- and second-order derivatives of $d^2J/d\lambda_+^2$ with respect to the control $u$ (see Appendix). These quantities are given by

$$\Delta \triangleq \Big[\Big[h^T\Big(\Omega^T + \Omega\Big)h + 2h^T\cdot(\sum_{k=1}^{n}(D_x h_k)\rho_k)^T\Big]v + (D_x g\cdot h)^T\rho - (\sum_{k=1}^{n}(D_x h_k)\rho_k)\cdot g + h^T D_x l^T\Big]$$

$$\Gamma \triangleq [h^T\Big(\Omega^T + \Omega\Big)h + h^T\cdot(\sum_{k=1}^{n}(D_x h_k)\rho_k)^T + \sum_{k=1}^{n}(D_x h_k)\rho_k\cdot h].$$

The parameter $R$, a positive definite matrix, denotes a metric on control effort.

The existence of control solutions in (5.6) depends on the inversion of the Hessian $H = \frac{\lambda^2}{2}\Gamma + R$. To practically ensure H is positive definite, I implement a spectral decomposition on the Hessian $H = VDV^{-1}$, where matrices $V$ and $D$ contain the eigenvectors and eigenvalues of $H$, respectively. I replace all elements of the diagonal matrix $D$ that are smaller than $\epsilon$ with $\epsilon$ to obtain $\bar{D}$ and replace $H$ with $\bar{H} = V\bar{D}V^{-1}$ in (5.6). I prefer the spectral decomposition approach to the Levenberg-Marquardt method ($\bar{H} = H + \kappa I \succ 0$), because the latter affects all eigenvalues of the Hessian and further distorts the second-order information. At saddle points, I set the control equal to the eigenvector of $H$ that corresponds to the most negative eigenvalue in order to descend along the direction of most negative curvature [**156**–**159**].

Synthesis based on (5.6) provides controls at time $t$ that guarantee to reduce the cost function (2.3) for systems that are controllable using first-order Lie brackets. Control solutions are computed by forward simulating the state over a time horizon $T$ and backward simulating the first- and second-order costates $\rho$ and $\Omega$. As is shown next, this leads to a very natural, and easily implementable, algorithm for applying cost-based feedback while avoiding iterative trajectory optimization.

### 5.2.2. Algorithmic Description of Control Synthesis Method

**Algorithm I**

```
1. Simulate states and costates with default dynamics $f_1$ over a time horizon $T$
2. Compute optimal needle variation controls
3. Saturate controls
4. Use a line search to find control duration that ensures reduction of the cost function
   (2.3)[3]
```

The second-order controls in (5.6) are implemented in a series of steps shown in Algorithm I and visualized in Fig. 5.1. I compare first- and second-order needle variation actions by implementing different controls in Step 2 of Algorithm I. For the first-order case, I implement controls that are the solution to a minimization problem of the first-order sensitivity of the cost function (2.3) and the control effort

$$\begin{aligned} u^*(t) &= \min_u \quad \frac{1}{2}\Big(\frac{dJ_1}{d\lambda_i^+} - \alpha_d\Big)^2 + \frac{1}{2}\|u\|_R^2 \\ &= (\Lambda + R^T)^{-1}(\Lambda v + h^T \rho \alpha_d), \end{aligned} \tag{5.7}$$

where $\Lambda \triangleq h^T\rho\rho^T h$ and $\alpha_d \in \mathbb{R}^-$ expresses the desired value of the mode insertion gradient term (see, for example, [**65**]). Typically, $\alpha_d = \gamma J_o$, where $J_o$ is the cost function (2.3)

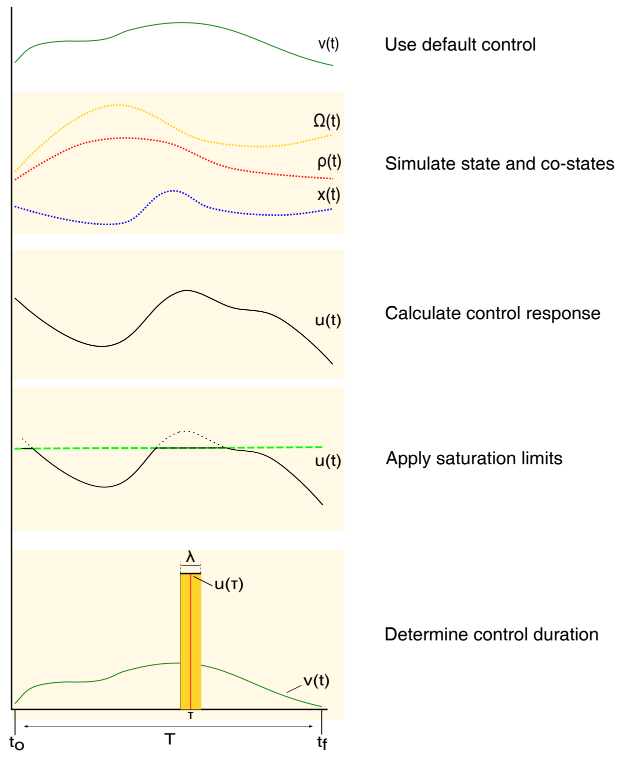


Figure 5.1. The steps of the controller outlined by Algorithm I. Using the default control, the states and co-states are forward-simulated for the time horizon $[t_o, t_o+T]$. The optimal control response is computed from (5.6), and saturated appropriately. At the end, the algorithm determines the finite duration of the inserted single action, evaluated at an application time $\tau$, with a line search.

computed using default dynamics $f_1$. For second-order needle variation actions, I compute controls using (5.6). As Fig. 5.1 indicates, the applied actuation is the saturated value of the control response of either (5.6) or (5.7), evaluated at the application time $\tau$.

While not shown here, [**2**] proves that the first-order needle variation control solutions (5.7) remain a descent direction after saturation. I extend this result to show that the entire control signal over the time horizon, and not a needle action, remains a descent direction when saturated by an arbitrary amount. While I have not yet formally proved a similar property for the second-order needle variation controls (11), one can test and identify if the saturated controls would decrease the cost function before applying any actuation. In addition, the results of this work rely on the sign and not the magnitude of the control solutions, suggesting that the saturated second-order solutions in (5.6) also provide a descent direction.

Further, needle variation actuation as shown in Fig. 5.1 may be practically infeasible or at least problematic for motors due to the abrupt changes in the control. There are two remedies to this issue. First, introducing additional filter states associated with the control can constraint the changes in actuation [**51**]. Second, one can show that the entire curve of the first-order needle variation solution is a descent direction. Assuming the same is true for the second-order solutions as well, one could either apply part of the continuous control solution around the time of application $\tau$ or filter the discontinuous actuation in hardware and still provide descent with more motor-friendly actuation.

### 5.2.3. Comparison to Alternative Optimization Approaches

Algorithm I differs from controllers that compute control sequences over the entire time horizon in order to locally minimize the cost function. Rather, the proposed scheme utilizes the *time-evolving sensitivity* of the objective to an infinitesimal switch from $v$ to $u$ and searches a one-dimensional space for a finite duration of a single action that will optimally improve the cost. It does so using a closed-form expression and, as a result, avoids the expensive iterative computational search in high-dimensional spaces, while it may still get closer to the optimizer with one iterate.

Specifically, in terms of computational effort, Algorithm I computes two $n \times 1$ (state (8.21) and first-order adjoint (2.5) variables) and one $n \times n$ (second-order adjoint (5.2)) differential equations and searches. All simulations presented in this chapter are able to run in real time, including the final 13-dimensional system. However, real-time execution is not guaranteed for higher dimensional systems. Nevertheless, the presented algorithm runs faster than the iLQG method for the simulations considered here.

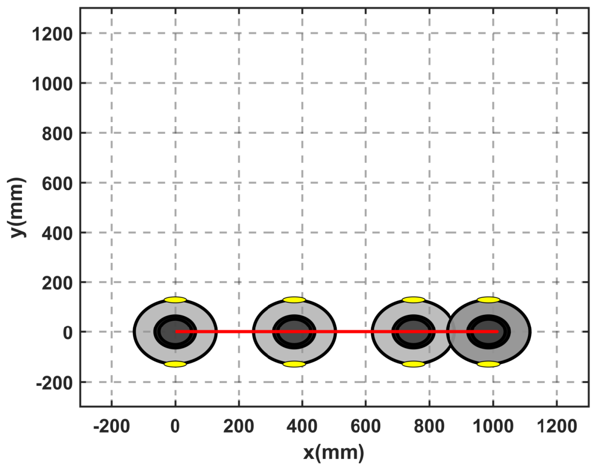
Further, compared to traditional optimization algorithms such as iLQG, needle variation solutions exist globally, demonstrate a larger region of attraction and have a less complicated representation on Lie Groups [**51**]. These traits naturally transfer to second-order needle controls (5.6) that also contain the first-order information present in (5.7). In addition, as this chapter demonstrates, the suggested second-order needle variation controller has formal guarantees of descent for systems that are controllable with first-order Lie brackets, which—to the best of my knowledge—is not provided by any alternative method.

Given these benefits, the authors propose second-order needle variation actions as a complement to existing approaches for time-sensitive robotic applications that may be subject to large initial error, Euler angle singularities, or fast-evolving (and uncertain) objectives.

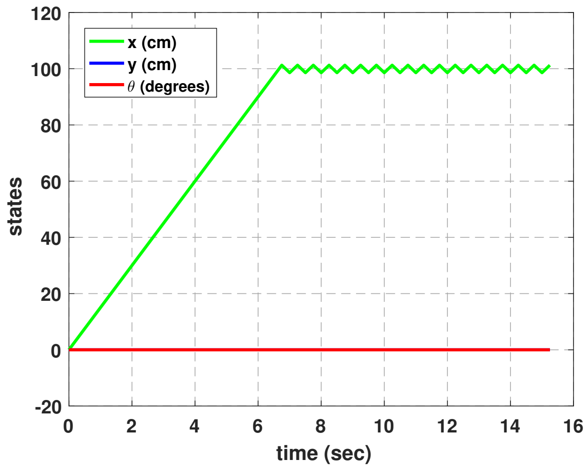
Next, I implement Algorithm I using first or second-order needle variation controls (shown in (5.7) and (5.6), respectively) to compare them in terms of convergence success on various underactuated systems.

## 5.3. Simulation Results

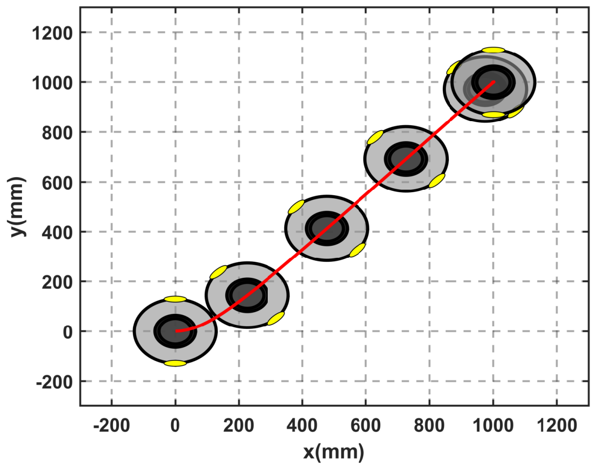
The proposed synthesis method based on (5.6) is implemented on three underactuated examples—the differential drive cart, a 3D kinematic rigid body, and a dynamic model of an underwater vehicle. The kinematic systems of a 2D differential drive and a 3D


y(mm)
x(mm)

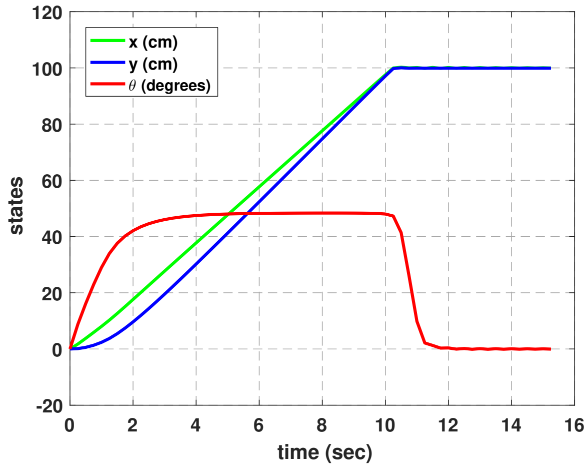
(a)


x (cm)
y (cm)
$\theta$ (degrees)
states
time (sec)

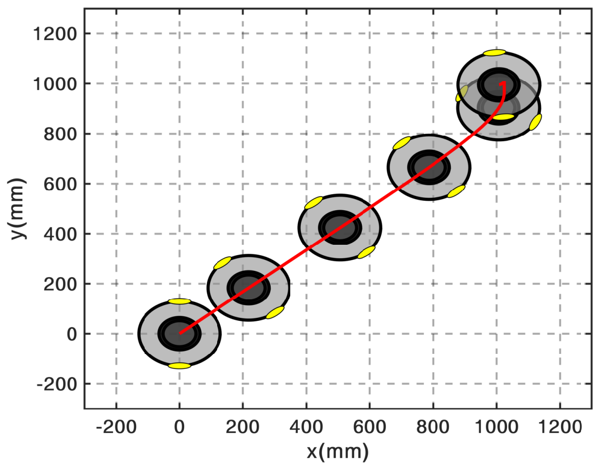
(b)


y(mm)
x(mm)

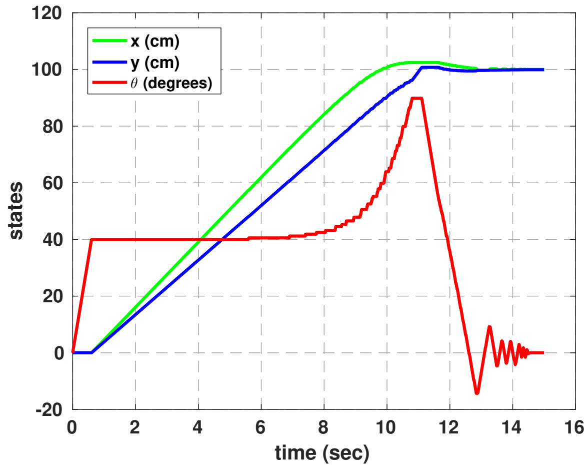
(c)


x (cm)
y (cm)
$\theta$ (degrees)
states
time (sec)

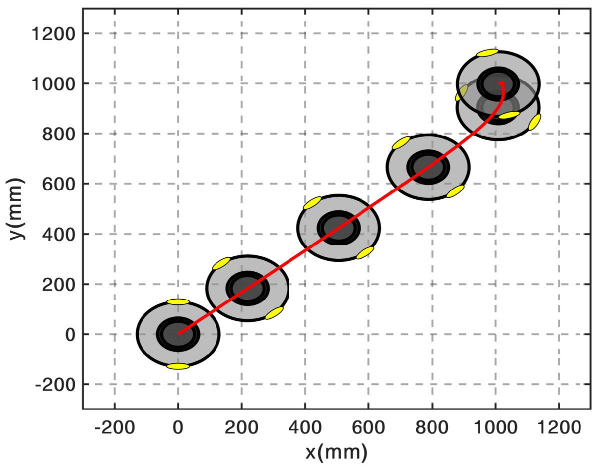
(d)


y(mm)
x(mm)

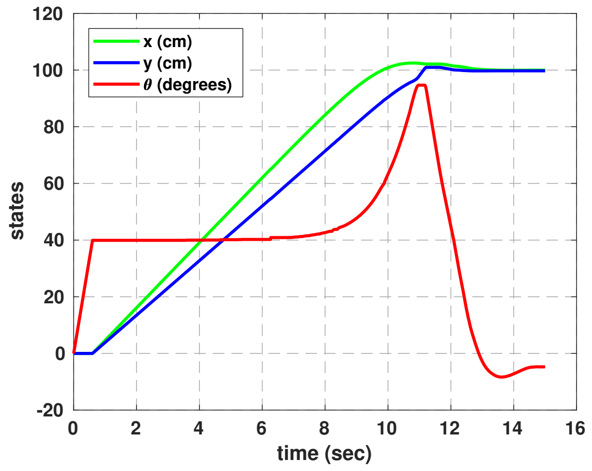
(e)


x (cm)
y (cm)
$\theta$ (degrees)
states
time (sec)

(f)

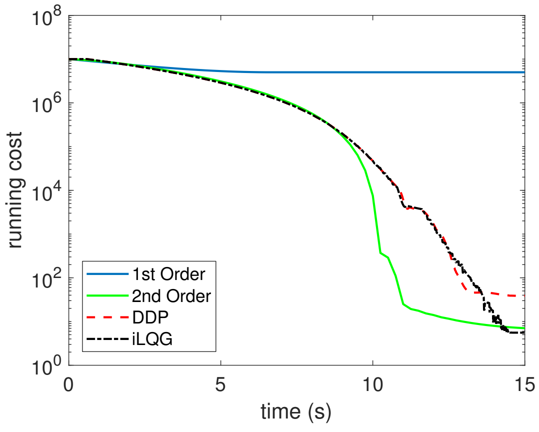


(g)

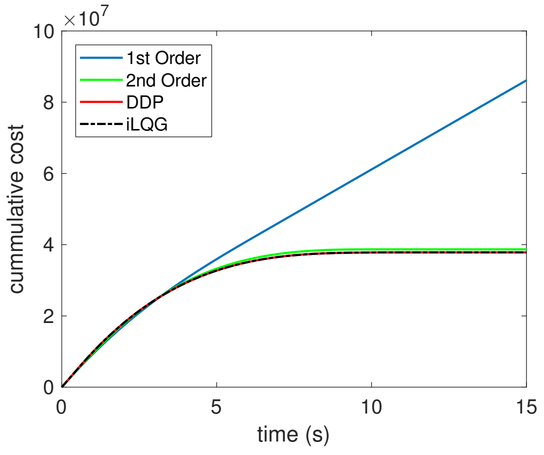


(h)

Figure 5.2. Differential drive using first-, second-order needle variation actions, iLQG, and DDP, from top to bottom. Snapshots of the system are shown at $t = 0, 2.5, 5, 7.5, 10,$ and $12.5$ sec. The target state is $[x_d, y_d, \theta_d] = [1000 \text{ mm}, 1000 \text{ mm}, 0]$.



(a)



(b)

Figure 5.3. Fig. 5.3a plots the running state cost; Fig. 5.3b plots the integrated (cumulative) cost, including the control cost. DDP and iLQG obtain the same cumulative cost, with slightly different trajectories (see Fig. 5.2). Second-order needle variation actions demonstrate improved convergence to the target over DDP and iLQG, despite optimizing over one single action at each iteration.

rigid body are controllable using first-order Lie brackets of the vector fields and help demonstrate Theorem 1. The underactuated dynamic model of a 3D rigid body serves to compare controls in (5.6) and (5.7), as well as make comparisons to other control techniques, in a more sophisticated environment. In all simulation results, I start with default control $v = 0$ and an objective function of the form

$$J(x(t)) = \frac{1}{2}\int_{t_o}^{t_f} \|\vec{x}(t)-\vec{x}_d(t)\|_Q^2 dt + \frac{1}{2}\|\vec{x}(t_f)-\vec{x}_d(t_f)\|_{P_1}^2,$$

where $\vec{x}_d$ is the desired state-trajectory, and $Q = Q^T \geq 0$, $P_1 = P_1^T \geq 0$ are metrics on state error.

### 5.3.1. 2D Kinematic Differential Drive

I use the differential drive system to demonstrate that first-order controls shown in (5.7) that are based only on the first-order sensitivity of the cost function (2.3) can be insufficient for controllable systems, contrary to controls shown in (5.6) that guarantee decrease of the objective for systems that are controllable using first-order Lie brackets (see Theorem 1).

The system states are its coordinates and orientation, given by $s = [x, y, \theta]^T$, with kinematic ($g = 0$) dynamics

$$f = r\begin{bmatrix} cos(\theta) & cos(\theta) \\ sin(\theta) & sin(\theta) \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix}\begin{bmatrix} u_R \\ u_L \end{bmatrix},$$

where $r = 3.6$ cm, $L = 25.8$ cm denote the wheel radius and the distance between them, and $u_R$, $u_L$ are the right and left wheel control angular velocities, respectively (these parameter values match the specifications of the iRobot Roomba). The control vectors $h_1$, $h_2$ and their Lie bracket term $[h_1, h_2] = 2\frac{r^2}{L}\Big[-sin(\theta), -cos(\theta)\Big]^T$ span the state space ($\mathbb{R}^3$). Therefore, from Theorem 1, there always exist controls that reduce the cost to first or second order.

Fig. 5.2 and 5.3 demonstrate how first-, second-order needle variations, iLQG, and DDP [**13**, **129**] perform on reaching a nearby target. I implement the iLQG and DDP algorithms to generate offline trajectory optimization solutions using the publicly available software.[4] Actions based on first-order needle variations (5.7) do not generate solutions that turn the vehicle, but rather drive it straight until the orthogonal displacement between the system and the target location is minimized. Actions based on second-order needle variations (5.6), on the other hand, converge successfully. The solutions differ from the trajectories computed by iLQG and DDP, despite using the same simulation parameters.

I note the fact that, besides the computational benefits, single-action approaches appear to be rich in information and perform comparably to offline schemes that attempt to minimize the objective by computing different control responses over the entire horizon. Given that the solutions of iLQG and DDP are very similar, and the fact that DDP is slower than iLQG due to expanding the dynamics to second order, I use only the iLQG algorithm as a means of comparison for the rest of the simulations presented in this work. The results in Fig. 5.2 based on second-order needle variations are generated in real time in MATLAB and approximately forty times faster than the iLQG implementation.

Fig. 5.4 shows a Monte Carlo simulation that compares convergence success using first- and second-order needle variations controls shown in (5.7) and (5.6), respectively,

[4]Available at http://www.mathworks.com/matlabcentral/fileexchange/52069-ilqg-ddp-trajectory-optimization.

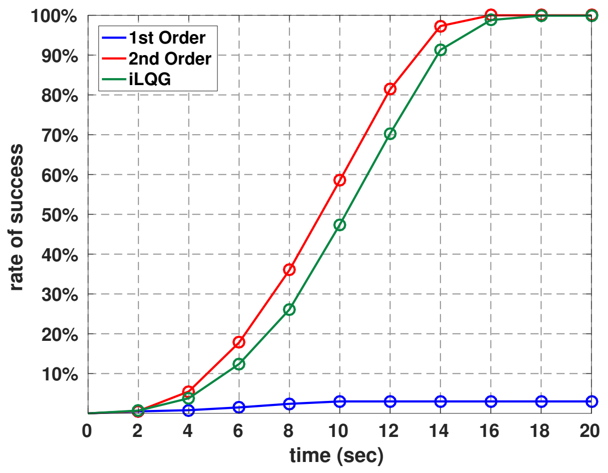


Figure 5.4. Convergence success rates of first- (5.7) and second-order (5.6) needle variation controls for the kinematic differential drive model. Simulation runs: 1000.

and iLQG. I sampled over initial coordinates $x_0, y_0 \in [-1500, 1500]$ mm using a uniform distribution and keeping only samples for which the initial distance from the origin exceeded $L/5$; $\theta_0 = 0$ for all samples. Successful samples are defined by being within $L/5$ from the origin with an angle $\theta < \pi/12$ within 60 seconds using feedback sampling rate of 4 Hz. Results are generated using $Q = \text{diag}(10, 10, 1000)$, $P_1 = \text{diag}(0, 0, 0)$, $T = 0.5$ s, $R = \text{diag}(100, 100)$ for (5.7), $R = \text{diag}(0.1, 0.1)$ for (5.6), $\gamma = -15$, $\lambda = 0.1$ and saturation limits on the angular velocities of each wheel $\pm 150/36$ mm/s for each control approach.[5] As shown in Fig. 5.4, the system always converges to the target using second-order needle variation actions, matching the theory.

[5]The metric on control effort is necessarily smaller for (5.6), due to parameter $\lambda$. The parameter is chosen carefully to ensure that control solutions from (5.6) and (5.7) are comparable in magnitude.

### 5.3.2. 3D Kinematic Rigid Body

The underactuated kinematic rigid body is a three dimensional example of a system that is controllable with first-order Lie brackets. To avoid singularities in the state space, the orientation of the system is expressed in quaternions [**160**, **161**]. The states are $s = [x, y, z, q_0, q_1, q_2, q_3]$, where $b = [x, y, z]$ are the world-frame coordinates and $q = [q_0, q_1, q_2, q_3]$ are unit quaternions. Dynamics $f = [\dot{b}, \dot{q}]^T$ are given by

$$\dot{b} = R_q v, \tag{5.8}$$

$$\dot{q} = \frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}\omega, \tag{5.9}$$

where $v$ and $\omega$ are the body frame linear and angular velocities, respectively [**162**]. The rotation matrix for quaternions is

$$R_q = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q1^2-q_2^2+q_3^2 \end{bmatrix}.$$

The system is kinematic: $v = F$ and $\omega = T$, where $F = (F_1, F_2, F_3)$ and $T = (T_1, T_2, T_3)$ describe respectively the surge, sway, and heave input forces, and the roll, pitch, and yaw input torques. I render the rigid body underactuated by removing the sway and yaw control authorities ($F_2 = T_3 = 0$).

The four control vectors span a four-dimensional space. First-order Lie bracket terms add two more dimensions to span the state space ($\mathbb{R}^6$) (the fact that there are seven states in the model of the system is an artifact of the quaternion representation; it does not affect controllability).

The vectors $h_1, h_2$, and $[h_2, h_3]$ span $\mathbb{R}^3$ associated with the world frame coordinates $\dot{x}, \dot{y}$, and $\dot{z}$. Similarly, vectors $h_3, h_4$, and $[h_4, h_3]$ span $\mathbb{R}^3$ associated with the orientation. Thereby, control vectors and their first-order Lie brackets span the state space and, from Theorem 1, optimal actions shown in (5.6) will always reduce the cost function (2.3).

To verify this prediction, I present the convergence success of the system on 3D motion. Using Monte Carlo sampling with uniform distribution, initial locations are randomly generated such that $x_0, y_0, z_0 \in [-50, 50]$ cm keeping only samples for which the initial distance from the origin exceeded 6 cm. I regard as a convergence success each trial in which the rigid body is within 6 cm to the origin by the end of 60 seconds at any orientation. Results are generated at a sampling rate of 20 Hz using $Q = 0$, $P_1 = \text{diag}(100, 200, 100, 0, 0, 0, 0)$, $T = 1.0$ s, $\gamma = -50000$, $\lambda = 10^{-3}$, $R = 10^{-6}\,\text{diag}(1, 1, 100, 100)$ for (5.6), and $R = \text{diag}(10, 10, 1000, 1000)$ for controls in (5.7). Controls are saturated at $\pm 10$ cm/s for the linear velocities and $\pm 10$ rad/s for the angular ones. Using 280 simulations over 24 seconds, 80% satisfy the success criterion within 12 seconds and 100% of trajectories satisfy the success criterion within 20 seconds. None of the simulations converge for the first-order needle variation controls, because they cannot displace the system in the $\hat{y}$ direction.

### 5.3.3. Underactuated Dynamic 3D Fish

I represent the three dimensional rigid body with states $s = [b,\ q,\ v,\ \omega]^T$, where $b = [x, y, z]$ are the world-frame coordinates, $q = [q_0, q_1, q_2, q_3]$ are the quaternions that describe the world-frame orientation, and $v = [v_x, v_y, v_z]$ and $\omega = [\omega_x, \omega_y, \omega_z]$ are the body-frame linear and angular velocities. The rigid body dynamics are given by $\dot{b}$ and $\dot{q}$ shown in (5.8) and (5.9) and

$$\begin{aligned} M\dot{v} &= Mv{\times}\omega{+}F, \\ J\dot{\omega} &= J\omega{\times}\omega{+}T, \end{aligned}$$

where the (experimentally determined) effective mass and moment of inertia of the rigid body are given by $M = \mathrm{diag}(6.04, 17.31, 8.39)$ g and $J = \mathrm{diag}(1.57, 27.78, 54.11)$ g·cm$^2$, respectively. This example is inspired by work in [**1**, **65**] and the parameters used for the effective mass and moment of inertia of a rigid body correspond to measurements of a fish. The control inputs are $F_2 = T_3 = 0$ and $F_3 \geq 0$.

The control vectors only span a four-dimensional space and, since they are state-independent, their Lie brackets are zero vectors. However, the Lie brackets containing the drift vector field $g$ (that also appear in the MIH expression) add from one to four (depending on the states) independent vectors such that control solutions in (5.6) guarantee decrease of the cost function (2.3) for a wider set of states than controls in (5.7).

Simulation results based on Monte Carlo sampling are shown in Fig. 5.5. Initial coordinates $x_0, y_0, z_0$ are generated using a uniform distribution in $[-100, 100]$ cm, discarding samples for which the initial distance to the origin is less than 15 cm. Successful

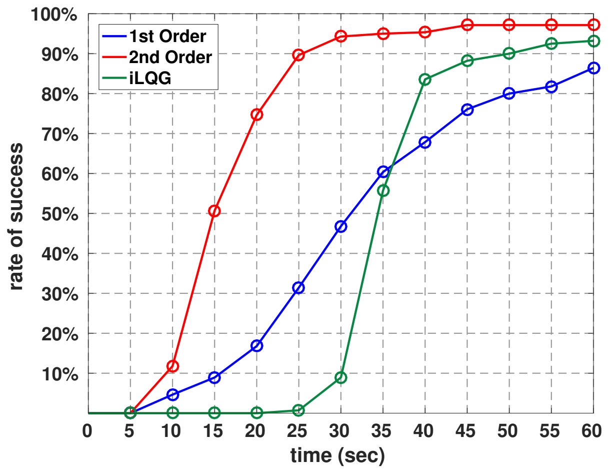


Figure 5.5. Convergence success rates of first- and second-order needle variation controls—(5.7) and (5.6), respectively—and iLQG for the underactuated *dynamic* vehicle model. Simulation runs: 280

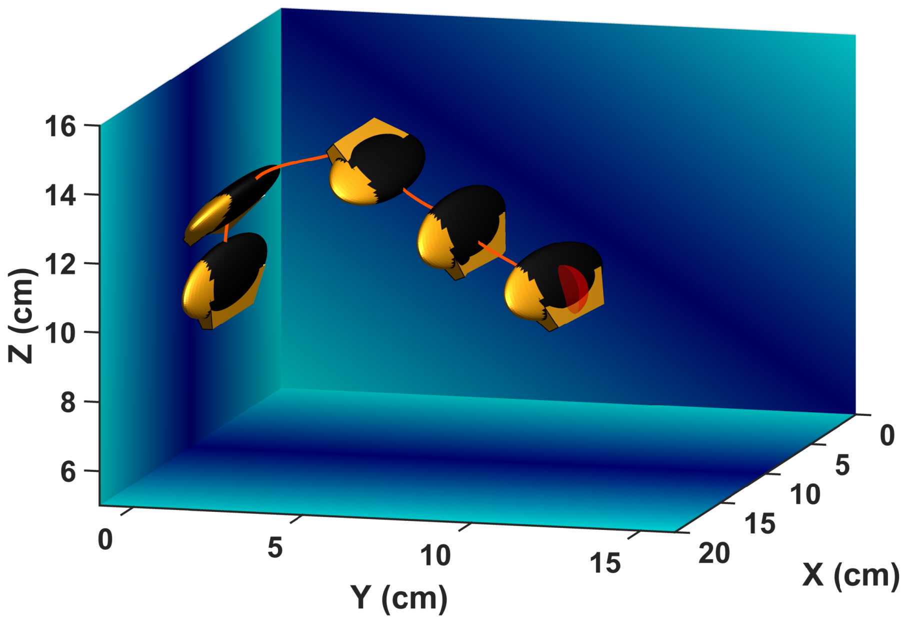
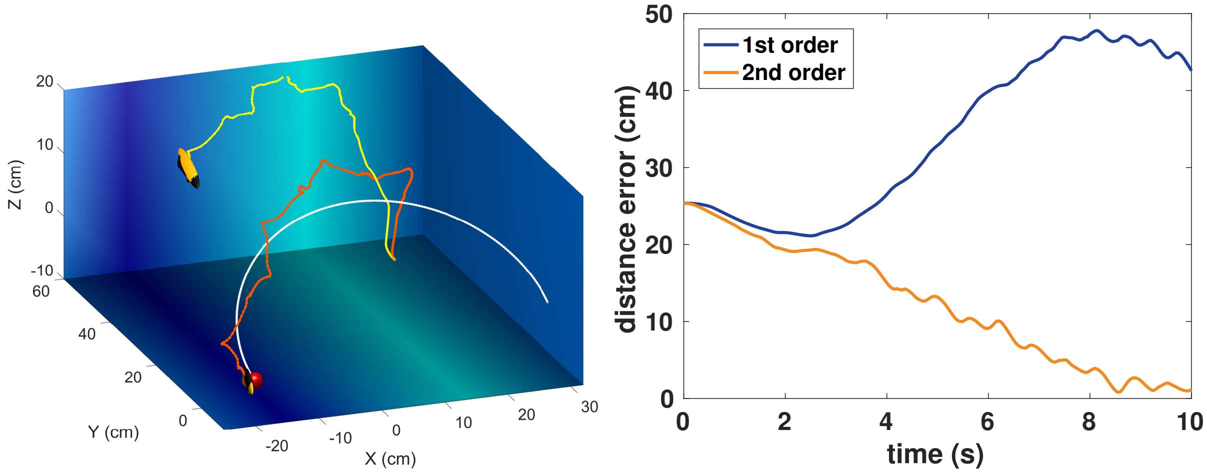


Figure 5.6. Snapshots of a parallel displacement maneuver using an underactuated dynamic vehicle model with second-order controls given by (5.6); first-order solutions (5.7) are singular throughout the simulation. Animation of these results is available in Extension 3.



(a) (b)

Figure 5.7. Tracking performance of the same system in the presence of +10 cm/s $\hat{y}$ fluid drift. The yellow system corresponds to first-order needle variation actions; the red one to second order. The target trajectory (red ball) is indicated with white traces over a 10-second simulation. Fig. 5.7b shows the error distance as a function of time, clearly demonstrating the advantage of the second-order approach. Animation of these results is available in Extension 4.

trials are the ones for which, within a simulation window of 60 seconds, the system approached within 5 cm to the origin (at any orientation) and whose magnitude of the linear velocities is, at the same time, less than 5 cm/s. Results are generated at a sampling rate of 20 Hz using $T$=1.5 s, $P_1$=0, $Q=\frac{1}{200}\text{diag}(10^3,10^3,10^3,0,0,0,0,1,1,1,2\cdot 10^3,10^3,10^3)$, $\gamma$=−5, $R=\text{diag}(10^3,10^3,10^6,10^6)$ for (5.7), $R=\frac{1}{2}\text{diag}(10^{-6},10^{-6},10^{-3},10^{-3})$ for (5.6), and $\lambda = 10^{-4}$. The same control saturations ($F_1 \in [-1,1]$ mN, $F_3 \in [0,1]$ mN, $T_1 \in [-0.1,0.1]\,\mu$N·m, and $T_2 \in [-0.1,0.1]\,\mu$N·m) are used for all simulations of the dynamic 3D fish. As shown in Fig. 5.5, controls computed using second-order needle variations converge faster than those based on first-order needle variations, and 97% of trials converge within 60 seconds.

Both methods converge over time to the desired location; as the dynamic model of the rigid body tumbles around and its orientation changes, possible descent directions of the cost function (2.3) change and the control is able to push the system to the target. Controls for the first-order needle variation case (5.7) are singular for a wider set of states than second-order needle variation controls (5.6) and, for this reason, they benefit more from tumbling. In a 3D parallel maneuver task, only second-order variation controls (5.6) manage to provide control solutions through successive heave and roll inputs, whereas controls based on first-order sensitivities (5.7) fail (see Fig. 5.6).

As controls in (5.6) are non-singular for a wider subset of the configuration state space than the first-order solutions in (5.7), they will provide more actions over a period of time and keep the system closer to a time-varying target. Fig. 5.7a demonstrates the superior trajectory tracking behavior of controls based on (5.6) in the presence of +10 cm/s $\hat{y}$ fluid drift. The trajectory of the target is given by $[x, y, z]$=$[\cos(\frac{3\mathrm{t}}{10})\,(20{+}10\cos(\frac{\mathrm{t}}{5})),\ \sin(\frac{3\mathrm{t}}{10})\,(20 + 10\cos(\frac{\mathrm{t}}{5})),\ 10\sin(\frac{2\mathrm{t}}{5})]$, with $T$=2 s, $\lambda$=0.01, $\gamma$=$-50000$, $Q$=diag(10,10,10,0,0,0,0,0,0,0,1,1,0.1), $P_1$=diag(10,10,10,0,0,0,0,0,0,0,0,0,0), $R$=diag($10^3$,$10^3$,$10^6$,$10^6$) for (5.7), and $R$=diag(10,10,$10^4$,$10^4$) for (5.6). The simulation runs in real time using a C++ implementation on a laptop with Intel® Core™ i5-6300HQ CPU @2.30GHz and 8GB RAM.

The drift is known for both first- and second-order systems and accounted for in their dynamics in the form of $\dot{b} = \dot{b} + \dot{b}_{\text{drift}}$, where $\dot{b}_{\text{drift}}$ is a vector that points in the direction of the fluid flow. Simulation results demonstrate superior tracking of second-order needle variation controls that manage to stay with the target, whereas the system that corresponds to first-order needle variation controls is being drifted away by the flow.

I also tested convergence success of the +10 cm/s $\hat{y}$ drift case. Initial conditions $x, y, z$ are sampled uniformly from a 30 cm radius from the origin, discarding samples for which the initial distance is less than 5 cm. I consider samples to be successful if, during 60 seconds of simulation, they approached the origin within 5 cm. Out of 500 samples, controls based on second-order variations converged 91% of the time (with average convergence time of 5.87 s), compared to 89% for first-order actions (with average convergence time of 9.3 s). Simulation parameters are $T$=1 s, $\gamma$=$-$25000, $Q$=$10^{-3}$diag(10,10,10,0,0,0,0,1,1,1,1,1,1), $P_1$=diag(100,100,100,0,0,0,0,$\frac{1}{2}$,$\frac{1}{2}$,$\frac{1}{2}$,0,0,0), $\lambda$=$10^{-4}$, $R$=diag(0.1,0.1,$10^4$,$10^4$) for (5.7), and $R$=$\frac{1}{2}$diag($10^{-5}$,$10^{-5}$,1,1) for (5.6).

## 5.4. Discussion

This chapter presents a needle variation control synthesis method for nonlinearly controllable systems that can be expressed in control affine form. Control solutions provably exploit the nonlinear controllability of a system and, contrary to other nonlinear feedback schemes, have formal guarantees with respect to decreasing the objective. By optimally perturbing the system with needle actions, the proposed algorithm avoids the expensive iterative computation of controls over the entire horizon that other NMPC methods use and is able to run in real time for the systems considered here.

Simulation results on three underactuated systems compare first-order needle variation controls, second-order needle variation controls, and iLQG controls and demonstrate the superior convergence success rate of the proposed feedback synthesis. Because second-order needle variation actions are non-singular for a wider set of the state space than controls based on first-order sensitivity, they are also more suitable for time-evolving objectives, as demonstrated by the trajectory tracking examples in this chapter. Second order needle variation controls are also calculated at little computational cost and preserve control effort. These traits, demonstrated in the simulation examples of this chapter, render feedback synthesis based on second- and higher-order needle variation methods a promising alternative feedback scheme for underactuated and nonlinearly controllable systems.

CHAPTER 6

# Real-Time Obstacle Avoidance with Second-Order SAC

This chapter exploits the controllability-based guarantees for convergence presented in Chapter 5 to achieve collision-avoidance. By accounting for obstacles in the objective, second-order SAC provably guarantees that the agent reaches the target in a collision-free manner in the presence of obstacles without relying on predefined trajectories. Using systems that are controllable with first-order Lie brackets, this chapters demonstrates that the proposed controller finds collision-free solutions in the presence of both static and moving obstacles.

## 6.1. Motion Planning for Controllable Systems in the Presence of Obstacles

Controllability in its classical sense concerns itself with the existence of an action trajectory that can move the agent to a desired state, subject to the differential constraints posed by the dynamics, in the absence of obstacles. Controllability is an inherent property of the dynamics and reveals all allowable motion, disregarding the presence of physical constraints in the environment.

Feasible path planning amidst obstacles is often treated separately from the optimal control problem. Most commonly, feasible trajectories are generated with efficient path planners, such as rapidly-exploring random tree (RRT) and probabilistic road map (PRM) methods [**163**, **164**]. The distinction between path planning and optimal control can be seen in work by [**46**, **48**] that uses such motion planners to generate trajectories among obstacles and then uses them as a reference to compute the optimal control. In this setting, nonholonomic motion consists of two stages, the path planning and the feedback synthesis that tracks the feasible trajectory.

Another solution to obstacle avoidance in motion planning is the use of barrier certificates [**165**]. Barrier certificates provably enforce collision-free behavior by minimally perturbing, in a least-squares sense, the control response in order to satisfy safety constraints. Feedback synthesis proceeds without accounting for obstacles and solutions are modified, only when necessary, via a quadratic program (QP) subject to constraints that ensure collision avoidance [**166–170**].

Additional solutions to obstacle avoidance include compensating functions that eliminate local minima in the objective caused by the obstacles [**171**], as well as designing navigation functions using inverse Lyapunov expressions [**172**]. The former method computes the local minima in the objective and constructs a plane surface function to remove them and make the objective convex. This process can be cumbersome, as one would have to locate all local minima in the objective induced by the obstacles and then calculate the compensating function. On the other hand, navigation functions, described in [**172**, **173**], are globally convergent potential functions and are system-specific.

Several of these collision-avoidance algorithms are not system-specific and could be implemented with the proposed controller, outlined in Section 5.2. In simulation results, presented in Section 6.2, I show collision-avoidance using only penalty functions in the objective, demonstrating that the proposed controller succeeds in tasks (collision avoidance) that traditionally require sophisticated treatment.

I use Theorem 1 to show that the proposed needle-variation controller will always converge to the global minimizer for convex functions. This statement is true independent of the presence of obstacles.

**Theorem 2.** *Consider dynamics given by* (8.21)*, a trajectory described by state and control* $(x,v)\neq(x^*,v^*)$*. Let* $\tilde{x}^k$ *describe the trajectory generated after* $k$ *iterations of Algorithm I. Further let* $x\in\mathcal{X}_{free}\forall\ t\in[t_o,t_f]$*, where* $\mathcal{X}_{free}\subset\mathcal{X}$ *denotes the collision-free part of the state-space. Consider an objective* (2.3) *that satisfies Assumption 2 and whose running cost term penalizes collisions, such that* $J(\tilde{x}^k)>J(x)$ *if* $\exists\ t\in[t_o,t_f]$ *where* $\tilde{x}\notin\mathcal{X}_{free}$*. If the objective* $J(x)$ *is convex with respect to the state* $x$ *in the unconstrained sense and the control vectors* $h_i$ *and the Lie brackets* $[h_i,h_j]$ *and* $[g,h_i]$ *span the state space* $(\mathbb{R}^N)$*, then the sequence of solutions* $\{\tilde{x}^k\}$ *generated by Algorithm I converges to* $x^*$*. Further,* $\tilde{x}^k\in\mathcal{X}_{free}\forall\ k\in\mathbb{R}^+$*.*

Proof. Algorithm I constructs control responses out of the first- and second-order mode insertion gradients. By extension of Theorem 1, it can guarantee to reduce the objective with each iteration (for some control $u$ of duration $\lambda$). Therefore,

$$J(\tilde{x}^k)>J(\tilde{x}^{k+1})\geq J_{min}, \tag{6.1}$$

where $J_{min}=J(x^*)$ is the (only) minimizer of the convex objective. It follows that,

$$\lim_{k\to\infty} J(\tilde{x}^k)=J_{min}. \tag{6.2}$$

Further, assume that there exists $t{\in}[t_o,t_f]$ and $k{\in}\mathbb{R}^+$ such that $\tilde{x}^k{\notin}\mathcal{X}_{free}$. Then $J(\tilde{x}^k){>}J(x)$, which contradicts (6.1). Using proof by contradiction, I conclude that

$$\tilde{x}^k{\in}\mathcal{X}_{free}\ \forall\ t{\in}[t_o,t_f],\ \forall\ k{\in}\mathbb{R}^+. \tag{6.3}$$

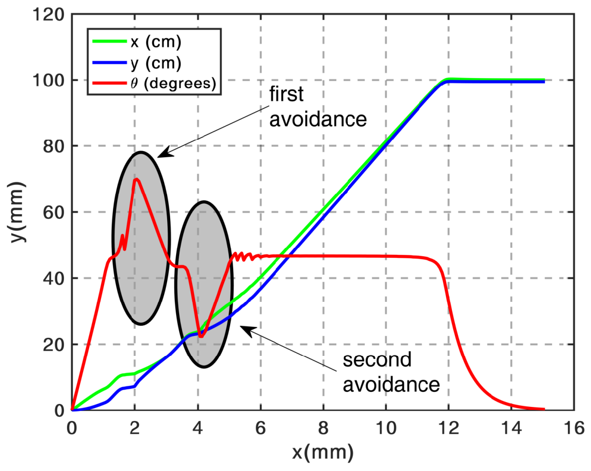
Assuming that a collision-free path exists between the agent and the target, it is straightforward that the minimizer trajectory is the target's location. Therefore, from (6.2) and (6.3), Algorithm I generates a sequence $\{\tilde{x}^k\}$ that converges to the target safely.[1] $\square$

With regards to Theorem 2, I should alert the reader that Algorithm I may not guarantee collision-avoidance if the default trajectory is not collision-free, that is if there exists $t{\in}[t_o,t_f]$ such that $x{\notin}\mathcal{X}_{free}$. Further, the result of Theorem 2 can generalize to non-convex functions that have only one minimum.

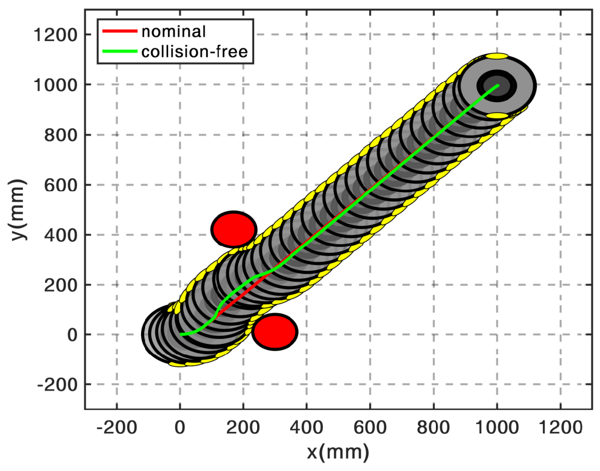
## 6.2. Simulation Results

Next, I illustrate the performance of the algorithm in the presence of obstacles. In all simulations, obstacles are considered in the objective in the form of a penalty function. In Fig. 6.1, I test the system in the same task as Fig. 5.2 in the presence of two obstacles, indicated with red spheres. In comparison with Fig. 5.2f, it is worth noting the two angle peaks, corresponding to each obstacle. After passing the obstacles, the system recovers the same angle profile.

[1]Although control responses are constructed from a second-order Taylor series approximation of the objective, the iterated sequence is guaranteed to decrease the real cost function at each iteration. If the real cost function is convex with respect to the state (in the unconstrained sense), the iterated sequence will converge towards the only minimizer. Using a sufficient descent condition, the algorithm is guaranteed to converge to a point where the unconstrained derivative of the objective is zero (i.e., $D_xJ(x){=}0$), which, from Assumption 4, only happens at the global minimizer.



(a)



(b)

Figure 6.1. Differential drive using second-order needle variation actions in the presence of obstacles. Fig. 6.1b shows the deviation from the nominal trajectory that is the solution to the no-obstacle task. The system performs two maneuvers to avoid each obstacle. These are evident in the angle deviation (compare to Fig. 5.2c).

Fig. 6.2 shows more complicated maneuvers using controls from (5.6). The controller, without relying on a motion planner, is able to generate collision-free trajectories and safely converge to the target in all cases. These simulations also demonstrate another aspect of Algorithm I. The differential drive always drives up to an obstacle and then narrowly maneuvers around, instead of preparing a turn earlier on. This behavior is to be expected of needle variation actions that instantly reduce the cost.

I next use the more complicated scenario of Fig. 6.2d to evaluate the second-order expansion of the objective, shown in (5.4), across the state-space (see Fig. 6.3). The first- and second-order mode insertion gradients are computed based on the second-order needle variation controls from (5.6). States are sampled in the space for $x$ and $y$ in increments of 5 mm, with $\theta$=0 everywhere. These results correspond to $\lambda$=0.001. The horizontal discontinuity that appears around $y$=750 mm is believed to be due to the effect of the

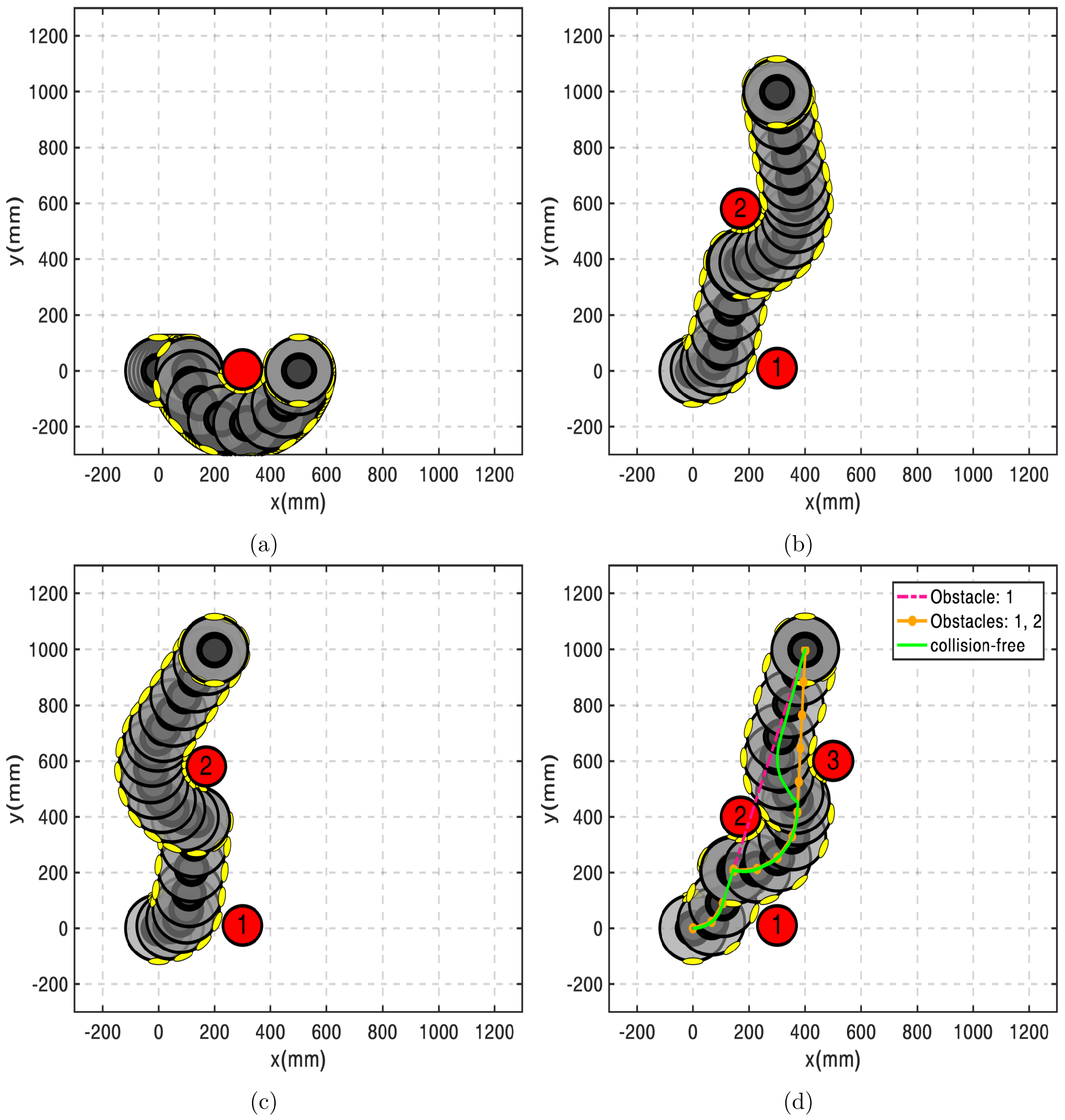


Figure 6.2. Trajectories of the differential drive in the presence of obstacles. Fig. 6.2d compares the solution to the trajectories generated when considering only a) obstacle 1, and b) obstacles 1 and 2, both of which collide with the obstacles. Simulations run in real time in MATLAB.

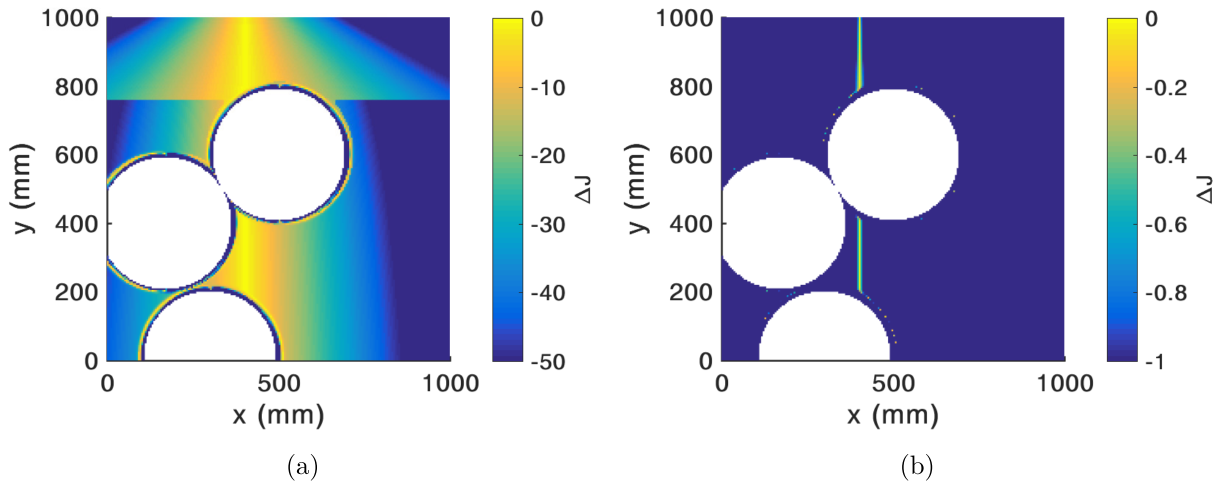
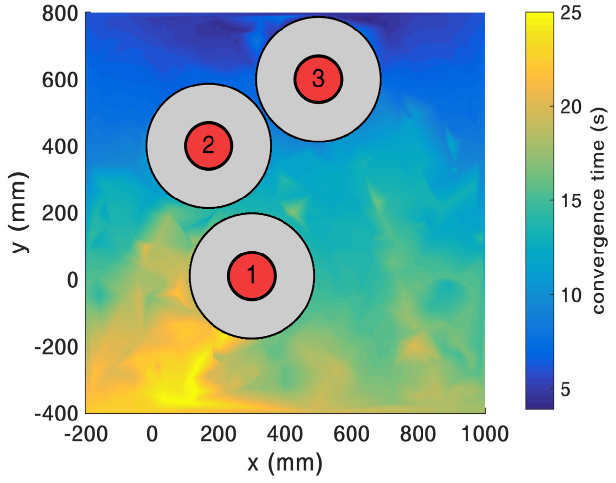


Figure 6.3. Cost reduction $\Delta J$, modeled after (5.4), for sampled $x$ and $y$ in the presence of obstacles, given second-order needle variation controls. The first- and second-order mode insertion gradients are evaluated with the controls from (5.6). Figures 6.3a and 6.3b are identical, but shown over a different range to illustrate that even when looking at small variations of the first-order mode insertion gradient, the second-order method is reliably negative. The bright vertical line in Fig. 6.3b is vertically aligned with the target located at [400 mm, 1000 mm], where first-order solutions are singular. No data are sampled inside the white circles, as these indicate the infeasible occupied region.

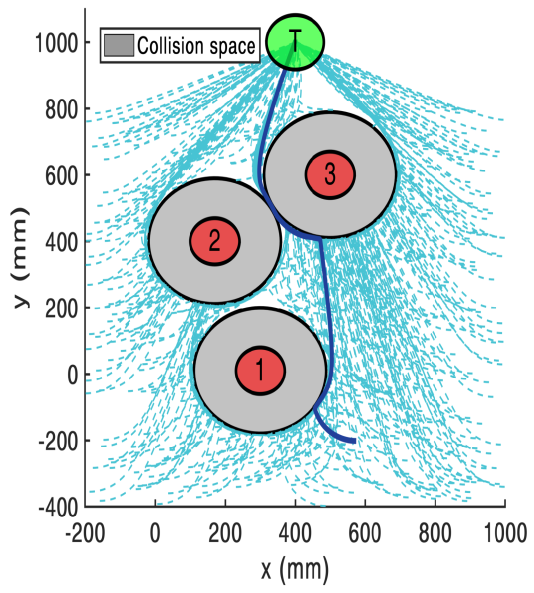
penalty functions. As Fig. 6.3 indicates, the change in cost is always negative, verifying Theorem 1, even in the presence of obstacles.

I further use a Monte Carlo simulation of 500 trials on the initial conditions to test convergence success (Fig. 6.4). I sample initial conditions [$x$,$y$] from a uniform distribution in [-200 mm, 1000 mm] $\times$ [-400 mm, 800 mm], where $\theta_o$=0 in all cases. All trials converged within 25 seconds.

Last, I test the differential drive in the presence of moving obstacles (Fig. 6.5). The controller is again able to avoid collision and converge to the target, without relying on additional motion planning techniques. The feedback rate used is 20 Hz and the trajectory of the obstacles is known to the agent throughout the time horizon. In these simulations,



(a)



(b)

Figure 6.4. Performance of second-order needle variation actions in the presence of static obstacles. The controller is able to converge to the target for all 500 trials and avoid collisions. Fig. 6.4a is an interpolated heat map that indicates the time to convergence as a function of initial position; Fig. 6.4b shows the trajectories followed by the center of mass of the agent. The gray area indicates the collision space, taking into account the width of the differential drive. (the simulation runs in real time in MATLAB). For visualization, watch Extension 1.

Table 6.1. **Table of Multimedia Extensions**

| Extension | Type | Description |
|---|---|---|
| 1 | Video | Collision-free convergence in the presence of static obstacles from random initial conditions |
| 2 | Video | Collision-free convergence in the presence of moving obstacles |
| 3 | Video | Parallel maneuver of 3D dynamic fish. |
| 4 | Video | Underactuated tracking of dynamic 3D Fish in the presence of drift. |

$T$=0.3 s. The multimedia extensions to this article can be found online by following the hyperlinks shown in Table 6.1.

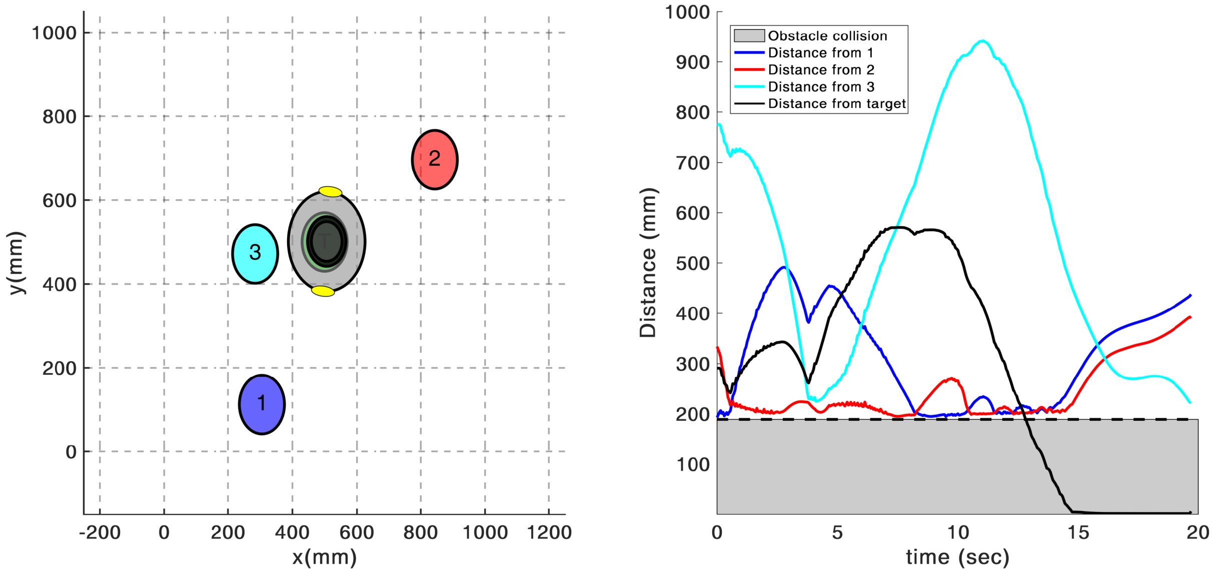


Figure 6.5. Performance of second-order needle variation actions in the presence of three moving obstacles. The left figure shows a snapshot of the simulation; the right figure plots the distance of the agent from each object and the target, where the gray area indicates the threshold minimum distance to avoid collision with the obstacles. The controller converges to the target in a collision-free manner (the simulation runs in real time in Matlab). For visualization, watch Extension 2.

# Part 2

# Physics-Based System Identification and Data-Driven Control

CHAPTER 7

# Background

## 7.1. Koopman Operator

The Koopman operator $\mathcal{K}$ linearly evolves functions of the states $s(t) \in \mathcal{S} \subseteq \mathbb{R}^N$ (i.e. $\Psi(s(t))$, commonly referred to as observables or basis functions) without loss of accuracy [**72**]. Given general nonlinear dynamics of the form

$$s_{k+1} = F(s_k), \tag{7.1}$$

where $F$ is the flow map, the Koopman operator advances the observables with the flow of the dynamics:

$$\mathcal{K}\Psi = \Psi \circ F. \tag{7.2}$$

Thus, it advances measurements of the states linearly. That is,

$$\frac{d}{dt}\Psi(s) = \mathcal{K}\Psi(s) \quad \text{and} \quad \Psi(s_{k+1}) = \mathcal{K}_d \Psi(s_k), \tag{7.3}$$

where $\mathcal{K}$ and $\mathcal{K}_d$ are the continuous-time and discrete-time operators, respectively, related by $\mathcal{K} = \log(\mathcal{K}_d)/\Delta t$ [**174**]. Although a linear representation, the Koopman operator evolves nonlinear dynamics with full fidelity throughout the state space, contrary to methods that locally linearize dynamics around a point or a trajectory. To allow for the effect of actuation, (7.3) is modified such that the observables include control terms $u$ as well [**73**, **95**]. For a more comprehensive review of the Koopman operator, I refer the reader to [**75**].

Expressing nonlinear systems in a linear manner is a desirable property for many reasons, such as investigating the global stability of a system [**175**], or extending the local linearization around a point to the whole basin of attraction [**176**]. In addition to studying the behavior of complex systems, the Koopman framework enables the use of linear optimal control for original nonlinear dynamics. Unfortunately, the infinite-dimensional nature of the Koopman operator makes practical use prohibitive.

## 7.2. Koopman Invariant Subspaces

There exist nonlinear systems that admit a finite-dimensional linear Koopman representation. Work in [**85**] analytically derives such Koopman invariant subspaces for nonlinear systems with a specific polynomial structure, whereas the authors in [**86**–**88**, **177**–**182**] have tried to identify such subspaces from data. For example, work in [**180**] identifies eigenfunctions that may be different from the system states and synthesizes the control objective in terms of the new coordinates. However, not only do such eigenfunctions not always exist, they may also not allow the recovery of the system states. In these studies, the authors demonstrate that the LQR control based on the linear representation can outperform LQR control calculated based on the original, nonlinear dynamics. Unfortunately, Koopman invariant subspaces have only been found for a few systems, mentioned above. In fact, there can be no finite-dimensional invariant subspace that includes the states for systems with multiple fixed points [**85**].

In the absence of a finite-dimensional Koopman invariant subspace, a linear propagation of states will induce errors. Regardless, the benefits of a linear model motivate obtaining an approximation to the Koopman operator that will evolve the nonlinear system with acceptable accuracy. Recent studies [**56**, **57**, **183**] use data-driven regression schemes to approximate the infinite-dimensional operator $\mathcal{K}$ with a finite-dimensional representation $\tilde{\mathcal{K}}$ [**57**, **73**, **82**].

## 7.3. Data-Driven Approximations of Koopman Operators

To obtain an approximation to the Koopman operator, $\tilde{\mathcal{K}} \in \mathbb{R}^{w\times w}$, one can choose a set of observable functions $\Psi(s)=[\psi_1(s),\psi_2(s),...,\psi_w(s)] : \mathbb{R}^N \mapsto \mathbb{R}^w$ (which can include the states $s$ themselves) and compute a data-driven model that best fits measurements. There are several methods to approximate Koopman operators, such as DMD [**89**], EDMD [**82**,**90**], Hankel-DMD [**91**], closed-form solutions [**92**, **93**], or regression techniques, such as Least Absolute Shrinkage and Selection Operator (LASSO) regression [**57**, **184**]. For efficiency purposes, researchers typically consider the least-squares solutions of the local one-time-step error across $P$ measurements given by

$$\tilde{\mathcal{K}}_d^{*}=\underset{\tilde{\mathcal{K}}_d}{\operatorname{argmin}}\ \frac{1}{2}\sum_{k=1}^{P}\|\Psi(s(t_k+\Delta t),u(t_k+\Delta t))-\tilde{\mathcal{K}}_d\Psi(s(t_k),u(t_k))\|^2, \tag{7.4}$$

where $\tilde{\mathcal{K}}_d \in \mathbb{R}^{W\times W}$ is the finite-dimensional approximation of the Koopman operator and $\Psi(s(t_k)) : \mathbb{R}^N \mapsto \mathbb{R}^W$ are the observables. Each measurement $k$ consists of the initial state $s(t_k)$, final state $s(t_k+\Delta t)$ and the actuation applied at the same instants, $u(t_k)$ and

$u(t_k{+}\Delta t)$, respectively. Equation (7.4) has a closed-form solution given by

$$\tilde{\mathcal{K}}_d^* = \mathcal{A}\mathcal{G}^\dagger, \tag{7.5}$$

with

$$\mathcal{A} = \sum_{k=1}^{P} \Psi(s(t_k{+}\Delta t), u(t_k{+}\Delta t))\Psi(s(t_k), u(t_k))^T$$

and

$$\mathcal{G} = \sum_{k=1}^{P} \Psi(s(t_k), u(t_k))\Psi(s(t_k), u(t_k))^T$$

where $^\dagger$ refers to the Moore-Penrose pseudoinverse. Measurements need not be from a single trajectory. Note, however, that the time spacing $\Delta t$ between measurements $s(t_k)$ and $s(t_k{+}\Delta t)$ must be consistent for all $P$ training measurements.

Koopman operators offer an easily implementable system identification framework that is conducive to linear control tools for nonlinear dynamical systems. The linear representation makes it easier to analyze the properties of the underlying system, such as model accuracy or regions of attraction [**175**, **176**]. Further, Koopman-operator-based control can even outperform feedback that utilizes full knowledge of the nonlinear dynamics [**85**, **86**].

The data-driven approximation of the Koopman operator is not inherently different from other system identification techniques. In fact, the Koopman operator can be approximated using any of the standard regression methods, such as ridge or lasso regression [**185**, **186**]. More importantly, contrary to standard system identification tools that may try to estimate unknown parameters or, more generally, the nonlinear dynamics of a system [**66**, **187**], the Koopman operator framework places the system identification task in the context of seeking linear transformations of the states, which is useful for control [**188**–**190**] and other purposes.

## 7.4. LQR Control of Nonlinear Dynamics Using Koopman Operators

Consider a linear system with states $s(t)\in\mathbb{R}^N$, control $u(t)\in\mathbb{R}^M$, and a discrete-time performance objective

$$J=\frac{1}{2}\sum_{t_k=0}^{\infty}\|s(t_k)-s_{des}(t_k)\|_Q^2+\|u(t_k)\|_R^2, \tag{7.6}$$

where $Q\succeq 0\in\mathbb{R}^{N\times N}$ and $R\succ 0\in\mathbb{R}^{M\times M}$ are weights on the deviation from the desired states and the applied control, respectively. Next, I use the Koopman operator dynamics to design an equivalent objective function and a control response for the original nonlinear system.

Further, consider the Koopman representation

$$\Psi(s(t_k+\Delta t),u(t_k+\Delta t))=\tilde{\mathcal{K}}_d\Psi(s(t_k),u(t_k)). \tag{7.7}$$

For simplicity, let $\Psi(s(t_k),u(t_k))=[\Psi_s^T(s(t_k)),\Psi_u^T(u(t_k))]^T$, where $\Psi_s(s(t_k))\in\mathbb{R}^{w_s}$ are the functions that depend on the states $s$ and $\Psi_u(u(t_k))\in\mathbb{R}^{w_u}$ are the functions that depend on the input $u$, where $w=w_s+w_u$. Using this notation, I rewrite (7.7) as

$$\begin{bmatrix}\Psi_s(s(t_k+\Delta t))\\ \Psi_u(u(t_k+\Delta t)\end{bmatrix}\approx\begin{bmatrix}A & B\\ \cdot & \cdot\end{bmatrix}\begin{bmatrix}\Psi_s(s(t_k))\\ \Psi_u(u(t_k))\end{bmatrix}, \tag{7.8}$$

where $(\cdot)$ is used to indicate the terms associated with predicting the evolution of control, which is of no interest in this thesis as it will determined by the feedback policy. The terms $A{\in}\mathbb{R}^{w_s\times w_s}$ and $B{\in}\mathbb{R}^{w_s\times w_u}$ are submatrices of $\tilde{\mathcal{K}}_d$ that describe the dynamics of the state-dependent functions and change only when $\tilde{\mathcal{K}}_d$ is updated. In addition, $(\cdot)$ is used to indicate the terms associated with predicting the evolution of control, which is of no interest in this thesis as it will determined by the feedback policy. Note that even when there are states coupled with control, dynamics can always be transformed to a control-affine form. When necessary, this is achieved with a variable replacement $u{\to}v$ and by introducing additional variables $\dot{v}{=}u$ such that the control becomes the derivative of the applied actuation [**180**]. In this thesis, I choose $\Psi_u(u(t_k)){=}u(t_k)$ to ensure that control appears linearly in the model. Then, the dynamics of the Koopman state-dependent basis functions are

$$\Psi_s(s(t_k{+}\Delta t)){\approx}A\Psi_s(s(t_k)){+}Bu(t_k). \tag{7.9}$$

Given the Koopman dynamics (7.9), I choose the performance objective

$$J_{\tilde{\mathcal{K}}}{=}\frac{1}{2}\sum_{t_k=0}^{\infty}\|\Psi_s(s(t_k)){-}\Psi_s(s_{des}(t_k))\|^2_{Q_{\tilde{\mathcal{K}}}}{+}\|u(t_k)\|^2_R, \tag{7.10}$$

where $Q_{\tilde{\mathcal{K}}}{\succeq}0{\in}\mathbb{R}^{w_s\times w_s}$ penalizes the deviation from the desired observable functions $\Psi_s(s_{des}(t_k)$. I let the first $N$ observables be the original states $s$ and set

$$Q_{\tilde{\mathcal{K}}}{=}\begin{bmatrix}Q & 0\\ 0 & 0\end{bmatrix}, \tag{7.11}$$

so that a meaningful comparison can be made with regards to the original nonlinear system and the associated objective function shown in (7.6). The Koopman representation is conducive to linear quadratic regulator (LQR) feedback of the form

$$u_(t_k) = -K_{LQR}(\mathbf{\Psi}_s(s(t_k)) - \mathbf{\Psi}_s(s_{des}(t_k))), \tag{7.12}$$

where $K_{LQR} \in \mathbb{R}^{M \times w_s}$, the LQR gains, can be readily calculated from $A$,$B$ in (7.8) [**191**, **192**]. Note that, for given LQR gains, the control is updated using only the functions $\mathbf{\Psi}_s(s(t_k))$, leading to minimal computation. For more details on the control policy used for the Koopman representation, the reader can refer to [**92**]. Last, I want to emphasize that the proposed approach for synthesizing data-driven Koopman representations can be used with different feedback schemes, such as MPC control.

CHAPTER 8

# Data-Driven Identification and Control Using Koopman Operators

This chapter presents a generalizable methodology for data-driven identification of nonlinear dynamics that bounds the model error in terms of the prediction horizon and the magnitude of the derivatives of the system states. Using higher-order derivatives of general nonlinear dynamics that need not be known, I construct a Koopman operator-based linear representation and utilize Taylor series accuracy analysis to derive an error bound. The resulting error formula is used to choose the order of derivatives in the basis functions and obtain a data-driven Koopman model using a closed-form expression that can be computed in real time. Using the inverted pendulum system, I illustrate the robustness of the error bounds given noisy measurements of unknown dynamics, where the derivatives are estimated numerically. When combined with control, the Koopman representation of the nonlinear system has marginally better performance than competing nonlinear modeling methods, such as SINDy and NARX. In addition, as a linear model, the Koopman approach lends itself readily to efficient control design tools, such as LQR, whereas the other modeling approaches require nonlinear control methods. The efficacy of the approach is further demonstrated with simulation and experimental results on the control of a tail-actuated robotic fish. Experimental results show that the proposed data-driven control approach outperforms a tuned PID (Proportional Integral Derivative) controller and that updating the data-driven model online significantly improves performance in the presence of unmodeled fluid disturbance. This chapter is complemented with a video: https://youtu.be/9_wx0tdDta0.

## 8.1. Synthesis of Basis Functions for Error-Bounded Koopman Representation

This section motivates using higher-order derivatives of nonlinear dynamics to populate the observables of an approximate Koopman operator. The benefits of a derivative-based representation are twofold. First, subject to a finite number of basis functions, it allows one to best capture, locally in time, nonlinear dynamics. For systems that admit a finite-dimensional Koopman invariant subspace, it is straightforward to show that the terms in the observable functions $\Psi(s)$ capture all higher-order derivatives of the original states. This is the reason why the linear representation matches the nonlinear dynamics with no error. This is also true for the invariant subspaces found in [**85**], where the Koopman observable functions are populated using the Carleman linearization approach [**193**–**195**]; for the polynomial systems considered there, the observable functions correspond to the higher-order derivatives of the nonlinear dynamics. When the derivative functions do not span an invariant subspace, populating the observables $\Psi(s_k)$ with higher-order derivatives instead of arbitrary basis functions generates, locally in time, an increasingly (with the order of derivatives) accurate linear representation of the nonlinear dynamics.

Second, the derivative-based representation enables the derivation of error bounds on future predictions. Notably, when the model is entirely data-driven, these error bounds might not be enforced, but offer sound bound estimates that still hold, as I illustrate in Subsection 8.1.4, but which are then dependent on the quality of data used in the data-driven process. To approximate the Koopman operator, so far studies have largely focused on data-driven methods of the form in (7.4) that consider only the local error across one time step, that is

$$\Psi(s_{k+1},u_{k+1})-\tilde{\mathcal{K}}_d\Psi(s_k,u_k). \tag{8.1}$$

Another measure of the accuracy of the Koopman representation is the global error, over an arbitrarily long time window across all time steps $m>0$ (see Fig. 8.1), that is

$$\Psi(s_{k+m},u_{k+m})-\tilde{\mathcal{K}}_d^m\Psi(s_k,u_k). \tag{8.2}$$

The derivative-based linear embedding methodology presented in this work enables the computation of global error bounds of the Koopman representation. By exploiting the accuracy properties of Taylor expansions, I can synthesize Koopman basis functions that bound the model error for any particular order of linear representation. The error bounds in turn allow one to select the lowest-order representation that meets a desired accuracy. This analysis is presented next.

The proposed linear embedding method does not require knowledge of the dynamics; instead, it requires only that the time derivatives of the system states of interest be available. The values of the derivatives can be either evaluated, using knowledge of the dynamics equations, or numerically estimated from state measurements (using finite differencing or other methods [**196**]). The method can be used

- for known nonlinear dynamics: each state derivative is analytically derived from the dynamics equation and constitutes a basis function for the Koopman operator (one basis function per derivative)
- for dynamics whose structure is known but coefficients might be unknown or changing, as I illustrate in Subsection 8.1.3: derivatives are analytically derived from each term that appears in the dynamics equation; each term that is computed constitutes a separate basis function for the Koopman operator (at least one basis function per derivative)
- for completely unknown dynamics, as I illustrate in Subsection 8.1.4: each state derivative is numerically calculated and constitutes a basis function for the Koopman operator (one basis function per derivative).

### 8.1.1. Error Bounds of Derivative-Based Koopman Operators

The evolution of a nonlinear function $f(t)$ that is continuously differentiable up to $n$th order can be approximated with a Taylor series as

$$\begin{aligned}\tilde{f}(t_{k+1}) = & f(t_k) + f'(t_k) \cdot (t_{k+1} - t_k) \\ & + \frac{f''(t_k)}{2!} \cdot (t_{k+1} - t_k)^2 + \cdots + \frac{f^{(n)}(t_k)}{n!} (t_{k+1} - t_k)^n,\end{aligned}$$

where tilde $(\tilde{\cdot})$ denotes the predicted value of a function, and not its true value.[1] To keep the algebraic expressions compact, let $t_{k+1} - t_k = \Delta t$ and $f_k^{(i)}, \triangleq f^{(i)}(t_k)$, $\forall\ i \in \mathbb{Z} \cap [1, n]$,

[1]I assume that the true values of the function $f$ and its derivatives are known at time $t_k$. Uncertainty about the initial values can be readily included in the formula of the global error, later shown in this chapter.

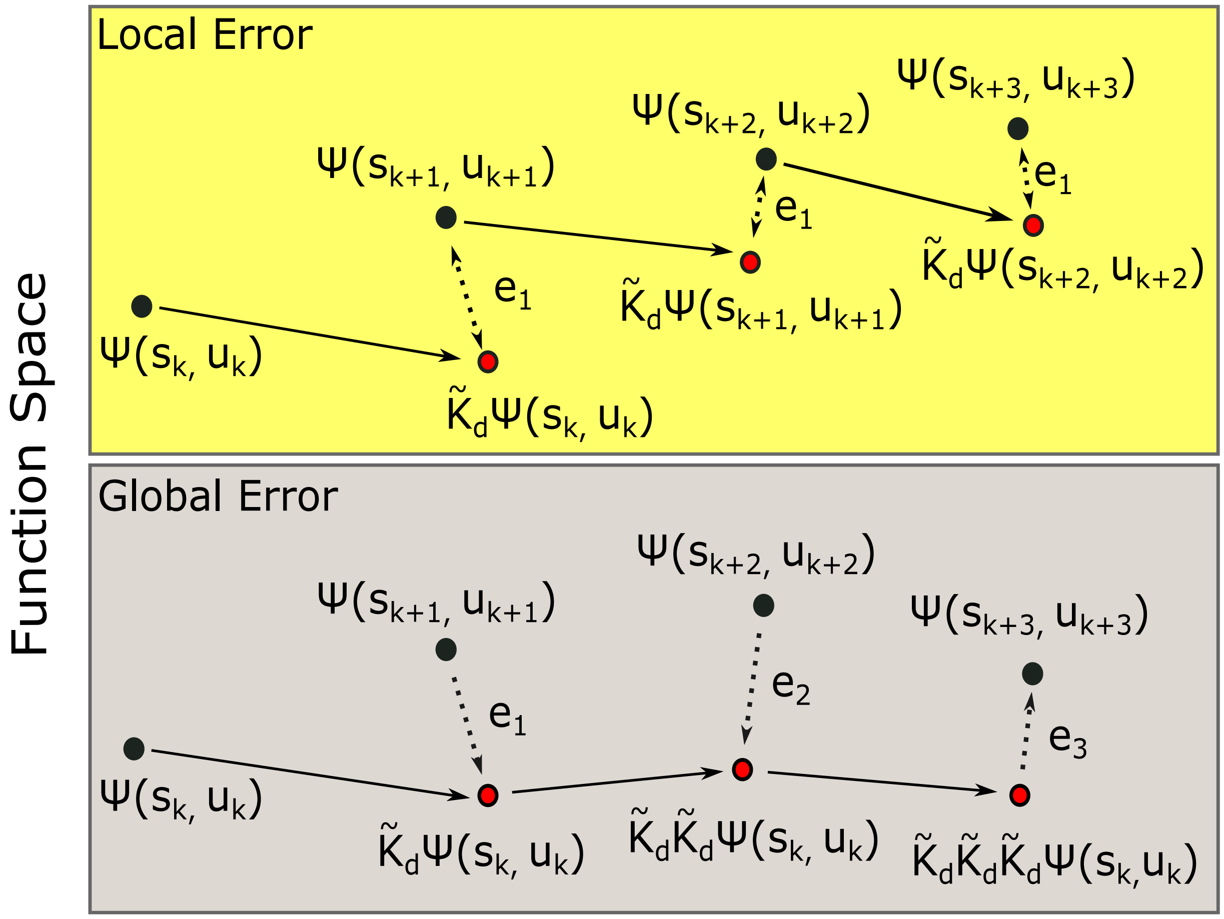


Figure 8.1. Local and global errors induced by approximate Koopman operators. The local error is the error induced by the operator across one step, assuming no error in the initial conditions. The global error is the total deviation away from the true states across multiple steps.

which simplifies the above expression to

$$\tilde{f}_{k+1} = f_k + f_k' \cdot \Delta t + f_k'' \cdot \frac{\Delta t^2}{2!} + \cdots + f_k^{(n)} \frac{\Delta t^n}{n!}. \tag{8.3}$$

Propagating a function using its derivatives allows one to use the accuracy of the Taylor series to characterize the error in the evolution of a function across one time step $\Delta t$. The local error induced by a Taylor series approximation using up to $n$ derivatives across one time step is $R_n(k)=f_{k+1}-\tilde{f}_{k+1}$. This error is calculated using Lagrange's remainder formula:

$$R_n(k)=\frac{f_c^{(n+1)}}{(n+1)!}\Delta t^{n+1}, \tag{8.4}$$

where $f_c^{(n+1)}\triangleq f^{(n+1)}(c)$ is the time derivative of order $n+1$ evaluated at some time $c\in[t_k,t_{k+1}]$. If there exists a positive real number $L$ such that $|f_c^{(n+1)}|\leq L$ for all $c\in[t_k,t_{k+1}]$, then the upper error bound in (8.4) becomes

$$|R_n(k)|\leq\frac{L}{(n+1)!}\Delta t^{n+1}. \tag{8.5}$$

For the purpose of applying linear control synthesis tools to linear representations of nonlinear dynamics, I bring the Taylor approximation in (8.3) to a linear matrix form:

$$\underbrace{\begin{pmatrix} \tilde{f}_{k+1} \\ \tilde{f}'_{k+1} \\ \tilde{f}''_{k+1} \\ \vdots \\ \tilde{f}^{(n)}_{k+1} \end{pmatrix}}_{\Psi(s_{k+1})} \approx \underbrace{\begin{pmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & \cdots & \frac{\Delta t^n}{n!} \\ 0 & 1 & \Delta t & \cdots & \frac{\Delta t^{n-1}}{(n-1)!} \\ 0 & 0 & 1 & \cdots & \frac{\Delta t^{n-2}}{(n-2)!} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}}_{\tilde{\mathcal{K}}_d} \underbrace{\begin{pmatrix} f_k \\ f'_k \\ f''_k \\ \vdots \\ f^{(n)}_k \end{pmatrix}}_{\Psi(s_k)}. \tag{8.6}$$

For a fixed $\Delta t$, expression (8.6) resembles (7.3), where the derivatives of the function $f_k$ are the observables $\Psi(s_k)$. When representing the Taylor series expansion, the derivative functions $f^{(i)}$ are known at time step $t_k$ and approximated at time $t_{k+1}$ by $\tilde{f}^{(i)}$. When training a Koopman operator, pairs of measurements of the states $s_k$ and $s_{k+1}$ are used to evaluate the basis functions at the corresponding time steps: $\Psi(s_k)$ and $\Psi(s_{k+1})$. Note that, in (8.6), all derivatives of $f_k$ are assumed to be different functions. The analytical expression (8.6) is equivalent to a Taylor series expansion (8.3) across one time step $\Delta t$ for all the basis functions. Therefore, the same error analysis (8.5) applies to each observable in (8.6).

When propagating a function across multiple time-steps using (8.6), the observable functions are themselves numerically propagated instead of being evaluated with measurable states at each time step, as is the case for a typical integration scheme. As a result, error accumulates not only in approximation of the original function $f$, but in the other observables as well. To track the error in the original $f$, therefore, it is necessary to be able to model the error in all of the basis functions. Using the accuracy of the Taylor series structure, I am able to model the error on every basis function and ultimately bound the model error in $f$.

**Theorem 3.** *Consider a general nonlinear function $f(t)$ that is continuously differentiable up to order $n$. Propagating $f(t)$ and its first $n$ derivatives using the Taylor-based linear representation* (8.6) *induces an error in $f(t)$ that is given by*

$$e_k = \sum_{i=1}^{k-1}\sum_{j=1}^{n} e_i^{(j)} \frac{\Delta t^j}{j!} + \sum_{i=0}^{k-1} f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1}}{(n+1)!}, \tag{8.7}$$

*where $n \in \mathbb{Z}^{\geq 0}$ is the number of derivative basis functions used, $k \in \mathbb{Z}^{\geq 1}$ is the number of time steps into the future, and, from Lagrange's remainder formula* (8.4), *$f_{i,i+1}^{(n+1)}$ is the $n+1$th time derivative of function $f$ evaluated at some time $t \in [t_i, t_{i+1}]$. The error bound* *is given by*

$$|e_k| \leq \frac{T^{n+1}}{(n+1)!} |f_{max}^{(n+1)}|, \tag{8.8}$$

*where* $T \triangleq k\Delta t$ *is the prediction time horizon and* $|f_{max}^{(n+1)}|$ *is the maximum magnitude of the* $n+1$*th derivative.*

Proof. For the derivation of the error expression (8.7), see Appendix A.4. For the derivation of the error bound formula (8.8), see Appendix A.5. □

To the best of my knowledge, this is the first work that provides prediction error bounds on the accuracy of a Koopman representation for general nonlinear dynamics. Prediction error bounds based on Taylor series had previously only been derived for a single step, and not for an arbitrary number of time steps into the future, as I derive in this chapter.

The error bound (8.8) is associated with the Koopman representation (8.6) for the dynamics of a single function $f$. The same methodology can be used to propagate multiple states of a system with coupled dynamics. Specifically, a system with states $s(t)$ and general nonlinear dynamics $\dot{s}(t) = g(s(t)) \in \mathbb{R}^N$ that are continuously differentiable up to order $n$

$$\frac{d}{dt} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} g_1(s) \\ g_2(s) \\ \vdots \\ g_N(s) \end{bmatrix} \tag{8.9}$$

can be propagated in discrete time as

$$\begin{bmatrix} s_{1,k+1} \\ s_{2,k+1} \\ \vdots \\ s_{N,k+1} \end{bmatrix} = \begin{bmatrix} s_{1,k}+g_{1,k}\cdot\Delta t+\cdots+g_{1,k}^{(n_1)}\frac{\Delta t^{n_1+1}}{(n_1+1)!} \\ s_{2,k}+g_{2,k}\cdot\Delta t+\cdots+g_{2,k}^{(n_2)}\frac{\Delta t^{n_2+1}}{(n_2+1)!} \\ \vdots \\ s_{N,k}+g_{N,k}\cdot\Delta t+\cdots+g_{N,k}^{(n_N)}\frac{\Delta t^{n_N+1}}{(n_N+1)!} \end{bmatrix}, \tag{8.10}$$

where $n_j$ for $j\in\mathbb{Z}\cap[1,N]$ indicates the highest-order of derivatives of $g_j$ used to propagate the $j$th state of the original dynamics (which does not have to be the same for all states). The above expression can be rewritten in a linear form similar to (8.6)

$$\underbrace{\begin{bmatrix} s_{1,k+1} \\ g_{1,k+1} \\ \vdots \\ g_{1,k+1}^{(n_1)} \\ \hdashline s_{2,k+1} \\ g_{2,k+1} \\ \vdots \\ g_{2,k+1}^{(n_2)} \\ \hdashline \vdots \\ \hdashline s_{N,k+1} \\ g_{N,k+1} \\ \vdots \\ g_{N,k+1}^{(n_N)} \end{bmatrix}}_{\Psi(s_{k+1})} = \underbrace{\begin{bmatrix} T(n_1) & \mathbf{0} & \cdots & \mathbf{0} \\ \hdashline \mathbf{0} & T(n_2) & \cdots & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} & \ddots & \vdots \\ \hdashline \mathbf{0} & \mathbf{0} & \cdots & T(n_N) \end{bmatrix}}_{\tilde{\mathcal{K}}_d} \underbrace{\begin{bmatrix} s_{1,k} \\ g_{1,k} \\ \vdots \\ g_{1,k}^{(n_1)} \\ \hdashline s_{2,k} \\ g_{2,k} \\ \vdots \\ g_{2,k}^{(n_2)} \\ \hdashline \vdots \\ \hdashline s_{N,k} \\ g_{N,k} \\ \vdots \\ g_{N,k}^{(n_N)} \end{bmatrix}}_{\Psi(s_k)}, \tag{8.11}$$

where

$$T(n_j)=\begin{bmatrix} 1 & \Delta t & \cdots & \dfrac{\Delta t^{n_j+1}}{(n_j+1)!} \\ 0 & 1 & \cdots & \dfrac{\Delta t^{n_j}}{n_j!} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \text{ for } j\in\mathbb{Z}\cap[1,N]. \tag{8.12}$$

Note how (8.11) is grouped into submatrices that propagate independently each state and its higher-order derivatives. The basis functions are the states and their derivatives. Propagating a nonlinear system with states $s$ and nonlinear dynamics $g(s(t))$ using (8.11) is equivalent to propagating each state separately using (8.6) and thus induces, for each state, an error given by an expression similar to (8.7).

The formulation in (8.11) uses basis functions that depend, for simplicity, only on the state $s$. When working with a system that has control inputs, one can treat controls $u$ in a similar fashion and calculate its higher-order derivatives, by introducing $u$ as dummy states that are the derivatives of the control input, a common practice [**86**].

**Corollary 3.1.** *Consider general nonlinear dynamics $\dot{s}(t)=g(s(t))$ that are continuously differentiable up to order $n$. Propagating $s(t)$ using* (8.11) *induces a bounded error on state $s_i$, $i\in\mathbb{Z}\cap[1,N]$, given by* (8.7)*, where $g(s(t))^{(i-1)}=f(t)^{(i)}$.*

Proof. Consider the propagation of each state $s_i$ and its derivative functions $g_i^{(\cdot)}$. Let each $s_i$ be a function $f_i$ whose $n_i^{th}$ derivative is $f^{(n_i)}$. Then, from Theorem 3, propagating each state $s_i \triangleq f_i$ with (8.6) induces an error given by (8.7). $\square$

The error bound (8.8) allows one to calculate the maximum possible error in each system state when propagating it with the fixed linear matrix (8.11). As a result, the bound can be used to determine the desired number of derivatives that are needed for each state that would generate minimal error given a fixed prediction horizon $T$ and subject to the nonlinear dynamics. Alternatively, the error bound can also be used to compute the maximum length of the prediction time horizon for which the state error is bound to remain under a threshold given a set number of derivative basis functions.

Because, in general, there is no closure of the higher-order derivatives and the series has to be truncated, the analytical expression (8.6) would only lead to an approximate Koopman operator, as is commented in [**85**]. Note, for example, that the highest derivatives in (8.11) are not updated at all. For this reason, I use data-driven techniques to obtain a $\tilde{\mathcal{K}}_d$ that more accurately advances all of the basis functions than the analytical expression. On the other hand, the error bound (8.8) applies to a linear propagation of nonlinear dynamics using (8.11) and therefore is no longer guaranteed when a data-driven operator is used instead. Nevertheless, it can still serve to measure how amenable nonlinear dynamics are to a linear representation by revealing the relationship between the magnitude and order of the derivatives. Furthermore, empirically, the data-driven model does resemble the Taylor-series structure (8.11) such that the error bounds remain relevant.[2] Using simulation results, I next verify the similarity of the data-driven operator to the analytical expression in (8.6), as well as the validity of the error bounds.

[2]Although the data-driven solution is not guaranteed to bound all local errors within the Taylor series accuracy, given a training dataset that is a representative part of the state space, solutions that largely deviate from the Taylor-series structure in (8.6) would generate large local errors in parts of the state space and thus be avoided by the least-squares solution (7.5).

### 8.1.2. Error Bound Estimation Using Data-Driven Operator

The error bound formula (8.8), derived for a linear representation of (8.11), remains relevant to a data-driven operator, when the latter has similar structure, i.e., small Frobenius distance, to the Taylor-series form (8.6). On the other hand, since an operator computed from data may not take exactly the form of (8.11), the bounds shown in (8.8) are not strictly enforced, but offer what I refer to as sound bound estimates in the remainder of the chapter. To calculate the bound estimates, one needs to compute $|f_{max}^{(n+1)}|$, the magnitude of the lowest-order derivative of the system states that is not used in the basis functions. When dynamics are known, this value can be calculated numerically. Alternatively, as I show next, one can exploit the Taylor-series structure of the data-driven operator to estimate the error bounds beyond the training set that has been used to generate the Koopman operator even when there is no knowledge of the dynamics.

Specifically, given a linear representation that approximates the Taylor-series structure (8.6), the local error across one time step induced by the data-driven model can be described by the Taylor series accuracy. Thus, using (8.8), the error across one time step ($k$=1) of a function $f$ can be written as

$$|e_1| \leq |f_{max}^{(n+1)} \frac{\Delta t^{(n+1)}}{(n+1)!}| = |f_{max}^{(n+1)}| \frac{\Delta t^{(n+1)}}{(n+1)!}, \tag{8.13}$$

where $e_1$ is available from the data-driven training process (7.4). Let $|e_1|_{max}$ be the maximum local error, i.e., $|e_1|_{max} \geq |e_1|$. Then, when the training data set is large enough, one can get

$$|e_1|_{max}\approx|f_{max}^{(n+1)}|\frac{\Delta t^{(n+1)}}{(n+1)!}, \tag{8.14}$$

which is rearranged to

$$|f_{max}^{(n+1)}|\approx|e_1|_{max}\frac{(n+1)!}{\Delta t^{(n+1)}}. \tag{8.15}$$

In short, I use the maximum error across one time step from the training process to estimate the term $|f_{max}^{(n+1)}|$, which in turn, using (8.8), allows us to estimate $|e_k|$, the error bound after $k$ time steps. Alternatively, when no analytical model of the dynamics is available, the value $|f_{max}^{(n+1)}|$ can also be estimated numerically using measurements of $f$.

### 8.1.3. Synthesis of Derivative-Based Koopman Observables with Structural Knowledge of Dynamics

The derivative-based approach proposed in this work populates Koopman observables with the system states $s$ and their derivative functions. Each derivative is a separate function that can be computed from the analytical expression when dynamics are fully known, or numerically estimated from measurements when no model exists. In this subsection, I show how I construct the basis functions to exploit structural knowledge of dynamics that have unknown coefficients.

For simplicity, I assume that the dynamics of each system state depend on a single term, i.e., a nonlinear function multiplied by a coefficient; the case of having multiple such terms can be handled similarly. In particular, consider a nonlinear system with states

$s \in \mathbb{R}^N$ and dynamics

$$\frac{d}{dt}\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} c_1 g_1(s) \\ c_2 g_2(s) \\ \vdots \\ c_N g_N(s) \end{bmatrix}, \tag{8.16}$$

where $c_i$, $i \in \mathbb{Z} \cap [1, N]$, are unknown coefficients and $g_i(s)$ are nonlinear functions of the states $s$. The second-order time derivatives of the states $s$ are

$$\frac{d^2}{dt^2}\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} c_1 g_1'(s) \\ c_2 g_2'(s) \\ \vdots \\ c_N g_N'(s) \end{bmatrix}, \tag{8.17}$$

where $g_i'(s)$ denotes the time derivative of $g_i$, and thus

$$c_i g_i'(s) = c_i\left(\frac{\partial g_i}{\partial s_1} c_1 g_1 + \cdots + \frac{\partial g_i}{\partial s_N} c_N g_N\right) \text{ for } i \in \mathbb{Z} \cap [1, N]. \tag{8.18}$$

For ease of discussion, I limit the analysis to the first two time derivatives, but the same process can continue to generate higher-order derivatives and, thus, additional basis functions.

Using the states $s_i$, the first-order derivatives $g_i$ and the individual terms $\frac{\partial g_i}{\partial s_j} g_j$ that appear in $g_i'(s)$, where $i, j \in \mathbb{Z} \cap [1, N]$, I populate the basis functions of the Koopman matrix.

In discrete time, the states are then propagated with

$$\begin{bmatrix} s_{1,k+1} \\ s_{2,k+1} \\ \vdots \\ s_{N,k+1} \end{bmatrix} = \begin{bmatrix} s_{1,k}+c_1 g_{1,k}\cdot\Delta t+c_1 g'_{1,k}\frac{\Delta t^2}{2!} \\ s_{2,k}+c_2 g_{2,k}\cdot\Delta t+c_2 g'_{2,k}\frac{\Delta t^2}{2!} \\ \vdots \\ s_{N,k}+c_N g_{N,k}\cdot\Delta t+c_N g'_{N,k}\frac{\Delta t^2}{2!} \end{bmatrix}. \tag{8.19}$$

Substituting for the $g'_i(s)$ terms, I show the expected form for a single state $s_1$:

$$\underbrace{\begin{bmatrix} s_{1,k+1} \\ g_{1,k+1} \\ \{\frac{\partial g_1}{\partial s_1} g_1\}_{k+1} \\ \{\frac{\partial g_1}{\partial s_2} g_2\}_{k+1} \\ \vdots \\ \{\frac{\partial g_1}{\partial s_N} g_N\}_{k+1} \end{bmatrix}}_{\Psi(s_{k+1})} = \underbrace{\begin{bmatrix} 1 & c_1\Delta t & c_1^2\frac{\Delta t^2}{2} & c_1 c_2\frac{\Delta t^2}{2} & \dots & c_1 c_N\frac{\Delta t^2}{2} \\ 0 & 1 & c_1\Delta t & c_2\Delta t & \dots & c_N\Delta t \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}}_{\tilde{\mathcal{K}}_d} \underbrace{\begin{bmatrix} s_{1,k} \\ g_{1,k} \\ \{\frac{\partial g_1}{\partial s_1} g_1\}_{k} \\ \{\frac{\partial g_1}{\partial s_2} g_2\}_{k} \\ \vdots \\ \{\frac{\partial g_1}{\partial s_N} g_N\}_{k} \end{bmatrix}}_{\Psi(s_k)}. \tag{8.20}$$

As before, to improve the accuracy of the linear representation, I use data to approximate a Koopman operator.

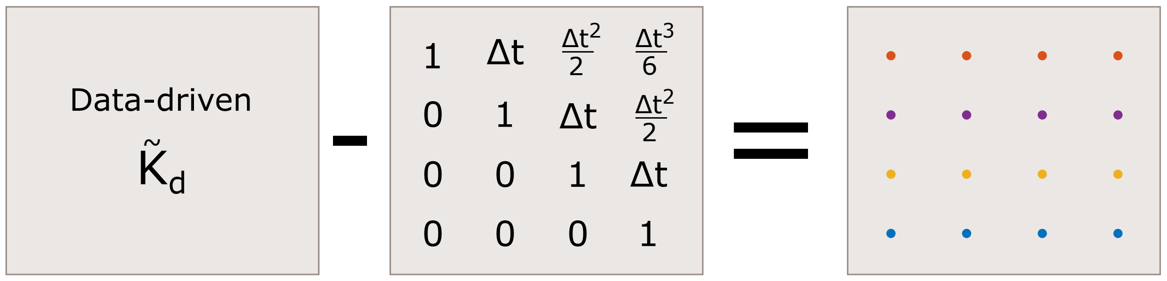
### 8.1.4. Assessment of Error Bound Estimates

Using the single pendulum system, I demonstrate the error bound estimates for the data-driven linear approximation, both when the dynamics are known and when they are unknown. In particular, I show that $|f_{max}^{(n+1)}|$ can be computed using the dynamics equations (model-based estimate) when those are available, or approximated using the residue error in the training process when the dynamics are unknown (data-driven estimate), as explained in Subsection 8.1.2. In both cases, once $|f_{max}^{(n+1)}|$ is calculated, the error bounds are computed using (8.8). The states are $s{=}[\theta{,}\omega]^T$ and dynamics are given by $\dot{s}{=}[\omega{,}\frac{g}{l}\sin(\theta){+}u]^T$, where $g{=}9.81\ {}^{\mathrm{m}}/_{\mathrm{s}^2}$ is the gravitational constant, $l{=}1$ m is the pendulum length, and $u$ is the control.

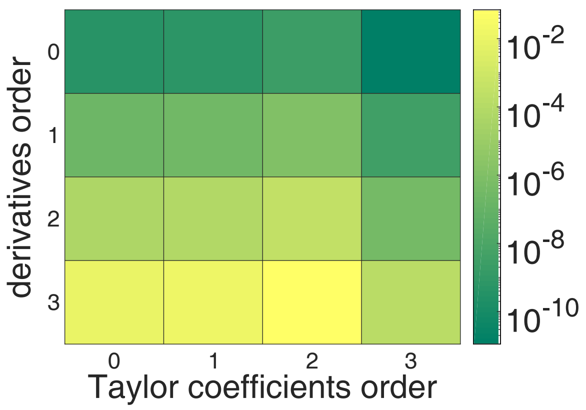
For the model-based estimate of the error bounds, the maximum magnitude of the $n$th derivative is (for each state) computed numerically by maximizing the symbolic expression over the domain of the state space that is used for training. For the data-driven estimate, the maximum magnitude is computed using (8.15) based on the training error. Note that, when propagating a data-driven operator instead of (8.11), the error bound estimates may in theory be violated.

I sample and forward-simulate 5000 initial states $s_0$ for $\Delta t{=}0.01$ s to obtain a Koopman operator $\tilde{\mathcal{K}}_d$ via (7.5), which I then use on a different randomly selected set of 5000 states to propagate the dynamics for a time horizon $T$. In both the training and the testing sets, uniform distributions of the initial states $\mathcal{U}_{\theta_0}(-2\pi\,\mathrm{rad},\ 2\pi\,\mathrm{rad})$ and $\mathcal{U}_{\omega_0}(-5\,\mathrm{rad/s},\ 5\,\mathrm{rad/s})$ are used. For each sample, both in training and in testing, I apply random inputs generated from a uniform distribution given by $\mathcal{U}_u(-5\mathrm{rad/s^2}{,}5\mathrm{rad/s^2})$. The observables include the angle $\theta$ and its first three derivatives, derived analytically based on the dynamics equation.

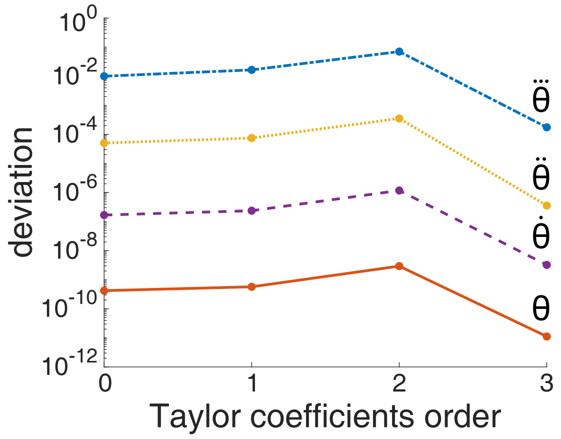
The obtained structure of the data-driven Koopman operator resembles the Taylor-series structure (8.6) (see Fig. 8.2), which adds validity to the data-driven error bound estimation. The error bound estimates and the actual errors are shown in Fig. 8.3. The error bound that is estimated from the structure of the data-driven operator without knowledge of the dynamics is reasonably accurate at predicting the maximum error. In


Data-driven
$\tilde{K}_d$
-
$\begin{matrix} 1 & \Delta t & \frac{\Delta t^2}{2} & \frac{\Delta t^3}{6} \\ 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{matrix}$
=

(a)



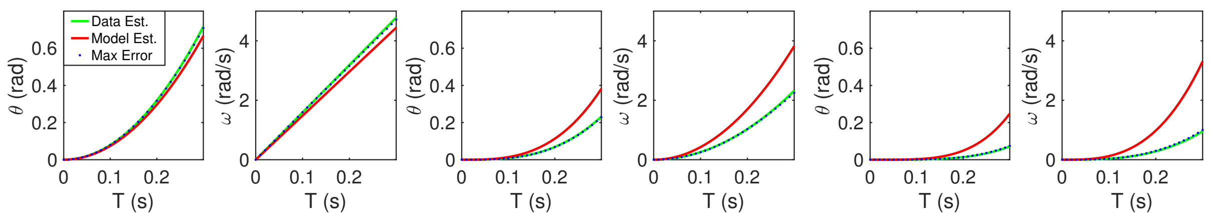
(b)



(c)

Figure 8.2. The deviation of the data-driven Koopman operator from the Taylor-based matrix (8.6) for the single pendulum system, where the derivative basis functions are constructed analytically from the known dynamics. Fig. 8.2b shows that the non-zero coefficients (upper triangle) of the linear Taylor expansion are accurately recovered from the data-driven operator. The zero coefficients (lower triangle) are replaced by small values that help minimize the least-squares error for the part of the state space used in the training set. The deviation differs by orders of magnitude across the basis functions, as seen in Fig. 8.2c. As expected, the deviation is smallest for $\theta$, as it is the one with the highest number of derivatives used in the basis functions.

addition, note that the maximum actual error and the error bound estimates have similar slopes with respect to the prediction horizon as well as the fact that both the actual error and the error bound estimates decrease with increasing order of derivatives used as basis functions.



(a) $n$=1 (b) $n$=2 (c) $n$=3

Figure 8.3. Simulated error bound estimates and actual error bounds for the single pendulum system as a function of the prediction horizon and for increasing orders of derivatives used as Koopman basis functions. The derivative basis functions are constructed analytically from the known dynamics. Both error bound estimates are calculated using (8.8), but differ in how they compute $|f_{max}^{(n+1)}|$. **Data Est.** is the model-free error bound estimate and uses the data-driven Koopman operator and (8.15) to compute $|f_{max}^{(n+1)}|$; **Model Est.** is the model-based error bound estimate and uses the analytical dynamics equations to compute $|f_{max}^{(n+1)}|$; **Max error** is the measured largest deviation as a function of time between the actual value of the state and the one predicted by the data-driven Koopman operator across all trajectories that evolve from randomly sampled initial conditions. Results are shown for three different orders of derivatives of $\theta$. Note that state $\theta$ has always one more derivative than $\omega$. The data-driven bound estimates and actual errors can be generated for different parameter choices using a Jupyter notebook at https://colab.research.google.com/drive/1EPX1XVUHr9gix-pZD_3Ydw7Npzz9n3Jj.

Next, I demonstrate the performance of the derivative-based data-driven Koopman operator and the error bound estimates when dynamics are unknown. Using the single pendulum system, only the angle and angular velocity are measured; higher-order derivatives are estimated using central finite differences [**197**]. To illustrate the robustness of the approach to noise, I add zero-mean Gaussian-distributed noise $\mathcal{N}(0,\sigma^2)$ to the measurements of $\theta$ and $\omega$, which are then also filtered through a moving average of 15 periods for noise reduction. The higher-order derivatives and the Koopman operator are computed from the filtered measurements. The term $|f_{max}^{(n+1)}|$ is computed as the maximum magnitude of $|f^{(n+1)}|$, which is also calculated using central finite differences [**197**].

In Fig. 8.4 I show results for $n$=2 and two levels of noise: low ($\sigma$=$\pi/180$) and high noise ($\sigma$=$15\pi/180$). Note that the actual maximum error induced by the Koopman operator remains almost identical, indicating that the operator is robust to high levels of noise. The error bound estimate for the low-noise scenario follows closely the error induced by the Koopman operator. In the high-noise scenario, the error bound estimate is more conservative. This is because the error bound formula is highly dependent on the calculated term $|f_{max}^{(n+1)}|$, which is likely to be miscalculated with noisy measurements. Note that if the training data do not represent the entire state space, the calculated value for $|f_{max}^{(n+1)}|$ is likely to underestimate the true value. On the other hand, including some safety margin in the $|f_{max}^{(n+1)}|$ term can make the error bounds more conservative. These results suggest that, although the error bound estimates may become less accurate with increasing levels of noise when dynamics are unknown, the performance of the derivative-based Koopman operator remains robust to noise, suggesting that the proposed methodology is a promising candidate for the prediction of unknown systems.

In conclusion, the error bounds are derived with respect to the analytical Taylor-based Koopman form of (8.6). However, I show (Fig. 8.2) that, in practice, the data-driven Koopman model has a very similar structure to (8.6) such that the error bounds remain relevant estimates. In addition, I verify that the error bounds reflect reasonably well the prediction error induced by data-driven Koopman models (Fig. 8.3), even in the presence of noise (Fig. 8.4). In practice, the error bounds, which depend on the prediction time horizon and the magnitude of the derivatives of the dynamics, provide a systematic method to determine the basis functions for a desired balance between model accuracy and complexity.

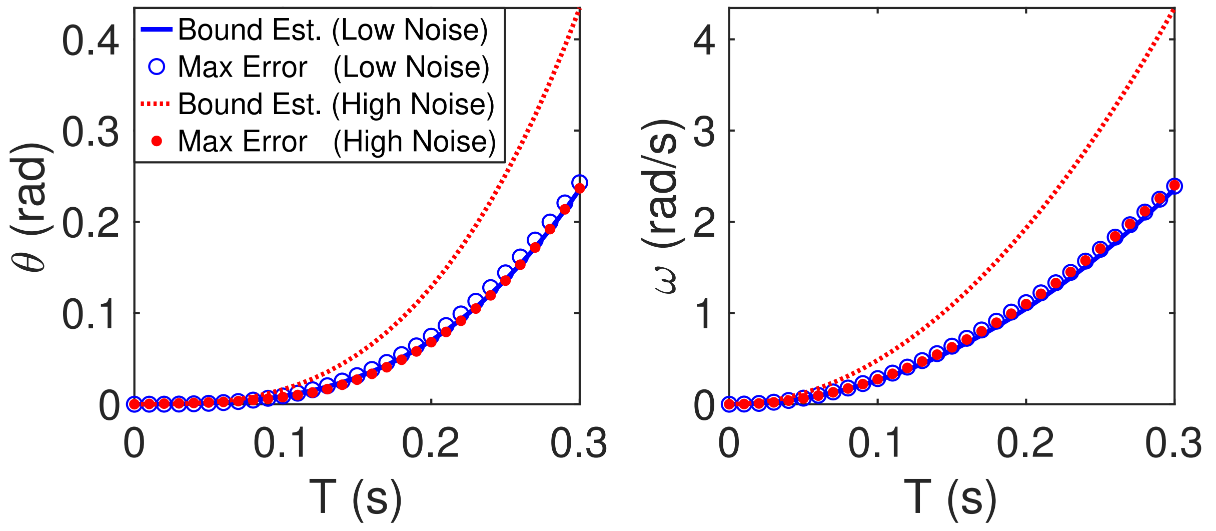


Figure 8.4. Simulated error bound estimates and actual maximum errors induced by the data-driven Koopman operators for the single pendulum system when dynamics are unknown and measurements are noisy. The derivative basis functions are calculated numerically from the state measurements—no analytical model is used.

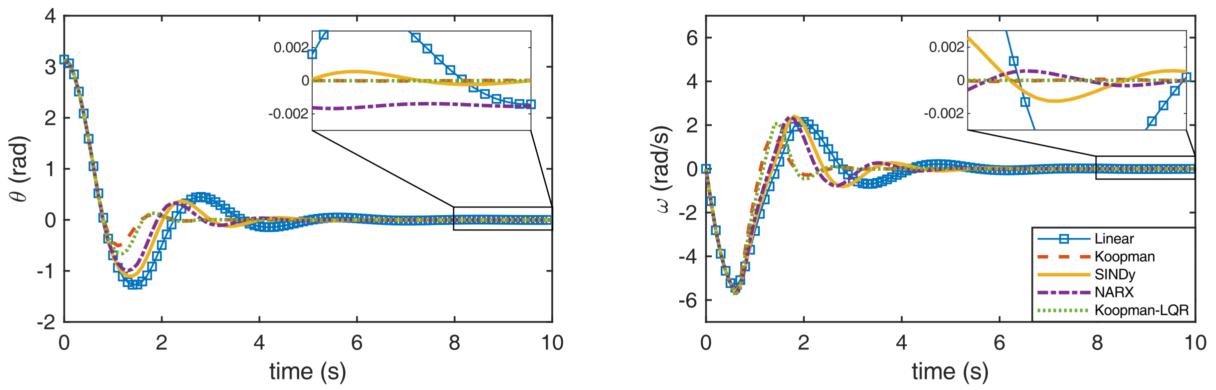


Figure 8.5. Control of a pendulum system based on data-driven models obtained using SINDy, NARX, a linear model based only on the system states, and a derivative-based Koopman model whose observables contain the state $\theta$ and its first- and second-order derivatives. The derivative basis functions are numerically estimated from state measurements both for training the Koopman model and online to implement control—no analytical model is used. All models are used to design MPC control and, in addition, I use the Koopman model for LQR feedback (Koopman-LQR). Koopman with MPC has the lowest cost (19.71) for the 10 s simulation, while NARX, SINDy and the linear model result in errors that are 12.43%, 18.32%, and 30.61% higher, respectively. Koopman-LQR leads to the second best performance (0.97% higher error in comparison to Koopman-MPC).

Note that, in this work, I do not argue that one should always use additional basis functions than the original system states; instead, one could analyze when and how to augment the basis functions of Koopman operators to improve the modeling accuracy while being cognizant of increased system order and complexity. In particular, the derived error bound estimates can serve as a guide on whether one should do so and by how many derivatives.

### 8.1.5. Comparison to alternative system identification methods

Finally, I compare the performance of the derivative-based data-driven Koopman representation approach to alternative system identification methods using the single pendulum system. Specifically, I use SINDy [**198**], NARX [**199**], a data-driven linear model, and the proposed derivative-based data-driven Koopman approach to obtain models and design MPC control to invert the pendulum to the upright position.

To train the models I uniformly sample and forward simulate 500 initial states $s_0$ for $\Delta t$=0.04 s. The initial states are sampled from uniform distributions given by $\mathcal{U}_{\theta_0}(-2\pi\,\text{rad}, 2\pi\,\text{rad})$ and $\mathcal{U}_{\omega_0}(-5\,\text{rad/s}, 5\,\text{rad/s})$. I use random inputs generated from a distribution given by $\mathcal{U}_u(-10\text{rad/s}^2, 10\text{rad/s}^2)$. I train the SINDy model using the open-source package [**200**] and the NARX model using the MATLAB Deep Learning Toolbox [**201**]. The higher-order derivatives used for SINDy and the derivative-based Koopman model are numerically estimated from the angle and angular velocity measurements.

Fig. 8.5 shows simulation results for the inversion of the single pendulum system using MPC control computed from models obtained with NARX, SINDy, a data-driven linear model based only on the system states ($n$=1, see Fig. 8.3), and a data-driven Koopman representation based on the state $\theta$ and its first- and second-order derivative ($n$=2). In addition, I also test LQR control on the Koopman model to demonstrate that similar control performance can be achieved with LQR gains that are computed once. The desired states are given by $s_{des}$=[0,0], the weights for the states and control are $Q$=diag(5,0.01) and $R$=0.001, respectively, and the prediction horizon is $T$=0.1 s.

I also compare the control performance of these methods for 30 initial conditions $\theta_0$ and $\omega_0$ sampled from uniform distributions given by $\mathcal{U}_{\theta_0}(-\pi\,\text{rad}, \pi\,\text{rad})$ and $\mathcal{U}_{\omega_0}(-2\,\text{rad/s}, 2\,\text{rad/s})$. The average of the 30 errors is 6.00 (with a standard deviation of 6.60) for the Koopman-LQR approach; 6.05 (with a standard deviation of 6.59) for the Koopman-MPC; 6.48 (with a standard deviation of 7.00) for NARX; 6.69 (with a standard deviation of 7.18) for SINDy; and 7.44 (with a standard deviation of 7.70) for the linear model.

These results show that the control performance of the Koopman-MPC method is marginally better than the linear-MPC, NARX-MPC and SINDy-MPC. Also, the Koopman-LQR approach delivers control performance comparable to the Koopman-MPC method, with the additional benefit that it lends itself to efficient control computation, which makes it an attractive choice for online robotic control applications. This is why I use Koopman-LQR in the simulations and experiments with the robotic fish in Section 8.2.

## 8.2. Data-Driven Control of Tail-Actuated Robotic Fish

I illustrate and validate the proposed data-driven modeling approach using a tail-actuated robotic fish. The states of the robotic fish are $s$=$[x,y,\psi,v_x,v_y,\omega]^T$, where $x$,$y$ are the 2D world-frame coordinates, $\psi$ is the orientation, $v_x$ and $v_y$ are the body-frame linear velocities (surge and sway, respectively), and $\omega$ is the body-frame angular velocity. I use $\alpha$

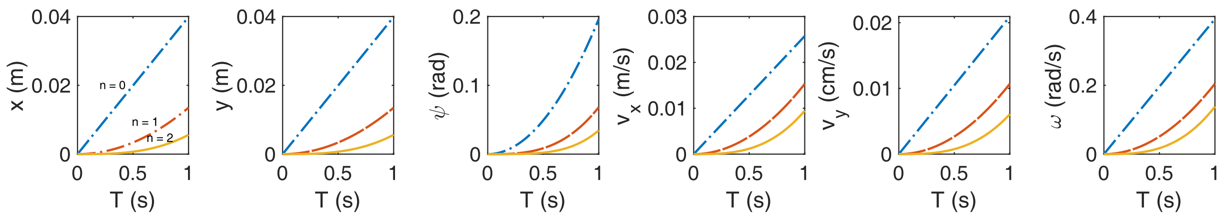


Figure 8.6. Error bound estimates based on derivative-based Koopman models of the robotic fish dynamics (8.21) for increasing order of derivatives used in the basis functions. The derivative basis functions are constructed analytically from the known dynamics. Each additional order of derivatives improves the error bound estimates over the selected prediction horizon. The error bound estimates are computed using (8.8) where $|f_{max}^{(n+1)}|$ is calculated from the training data.

to indicate the angle of the tail, actuated with $\alpha(t){=}\alpha_o{+}\alpha_a\sin(\omega_a t)$, where $\alpha_a$,$\alpha_o$,$\omega_a$ are the amplitude, bias, and frequency of the tail beat. To simplify the problem, I keep the frequency fixed at $\omega_a{=}2\pi$ rad/s.

I describe the dynamics of the system with an average model [**202**] given by

$$\dot{s}=\begin{bmatrix}\dot{x}\\\dot{z}\\\dot{\psi}\\\dot{v}_x\\\dot{v}_y\\\dot{\omega}\end{bmatrix}\triangleq\begin{bmatrix}v_x\cos(\psi)-v_y\sin(\psi)\\v_x\sin(\psi)+v_y\cos(\psi)\\\omega\\f_1(s)+K_f f_4(\alpha_0,\alpha_a,\omega_a)\\f_2(s)+K_f f_5(\alpha_0,\alpha_a,\omega_a)\\f_3(s)+K_m f_6(\alpha_0,\alpha_a,\omega_a)\end{bmatrix},\tag{8.21}$$

where

$$
\begin{aligned}
f_1(s) &= \frac{m_2}{m_1}v_y\omega - \frac{c_1}{m_1}v_x\sqrt{v_x^2+v_y^2} + \frac{c_2}{m_1}v_y\sqrt{v_x^2+v_y^2}\arctan(\frac{v_y}{v_x}) \\
f_2(s) &= -\frac{m_1}{m_2}v_x\omega - \frac{c_1}{m_2}v_y\sqrt{v_x^2+v_y^2} - \frac{c_2}{m_2}v_x\sqrt{v_x^2+v_y^2}\arctan(\frac{v_y}{v_x}) \\
f_3(s) &= (m_1-m_2)v_xv_y - c_4\mathrm{sgn}(\omega)\omega^2 \\
f_4(\alpha_0,\alpha_a,\omega_a) &= \frac{m}{12m_1}L^2\omega_a^2\alpha_a^2(3-\frac{3}{2}\alpha_o^2-\frac{3}{8}\alpha_a^2) \\
f_5(\alpha_0,\alpha_a,\omega_a) &= \frac{m}{4m_2}L^2\omega_a^2\alpha_a^2\alpha_o \\
f_6(\alpha_0,\alpha_a,\omega_a) &= -\frac{m}{4J_3}L^2c\omega_a^2\alpha_a^2\alpha_o
\end{aligned}
\tag{8.22}
$$

and $m_1 = m_b - m_{a_x}, m_2 = m_b - m_{a_y}, J_3 = J_{b_z} - J_{a_z}, c_1 = \frac{1}{2}\rho SC_D, c_2 = \frac{1}{2}\rho SC_L, c_4 = \frac{1}{J_3}K_D, c_5 = \frac{1}{2J_3}L^2mc$. Parameter $m_b$ is the mass of the robotic fish, $m_{a_x}$ and $m_{a_y}$ are the hydrodynamic derivatives that represent the added masses of the robotic fish along the $x$ and $y$ directions, respectively, $J_{a_z}$ and $J_{bz}$ are the added inertia effect and the inertia of the body about the $z$-axis, respectively, $m$ is the mass of the water displaced by the tail per unit length, $\rho$ is the water density, $L$ is the tail length, $c$ is the distance from the body center to the pivot point of the actuated tail, $C_D, C_L, K_D$ are drag force, lift, and drag moment coefficients, respectively, and $K_f$ and $K_m$ are scaling coefficients measured experimentally [**203**].

I train a Koopman operator using the control-affine form of the dynamics (8.21) that is obtained by substituting

$$
u_1 = \alpha_a^2(3-\frac{3}{2}\alpha_o^2-\frac{3}{8}\alpha_a^2) \qquad u_2 = \alpha_a^2\alpha_o. \tag{8.23}
$$

Because the computed control is in terms of the variables $u_1$ and $u_2$, it needs to be mapped to implementable values for the amplitude, $\alpha_a$, and the bias, $\alpha_o$, of the tail flapping. To convert $u_1$ and $u_2$ back to the physical actuation variables, I use a constrained global minimization solver based on Sequential Quadratic Programming (SQP) that finds the nearest, in the space of $u_1$ and $u_2$, feasible actuation values for $\alpha_a$ and $\alpha_o$. Given $u_1$ and $u_2$, the constrained optimization problem is posed as

$$\begin{gathered}\operatorname*{argmin}_{\alpha_a,\alpha_o}\sqrt{\Big(u_1-\alpha_a^2(3-\frac{3}{2}\alpha_o^2-\frac{3}{8}\alpha_a^2)\Big)^2+\Big(u_2-\alpha_a^2\alpha_o\Big)^2} \\ \text{subject to: } \alpha_a\in[0,30°] \text{ AND } \alpha_o\in[-45°,45°].\end{gathered} \tag{8.24}$$

This minimization is solved before every control update.

Given the unilateral constraints on the forward motion of the tail-actuated robotic fish (it cannot move backwards), directly tracking position coordinates becomes rather challenging. For example, when the target lies behind the robotic fish, the control solution generates a negative amplitude (to generate backward motion) that is infeasible and thus the system stops moving. This behavior has been observed and tackled in [**204**] by translating position coordinates into different error states, associated with the body-frame velocities of the system. Similarly, in this work, I argue that one can express all feasible trajectories for the tail-actuated robotic fish in terms of an angle and a forward velocity profile.

### 8.2.1. Simulation Results

In this subsection, I present simulation results on the data-driven modeling and LQR control on the tail-actuated robotic fish. To decide the optimal order of derivative basis functions, I compare the error bound estimates over $T$=1 s, which is the feedback rate, using different orders of derivatives. Results are shown in Fig. 8.6. I select $n$=2 for a reasonable balance between the increasing complexity of calculating higher-order derivatives and model accuracy. Next, I populate the observable functions with the states, their first- (using (8.21)) and second-order derivatives, which are derived analytically from the average model. To allow the identification of unknown or changing coefficients (as discussed in Subsection 8.1.3), I consider each term in the derivatives individually. For example, $\frac{d}{dt}v_y\omega=\dot{v}_y\omega+v_y\dot{\omega}$ generates multiple basis functions, where $\dot{v}_y$ and $\dot{\omega}$ are given by (8.21). Using separate functions for the time derivatives of each individual term that appears in the dynamics is similar to using the time derivatives of the entire equation of a state (e.g. $\ddot{v}_y(t)$). Despite increasing the number of basis functions, I prefer the first approach because it does not require knowing the coefficients of the individual terms in advance (e.g. $\frac{m_2}{m_1}$). As a result, I can readily train the Koopman operator on other robotic tail-actuated fish that have a different morphology. In this way, I end up with the system states, the control inputs, and 54 additional scalar functions, with $\Psi_s(s)\in\mathbb{R}^{60}$ and $\Psi_u(u)=u\in\mathbb{R}^2$.

Note that I choose control-dependent basis functions to be $u$ so as to use LQR feedback. One can also choose basis functions that include nonlinear control terms in combination with a different control policy, such as NMPC [**86**, **205**]. Alternatively, one can always convert dynamics that are nonlinear in control by introducing new dummy variables (e.g. $v_i$) as the control input that are the derivatives of the system actuation ($\dot{u}_i=c_i(v_i-u_i)$), where $c_i\in\mathbb{R}^+$ dictate the rate of change [**86**]. This is in practice also closer to the physical implementation of actuation that cannot instantly change values. In this way, it is then

Table 8.1. Simulation parameters for the tail-actuated fish model dynamics (8.21).

| Simulation Parameters | | | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| $m_b$ | $0.725\,\mathrm{kg}$ | $\rho$ | $1000\,\mathrm{kg/m^3}$ |
| $m_{ax}$ | $-0.217\,\mathrm{kg}$ | $S$ | $0.03\,\mathrm{m^2}$ |
| $m_{ay}$ | $-0.7888\,\mathrm{kg}$ | $C_D$ | 0.97 |
| $J_{bz}$ | $2.66{\times}10^{-3}\,\mathrm{kg{\cdot}m^2}$ | $C_L$ | 3.9047 |
| $J_{az}$ | $-7.93{\times}10^{-4}\,\mathrm{kg{\cdot}m^2}$ | $K_D$ | $4.5{\times}10^{-3}$ |
| $L$ | $0.071\,\mathrm{m}$ | $K_f$ | 0.7 |
| $d$ | $0.04\,\mathrm{m}$ | $K_m$ | 0.45 |
| $c$ | $0.105\,\mathrm{m}$ | | |

possible to include nonlinearities in $u_i$, while the system remains still linear with respect to the control input $v_i$ designed by the user.

Next, I train an approximate Koopman operator using (7.5). To generate data $s_k$ and $s_{k+1}$, I sample $P$=3000 initial conditions for the states with uniform distributions given by $\mathcal{U}_{x_0}(-0.5\,\mathrm{m},\ 0.5\,\mathrm{m})$, $\mathcal{U}_{y_0}(-0.1\,\mathrm{m},\ 0.1\,\mathrm{m})$, $\mathcal{U}_{\psi_0}(-\pi/4\,\mathrm{rad},\ \pi/4\,\mathrm{rad})$, $\mathcal{U}_{v_x}(0,\ 0.04\,\mathrm{m/s})$, $\mathcal{U}_{v_y}(-0.0025\,\mathrm{m/s},\ 0.0025\,\mathrm{m/s})$, $\mathcal{U}_{\omega}(-0.5\,\mathrm{rad/s},\ 0.5\,\mathrm{rad/s})$. For each sample, I apply random inputs generated from a uniform distribution given by $\mathcal{U}_{\alpha_0}(-45°,\ 45°)$ for the tail angle bias and $\mathcal{U}_{\alpha_a}(0{,}30°)$ for the tail angle amplitude of oscillations. Then, for each sample of initial conditions $s_k$ and controls $u_k$, I use dynamics (8.21) and parameters shown in Table 8.1 to propagate the states with the given control for $\Delta t$=0.005s and obtain the final states $s_{k+1}$. I use the set of $s_k$,$s_{k+1}$,$u_k$ to compute the approximate discrete Koopman operator (7.5). Note that the value of $u_{k+1}$ can be arbitrary, since I am not trying to predict the evolution of the control-dependent basis functions. Once I have trained the Koopman operator, I convert it to the continuous time via $\tilde{\mathcal{K}}$=log$(\tilde{\mathcal{K}}_d)/\Delta t$, extract the state- and control-linearization matrices $A$ and $B$, choose the weight matrices $Q$ and $R$ and compute the infinite-horizon LQR gains.

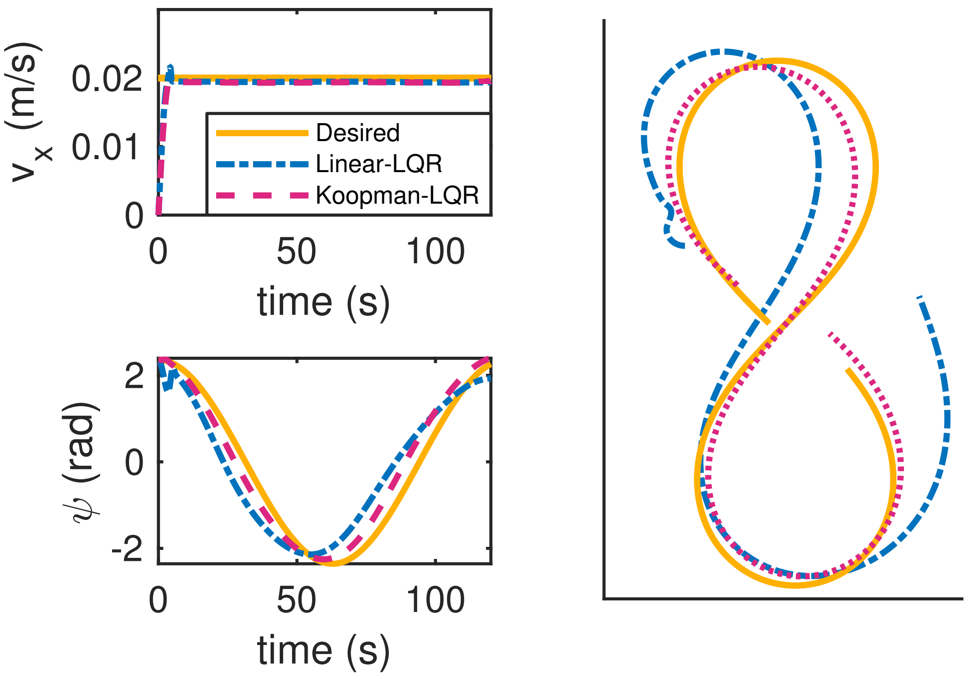


Figure 8.7. LQR-controlled robotic fish in simulation. The LQR gains are generated once using the learned Koopman operator. The derivative basis functions are constructed analytically from the known dynamics. The desired trajectory is given in terms of the angle and the forward velocity. Since the position coordinates are not included in the performance objective (7.10), the controlled trajectories are individually shifted to align with the desired figure-8 shape as closely as possible. Despite using fixed LQR gains, the controlled systems successfully track the desired states that were designed to produce a figure-8 pattern. Koopman-LQR has the lower cost (3.35) for the 120 s duration, while the approach based on the data-driven linear model with the same set of states as in (8.21) results in an error that is 95% higher.

Figure 8.7 shows the velocity tracking performance of the derivative-based Koopman model in comparison to a linear data-driven model for the system (8.21) (with the same set of states as in (8.21)) when using LQR-feedback. The desired trajectory is a figure-8 described by $s_{des}$=[0,0,135$\cdot\pi/$180$\cdot$sin(0.05$t$+$\pi/$2),0.02,0,0.05$\cdot$135$\cdot\pi/$180$\cdot$cos(0.05$t$+$\pi/$2)]. The weights are $Q$=diag(0,0,0.1,4000,0,0) and $R$=diag(0.01,0.01). As is seen in the figure

Table 8.2. Amplitude and bias inputs used to collect training dataset.

| | Actuation values | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Amp (°) | 15 | | | | | 20 | | | | | 25 | | | | | 30 | | | | |
| Bias (°) | 0 | ±20 | ±30 | ±40 | ±50 | 0 | ±20 | ±30 | ±40 | ±50 | 0 | ±20 | ±30 | ±40 | ±50 | 0 | ±20 | ±30 | ±40 | ±50 |

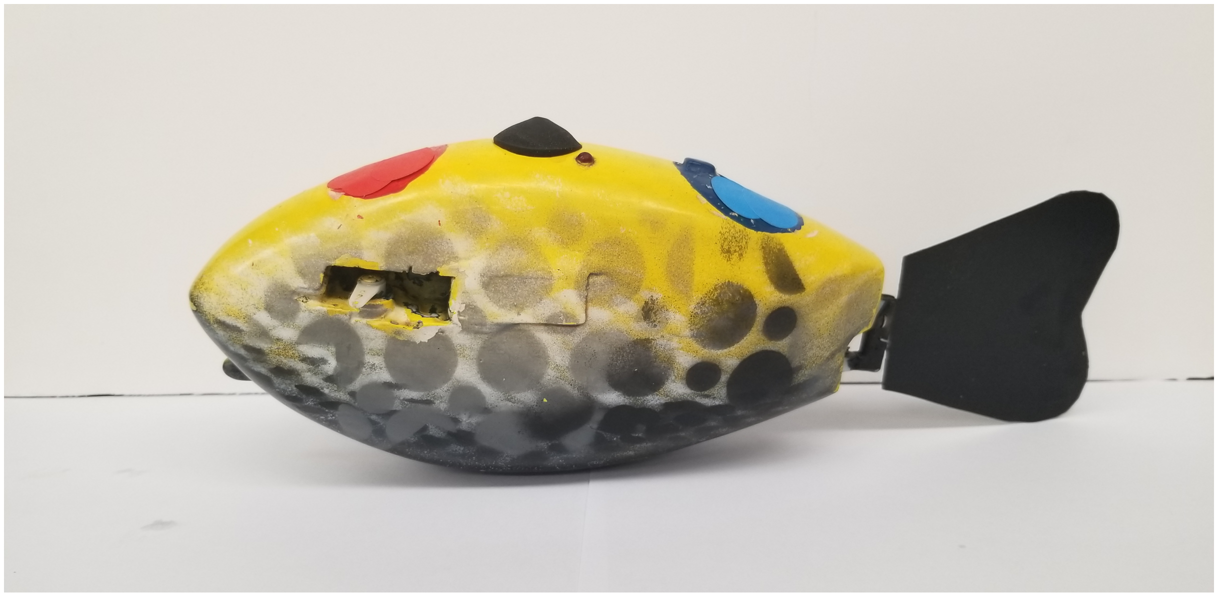
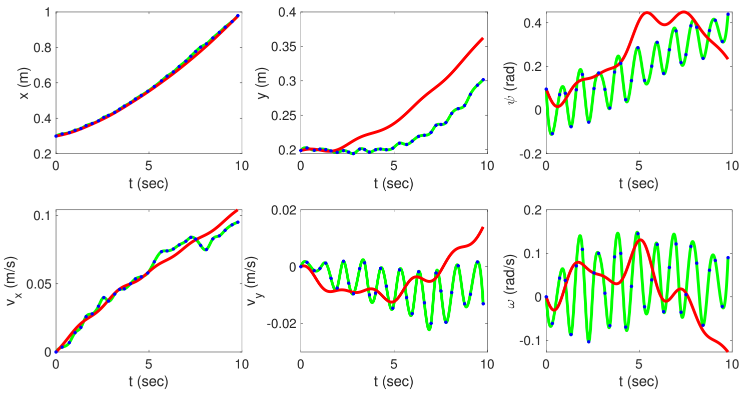


Figure 8.8. Tail-actuated robotic fish used in experiments, developed by the Smart Microsystems Lab at Michigan State University. It maneuvers in water by oscillating its tail fin.

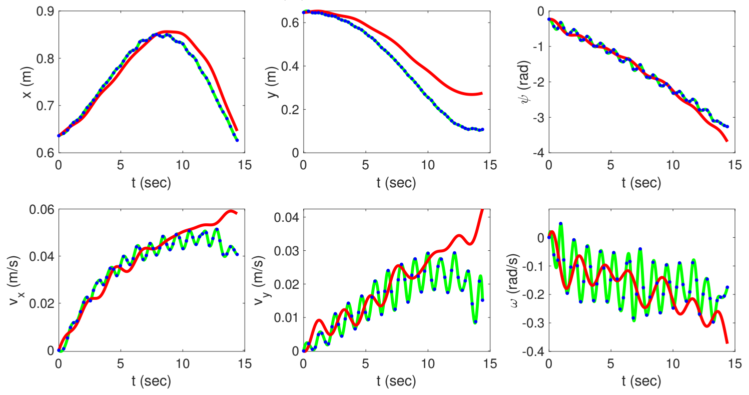
the proposed derivative-based Koopman modeling approach leads to improved control performance.

### 8.2.2. Experimental Results

**8.2.2.1. Experimental Setup.** I next use the physical robot shown in Fig. 8.8 to experimentally test the proposed approach. An overhead camera captures the red and blue marks on the robotic fish (see Fig. 8.8) and calculates the coordinates of its center and its orientation at about 3Hz. The body-frame velocities are estimated using a Kalman filter. The state derivative functions, provided by the average model, are then evaluated with the



(a) $\alpha_0$=0,$\alpha_a$=30°



(b) $\alpha_0$=45°,$\alpha_a$=25°

Figure 8.9. Fitness between Koopman model and experimental measurements. The green line shows data interpolated from experimental measurements (blue dots) every $\Delta t$=0.005s. The red line shows the evolution of the states using the Koopman model. The actuation is constant for each of the two runs and is indicated in the caption.

states. To simplify the modeling and control task, the tail-beat frequency used in both the training phase and the testing phase is kept constant at $\omega_a$=2$\pi$rad/s. In order not to disturb the periodic movement of the tail oscillation during tracking, feedback control is updated at roughly one second. The control commands are communicated to the robot via Xbee (RF communication).

**8.2.2.2. Training Phase.** For the training phase, I collect experimental measurements using the robotic fish to compute the approximate Koopman operator. Throughout each run, I apply constant tail bias and amplitude for the oscillations of the tail fin. I conduct a total of 72 runs, with two trials for each of the 36 different combinations of actuation parameters, shown in Table 8.2. I train the Koopman operator using the same basis functions as discussed in Subsection 8.2.1.

To create a consistent mapping with the Koopman operator, all pairs of measurements $s_k$ and $s_{k+1}$ need to be spaced (in time) equally apart. For this reason, and also to decrease the time between measurements to fine levels (without the constraints of my sampling and filtering methods), the obtained data is interpolated at $\Delta t$=0.005s. The interpolated data is then used to obtain an approximate Koopman operator according to (7.4).

To measure how well the Koopman model captures the nonlinear dynamics of the robotic fish, I use $\tilde{\mathcal{K}}$, learned from the experimental data, to propagate the identified model continuously based on the initial states of each of the 72 experimental runs. Then, the predicted simulated trajectories are compared against the corresponding experimental ones. For the purposes of illustration, two such comparisons are shown in Fig. 8.9. The linear Koopman model, despite not perfect, reasonably follows the experimental data for at least five seconds. Note that, because I only minimize the single-step prediction error (7.5), it is more likely (compared to minimizing a multi-step error) that I compute an unstable Koopman operator, even if the dynamics are stable. In that case, the long-term predictions would exponentially deviate and become inaccurate. Imposing stability properties on the operator is the focus of ongoing work [**206**, **207**].

**8.2.2.3. Testing Phase.** I use the data-trained Koopman operator to implement linear feedback control (LQR) for tracking. Using the weight matrices $Q$ and $R$, which penalize the tracking error and control effort, respectively, I define the minimization problem (7.10) and calculate the infinite-horizon LQR gains. Contrary to work in [**92**], and in order to illustrate the simplicity and robustness of the proposed scheme, I keep the same weights across all different tasks, such that the same LQR gains, (unless the model is updated) are used in every type of trajectory. The resulting feedback has the form shown in (7.12). As mentioned earlier, I argue that, to follow any trajectory, it suffices to track a desired orientation and forward velocity and so I design the LQR weights accordingly. Specifically, the weights used are $Q$=diag(0,0,0.1,1000,0,0) and $R$=diag(1,1). The weights for the angle and forward velocity are disproportionate to account for the difference in scale between the velocity of the robotic fish, typically in the order of 0.01 m/s, and the body orientation, expressed in radians.

I implement the proposed data-driven Koopman methodology in two ways. One approach is computing the model and LQR gains offline once; the other is updating the model and recalculating the LQR gains online in real time. To update the Koopman operator online in a memory-efficient manner, I do not store any previous measurements. Instead, I use (7.5) and split it into the $P$ measurements used last to calculate the Koopman operator and the $\Delta P$ new measurements since the last update, where $P_{total}=P+\Delta P$ is the total number of measurements used. Then,

$$\tilde{\mathcal{K}}^{*}_{d,new}=\mathcal{A}_{new}\mathcal{G}^{\dagger}_{new}, \tag{8.25}$$

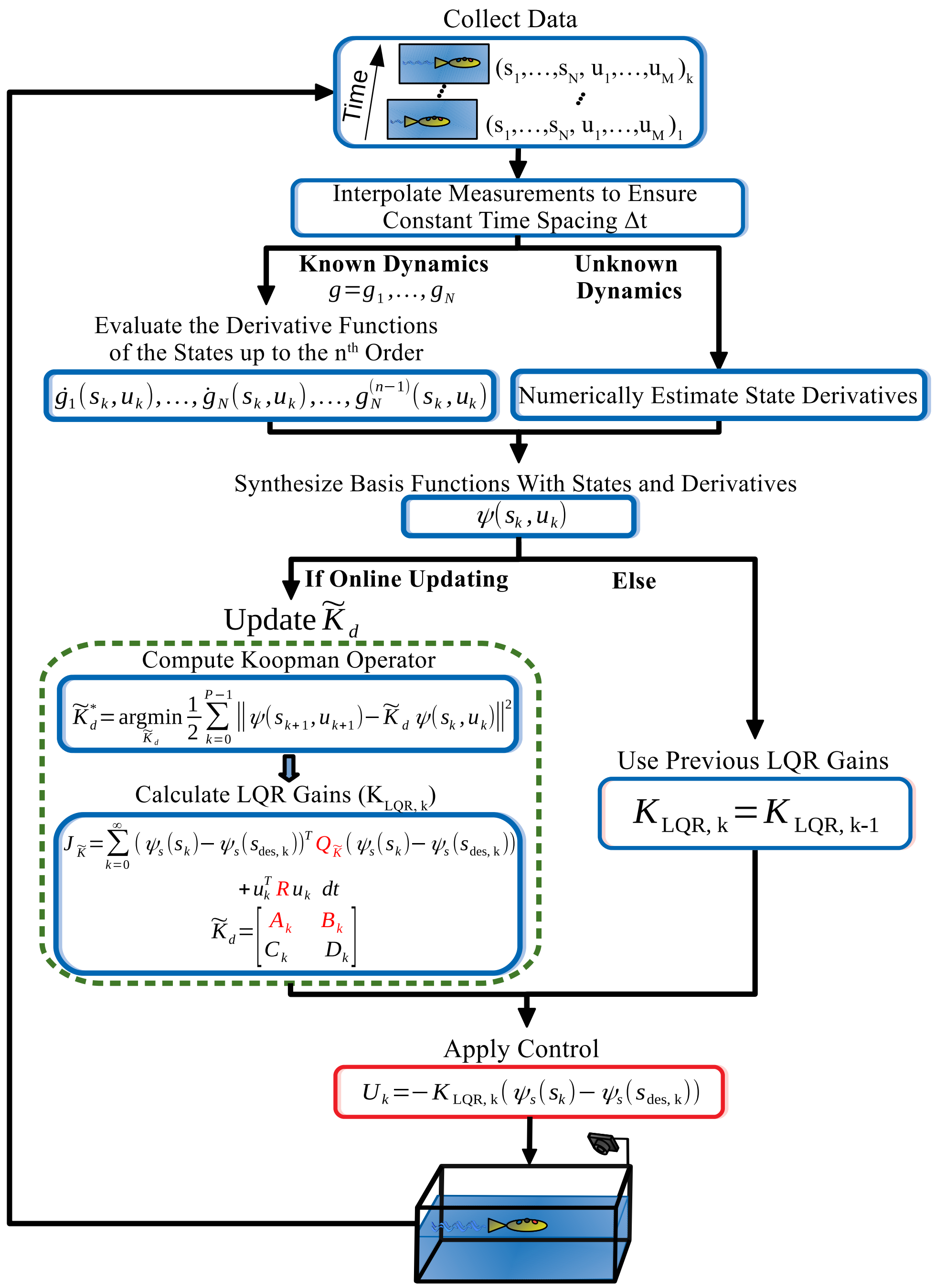

Collect Data
Time
$(s_1,\dots,s_N, u_1,\dots,u_M)_k$
$(s_1,\dots,s_N, u_1,\dots,u_M)_1$
Interpolate Measurements to Ensure Constant Time Spacing Δt
Known Dynamics $g=g_1,\dots,g_N$
Unknown Dynamics
Evaluate the Derivative Functions of the States up to the nth Order
$\dot{g}_1(s_k,u_k),\dots,\dot{g}_N(s_k,u_k),\dots,g_N^{(n-1)}(s_k,u_k)$
Numerically Estimate State Derivatives
Synthesize Basis Functions With States and Derivatives
$\psi(s_k,u_k)$
If Online Updating
Else
Update $\widetilde{K}_d$
Compute Koopman Operator
$\widetilde{K}_d^* = \underset{\widetilde{K}_d}{\operatorname{argmin}} \frac{1}{2}\sum_{k=0}^{P-1} \|\psi(s_{k+1},u_{k+1}) - \widetilde{K}_d\,\psi(s_k,u_k)\|^2$
Calculate LQR Gains ($K_{\text{LQR, k}}$)
$J_{\widetilde{K}} = \sum_{k=0}^{\infty} (\psi_s(s_k) - \psi_s(s_{\text{des, k}}))^T Q_{\widetilde{K}} (\psi_s(s_k) - \psi_s(s_{\text{des, k}})) + u_k^T R u_k \; dt$
$\widetilde{K}_d = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix}$
Use Previous LQR Gains
$K_{\text{LQR, k}} = K_{\text{LQR, k-1}}$
Apply Control
$U_k = -K_{\text{LQR, k}}(\psi_s(s_k) - \psi_s(s_{\text{des, k}}))$

Figure 8.10. Outline of the proposed methodology for LQR control using derivative-based Koopman operators.

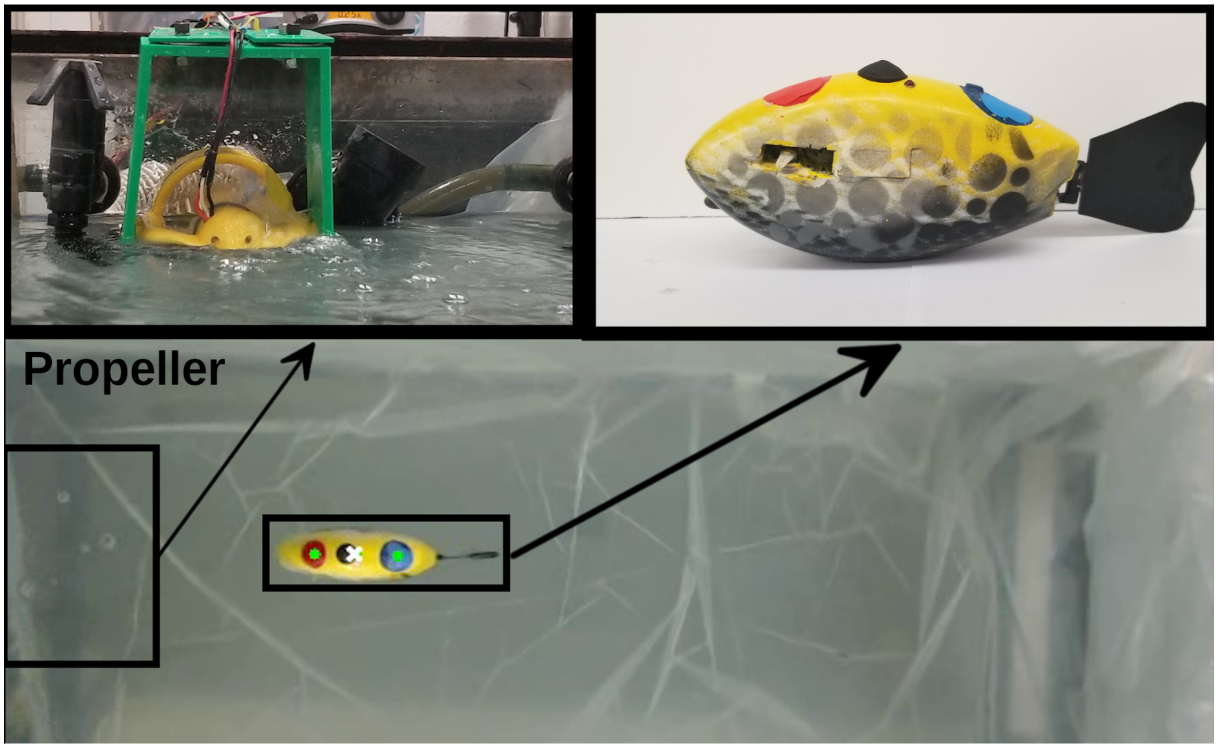


Figure 8.11. Experimental setup for creating fluid disturbances. The motor is halfway submerged in water, generating ripples with its propellers.

where

$$\begin{aligned}
\mathcal{A}_{new} &= \frac{1}{P_{total}}(\mathcal{A}P + \sum_{k=P}^{P_{total}-1} \Psi(s_{k+1}, u_{k+1})\Psi(s_k, u_k)^T) \\
\mathcal{G}_{new} &= \frac{1}{P_{total}}(\mathcal{G}P + \sum_{k=P}^{P_{total}-1} \Psi(s_k, u_k)\Psi(s_k, u_k)^T)
\end{aligned} \tag{8.26}$$

and $\mathcal{A}$ and $\mathcal{G}$ are given by (7.5). The derivation of the formula is shown in Appendix A.6. Then, the LQR gains are recomputed as shown in Section 7.4 using the updated Koopman operator. I show an outline of the process in Fig. 8.10.

The proposed method is compared to a PID controller, a widely-used model-free control method. PID feedback is quite effective, requires low computational effort, and does not

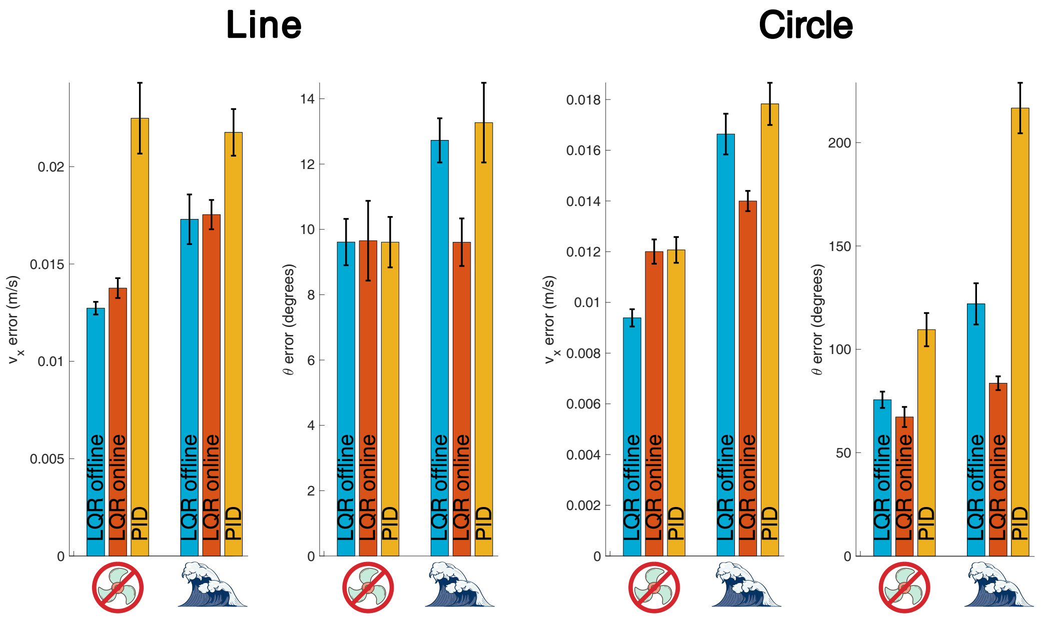


Figure 8.12. Experimental results: Average error scores for velocity and angle tracking for PID and two variants of Koopman-LQR, trained offline and updated online, with and without fluid flow indicated respectively with the waves and no-fan icons. The four subplots compare the performance for the linear and circular trajectories for each state separately. The error bars indicate the standard error. The proposed Koopman operator scheme outperforms PID in all tests. Further, updating Koopman-LQR online improves the performance in the presence of the unmodeled fluid flow. A video of experimental runs is shown at https://youtu.be/9_wx0tdDta0.

require exact knowledge of the dynamics or any of the model coefficients [**208**]. In fact, it has been shown to perform remarkably well on motion and speed control of robotic fish [**209**, **210**]. However, this method often requires intensive gain tuning. To tune the PID controller, I utilize the Ziegler-Nichols' closed-loop method [**211**]. By comparing my method to PID, I hope to demonstrate that the proposed method is a promising, data-driven feedback scheme that compares well against a popularly used, effective controller, yet without the additional overhead of intensive parameter tuning. The two methods are compared on tracking linear and circular trajectories that are described in terms of the desired orientation and forward velocity.

The comparison of the proposed Koopman-LQR method and the PID is conducted over ten trials for both types of trajectories. I further compare their performance in the presence of fluid disturbance, generated by a propeller (see Fig. 8.11). The results are presented in Fig. 8.12, which displays the average error in the two tracked states, together with the standard error. The standard error is a measure of the expected variability in the average error, contrary to the standard deviation that measures the expected variability away from the mean. Note that, for the case of LQR with gains computed offline, the data used to obtain the Koopman representation was collected in the absence of fluid disturbance.

From the results, the proposed data-driven Koopman-LQR scheme, regardless of whether it is updated online, outperforms PID in all tasks, with or without fluid disturbance, except for tracking the orientation in the linear trajectory, where both algorithms have similar performance. The difference in the performance between the Koopman-LQR and the PID is highlighted in the tracking of the circular trajectory in the presence of fluid flow, where the angle error is significantly higher for PID (Fig. 8.12). Given that angle tracking (in the linear trajectory) is the only metric where PID is comparable to the proposed method, I attribute the difference in performance in the presence of fluid flow to the robustness of the proposed approach and its ability to track the desired trajectory in a confined space in the presence of unmodeled dynamics.

In the absence of fluid disturbance, the two implementations of the proposed method (that is, with and without updating the model and LQR gains in real time) have comparable results. It is only in tracking the circular trajectory that the offline implementation tracks the desired velocity better. I conjecture that this is due to the collisions of the robotic fish with the side wall (and the unmodeled boundary conditions) that take place because of the confined space. Introducing such discontinuous disturbances likely deteriorates the learned model temporarily; yet it still outperforms the well tuned PID controller. On the other hand, the major benefit of updating the model online is, as one would expect, in the presence of fluid disturbance. There, the real-time updated method significantly outperforms the offline-trained Koopman-LQR scheme, highlighting the importance of updating the model online in environments that constantly change.

## 8.3. Discussion

In this chapter, I use the Koopman operator framework to develop derivative-based data-driven linear representations of nonlinear systems, suitable for real-time feedback. The proposed synthesis of the observable functions aims at minimizing the representation error without requiring knowledge of the dynamics. Utilizing Taylor series error bounds, I characterize the error for the analytical expression and use the error bound formula to decide the basis functions for obtaining a linear data-driven Koopman representation. I then control the linear model using LQR feedback, which enables very fast control synthesis. In fact, unless the model is updated online, the LQR gains are computed only once. I demonstrate the efficacy of the proposed approach with simulation and experimental results using a case study of a tail-actuated robotic fish.

Although the proposed method can be used for any system that can benefit from data-driven methods or reduction of the nonlinearity, underwater robotics is perhaps the most suitable application for this method, due to the inherent environmental uncertainty, the highly nonlinear dynamics, and the need for controllers that use limited computation (to preserve battery use or due to limited computational power). While this method could certainly be applied to other systems, it perhaps would not be as useful for low-dimensional systems, with known dynamics and few nonlinearities.

CHAPTER 9

# Memory-Efficient Learning of Stable Linear Dynamical Systems

This chapter presents a novel algorithm, called SOC, for learning stable LDSs for prediction and control. Using a recent characterization of matrix stability [**212**], I derive a gradient-descent algorithm that iteratively improves the reconstruction error of a projected stable model. Contrary to current top-performing methods that start from the least-squares (LS) solution and iteratively push the LDSs towards the stability region, SOC enforces stability in each step. As a result, it returns a stable LDS even after one iteration. Furthermore, whereas alternative methods terminate upon reaching stability, SOC can iterate on already stable solutions to improve the reconstruction error. It can therefore be used to further improve the solutions of other methods.

The proposed method is *provably* more memory-efficient, with an $\mathcal{O}(n^2)$ space complexity—$n$ being the state dimension—compared to $\mathcal{O}(n^4)$ of the competing alternative schemes for stable LDS. For systems with inputs, I derive the gradient directions that update both state and control linear matrices. By doing so, I expand the space of possible solutions and enable the discovery of models achieving lower error metrics compared to searching only for a stable state matrix which, to the best of my knowledge, is what the current top-performing algorithms do.

To demonstrate the superior performance of the SOC method, I test it on the task of learning dynamic textures from videos (using benchmark datasets that have been used to assess models that learn stable LDSs), as well as learning and controlling (in simulation and experiment) the Franka Emika Panda robotic arm [**106**]. When compared to the current top-performing models, a constraint generation (CG) [**4**] and a weighted least squares (WLS) [**107**] approach, the proposed method achieves an orders-of-magnitude lower reconstruction error, robustness even in low-resource settings, and better control performance.

## 9.1. Notation

In this section, I review the notation and the setup used in identifying stable linear dynamical systems from data. I consider states $x{\in}\mathbb{R}^N$, controls $u{\in}\mathbb{R}^M$ and discrete time LDSs modeled as

$$y_t{\equiv}x_{t+1}{=}Ax_t{+}Bu_t, \tag{9.1}$$

where $A{\in}\mathbb{R}^{N\times N}$ and $B{\in}\mathbb{R}^{N\times M}$ are the state and control matrices, respectively. For systems without inputs, one can simply set $B{=}0$. Let $\mathcal{S}_{A,B}{=}\{(A,B) \mid x_{t+1}{=}Ax_t{+}Bu_t\}$ denote the solution space of the matrices $A$ and $B$ that describe a LDS of the form (9.1).

Stability of linear systems is typically analyzed using the eigenvalues of the state transition matrix. Let $\{\lambda_i(A)\}_{i=1}^N$ denote the eigenvalues of an $N{\times}N$ matrix A in decreasing order of magnitude and $\mathsf{S}$ be the set of all stable matrices of size $N{\times}N$. Further, let $\rho(A){\equiv}|\lambda_1(A)|$ denote the spectral radius of $A$.

### 9.1.1. Learning Data-Driven LDSs

Given $p$ pairs of measurements $(x_t, y_t)$, learning LDSs from data typically takes the form

$$\hat{A} = \inf_{A} \frac{1}{2} \|Y - AX\|_F^2, \tag{9.2}$$

where $Y = [y_1 y_2 \ldots y_p] \in \mathbb{R}^{N \times p}$, $X = [x_1 x_2 \ldots x_p] \in \mathbb{R}^{N \times p}$, and $||\cdot||_F$ is the Frobenius norm. The LS solution is then computed as

$$A_{ls} = YX^{\dagger}. \tag{9.3}$$

where $X^{\dagger}$ denotes the Moore-Penrose inverse (or pseudo-inverse) of $X$. The optimization problem in (9.2) does not impose stability constraints on $\hat{A}$. The objective of learning stable LDSs is typically formulated as

$$\hat{A} = \inf_{A \in \mathbb{S}} \frac{1}{2} \|Y - AX\|_F^2, \tag{9.4}$$

and is highly nonconvex.

The current top-performing methods for computing stable LDSs are a constraint generation [**4**] and a weighted least squares [**107**] approach. CG formulates the optimization as a quadratic program without constraints, which is an approximation to the original problem. It then iterates on the solution to the approximate optimization by adding constraints and terminates when a stable solution is reached. WLS determines the components of the LS transition matrix that cause instability and uses a weight matrix to enforce stability, while minimizing the reconstruction error. Note that both methods consider an entire sequence of observations, say $\mathcal{D} \in \mathbb{R}^{N \times p}$, such that $X = \mathcal{D}_{[0:p-1]}$ and

$Y{=}\mathcal{D}_{[1:p]}$, thereby making the assumption that all measurements belong to a unique time-series dataset. In the case of the WLS method, this assumption is necessary and the method fails dramatically for datasets with disjoint windows of time, as I demonstrate in the results section. CG and SOC, on the other hand, do not require contiguous observations.

### 9.1.2. Subspace Methods

For high-dimensional LDSs, such as in the case of image reconstruction, it is computationally prohibitive to learn a state transition matrix. Even for small images of size $100{\times}100$ pixels, the dimensionality of the state transition matrix $A$ would be $100^4$. For such high-dimensional systems, models are obtained using subspace methods that reduce the dimensionality of the learning task. Subspace methods for learning LDSs typically use singular value decomposition (SVD) on the original dataset [**213**] decomposing the observation matrix $\mathcal{D} \approx \mathcal{U}\Sigma V^T$, where $r{<}N$ is the subspace dimension, $\mathcal{U}{\in}\mathbb{R}^{N\times r}$ and $V{\in}\mathbb{R}^{p\times r}$ are orthonormal matrices and $\Sigma{=}\{\sigma_1,...,\sigma_r\}{\in}\mathbb{R}^{r\times r}$ contains the $r$ largest singular values. Then, the learning optimization is performed on the reduced observation matrix $D_r{=}\Sigma V^T$, where $X_r{=}D_{r[0:p-1]}$ and $Y_r{=}D_{r[1:p]}$. $\mathcal{U}$ is used to project the solutions back to the original state space. For a more complete description of the subspace method, the reader can refer to [**214**, **215**].

## 9.2. SOC Algorithm

The optimization problem for finding stable LDSs has traditionally only considered solving for a stable matrix $A$ that minimizes the reconstruction loss. In this work, I formulate the objective as

$$[\hat{A},\hat{B}]=\inf_{A\in\mathbb{S},B}\frac{1}{2}\|Y-AX-BU\|_F^2, \tag{9.5}$$

to expand the solution space and solve both for a stable state matrix $A$ and a matrix $B$. I denote the least-square solution for the control system by $[A_{ls},B_{ls}]=Y\cdot[X;U]^{\dagger}$.

### 9.2.1. Optimization Objective and Gradient Descents

The proposed algorithm uses a recent characterization of stable matrices [**212**]. Specifically, a matrix $A$ is stable if and only if it can be written as $A=S^{-1}OCS$, where $S$ is positive definite, $O$ is orthogonal, and $C$ is a positive semidefinite contraction (that is, $C$ is a positive semidefinite matrix with norm less than or equal to 1). Using this property, I formulate the optimization problem as

$$[\hat{A},\hat{B}]=\inf_{S\succ 0,O \text{ orthogonal},C\succeq 0,\|C\|\leq 1}\frac{1}{2}\|Y-S^{-1}OCSX-BU\|_F^2, \tag{9.6}$$

where $\hat{A}\equiv S^{-1}OCS$. I then derive the gradient directions with respect to the four matrices $S$,$O$,$C$, and $B$ as follows:

$$\nabla_S f(S,O,C,B)=S^{-T}EX^TS^TC^TO^TS^{-T}-C^TO^TS^{-T}EX^T \tag{9.7}$$

$$\nabla_O f(S,O,C,B)=-S^{-T}EX^TS^TC^T \tag{9.8}$$

$$\nabla_C f(S,O,C,B)=-O^TS^{-T}EX^TS^T \tag{9.9}$$

$$\nabla_B f(S,O,C,B)=-EU^T \tag{9.10}$$

where $E{=}Y{-}S^{-1}OCSX{-}BU$ and I use a gradient descent algorithm to reach a local minimum of the reconstruction cost. The derivation of the gradients is presented in Appendix A.7. Note that, contrary to CG and WLS that search stable LDS in $\mathcal{S}_{A,B_{ls}}$ by iterating over only $A$, my algorithm updates both linear matrices $A$ and $B$, thereby expanding the feasible solution space to $\mathcal{S}_{A,B}$, where $\mathcal{S}_{A,B}{\supset}\mathcal{S}_{A,B_{ls}}$. Henceforth, I refer to the proposed algorithm as SOC.

## 9.3. Experiments

I implement LS, CG, WLS, and the proposed SOC method for learning LDSs and compare their performance on dynamical systems with and without control inputs. For systems without inputs, I focus on learning dynamic texture from frame sequences extracted from videos using standard benchmark datasets [**216**–**218**]. For systems with inputs, I use experimental data from the Franka Emika Panda robotic manipulator and illustrate the learning and control performance of all the methods considered. I split the results in three parts: memory requirements, reconstruction error performance, and control performance. For an impartial assessment, I perform all comparisons in MATLAB using the publicly available code of the CG and WLS algorithms[1]. All simulations are performed using MATLAB R2019b on a machine with a 14-core Intel E5-2680v4 2.4-GHz CPU with 20GB RAM.

### 9.3.1. Memory Usage

First, I compare the three algorithms on memory requirements. For an objective comparison, I only measure the size of all MATLAB workspace variables created by the algorithms.

[1]https://github.com/huangwb/LDS-toolbox

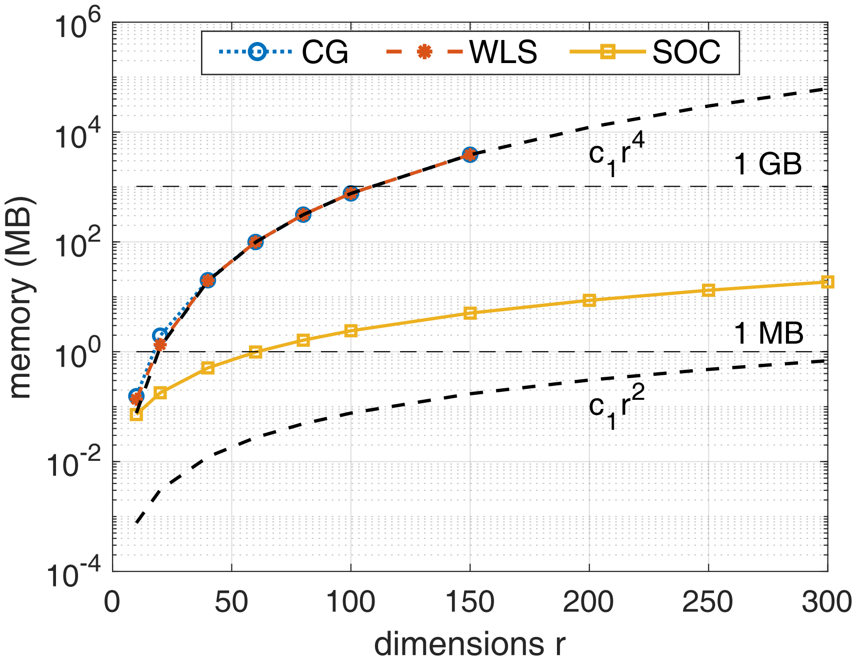


Figure 9.1. Memory requirements as a function of dimensions $r$, where $c_1{=}8/2^{20}$. CG and WLS scale proportionately to $r^4$, whereas SOC scales proportionately to $r^2$. For $r{=}150$, SOC uses about 5.04 MB, whereas CG and WLS about 3.78 GB. Due to memory limits, WLS and CG failed to run at higher dimensions.

That is, I consider a matrix with 4 double-precision cells to use 32 bytes. I compare the algorithms on a sequence of frames extracted from a coffee cup video downloaded from Youtube[2]. I use this video because it exhibits dynamical motion and has a sufficient number of frames to allow for relatively higher subspace dimensions (the SVD decomposition limits the subspace dimension to be no larger than the number of frames).

The results are shown in Figure 9.1. SOC scales proportionately to $r^2$, whereas both CG and WLS scale proportionately to $r^4$. At $r{=}150$, SOC uses about 5 MB of memory; CG and WLS use about 3.78 GB of memory and fail to run at higher dimensions due to memory constraints. Though such high dimensions may perhaps seem out of scope for

[2]https://www.youtube.com/watch?v=npkBC4GYodg

the image reconstruction examples demonstrated next, they can typically occur in the field of robotics. For example, a recent study [**57**] used a linear data-driven Koopman representation with dimensions $r$=330 to identify and control a pneumatic soft robotic arm. For this dimension, WLS and CG would require about 88 GB of memory and SOC would need about 25 MB. As a result, only SOC would be able to successfully train a stable Koopman model on a standard personal laptop and, as I show in the control performance section, failing to impose stability on the learned model can lead to unsafe robot movements.

### 9.3.2. Error Performance

To measure the predictive accuracy of the learned representations, I use three benchmark datasets: UCLA [**216**], UCSD [**217**], and DynTex [**218**]. The UCLA dataset consists of 200 gray-scale frame sequences that demonstrate 50 different categories of dynamic motion (e.g. flame flickering, wave motion, flowers in the wind), each captured from 4 different viewpoints. Every frame sequence contains 75 frames of size 48×48 pixels. The UCSD dataset consists of 254 frame sequences showing highway traffic in different weather conditions. Each sequence contains between 42 and 52 frames of size 48×48 pixels. For the DynTex dataset, I use 99 sequences from 5 groups of dynamic texture (`smoke` and `rotation` from the Beta subset and `foliage`, `escalator`, and `flags` from the Gamma subset) that exhibit periodic motion. The frames are of size 352×288 pixels. I convert the frames to grayscale and use the bicubic interpolation algorithm implemented in the Python library `pillow` to scale down the frames without ratio distortion down to 48×39 pixels. Each DynTex sequence contains between 250 and 1576 frames.

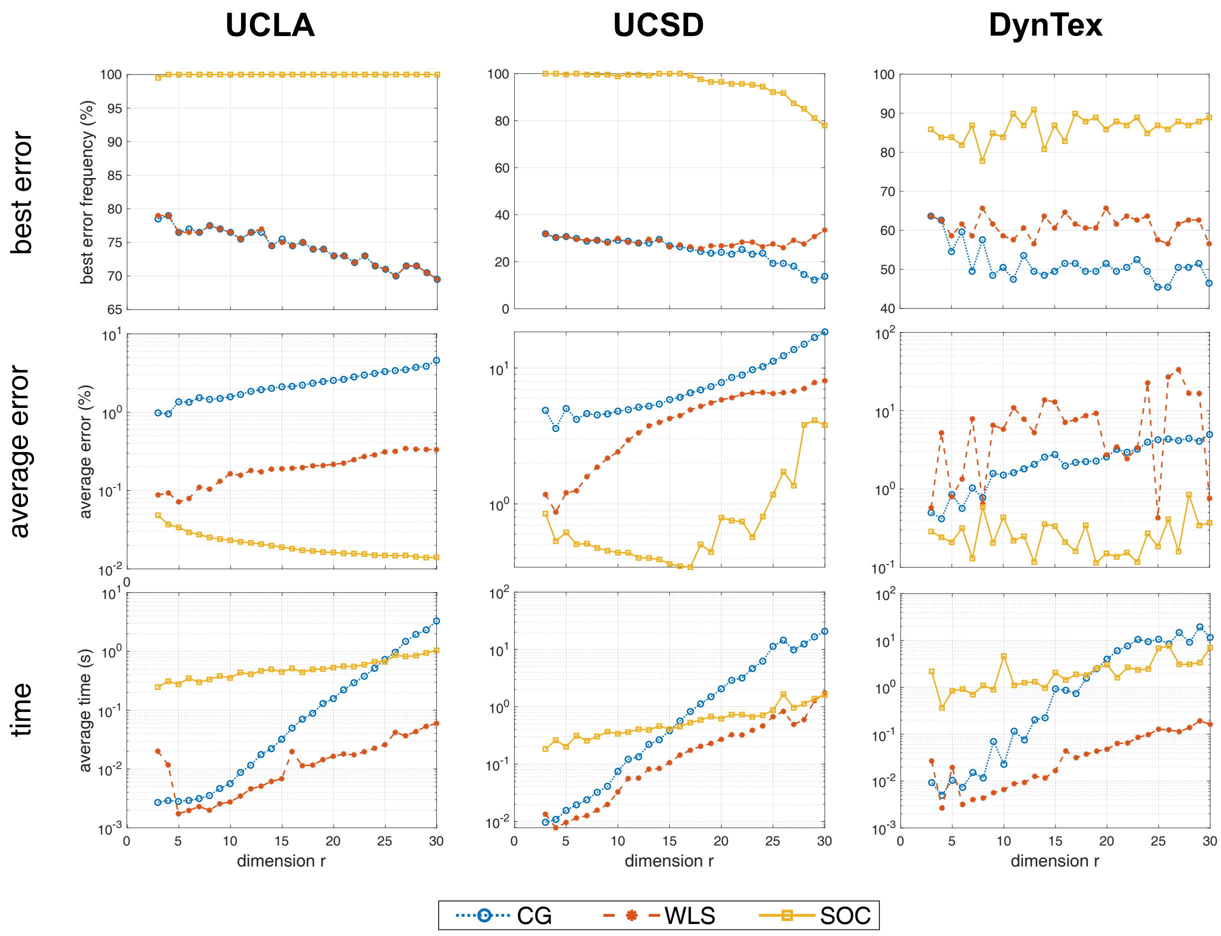


Figure 9.2. Learning performance of CG, WLS, and SOC for varying subspace dimensions performed on three datasets: UCLA, UCSD, and DynTex. In all cases, SOC has the highest best error frequency, has lower average error and, in terms of execution time, scales better than the other methods.

As explained in Section 9.1, the dimensionality of images can be prohibitively high and cause slow computations or memory failures: the transition matrix for an image of size as small as 48×48 pixels would require hundreds of TBs for CG and WLS to run. For this reason, I use subspace methods to reduce the problem dimensionality. For each dataset, I consider a set of subspace dimensions $r \in \{3,30\}$. Then, for each dimension, I use the four methods (LS, CG, WLS, and SOC) to obtain an LDS for each of the frame sequences. To compare the performance of the four algorithms, I use the reconstruction error relative to the LS solution: $e(\hat{A}){=}\frac{e(\hat{A})-e(A_{ls})}{e(A_{ls})}{\times}100$.

I report the results in Figure 9.2 and focus on three metrics: best error frequency, average reconstruction error, and execution time. The best error plots the percentage of frame sequences for a given dimension for which an algorithm computed the best relative error (that is, lower than or equal to the other two methods). The average error and time plots show the average reconstruction error and average execution time of all frame sequences for each dimension, respectively.

The results demonstrate certain patterns that are present in all datasets considered. First, SOC computes the best error for more frame sequences than the other methods across any dimension. In the UCLA and UCSD datasets, the SOC best error frequency reaches 100% for the majority of the dimensions; compared to less than 80% (for UCLA) and 40% (for UCSD) shown by both of the other two methods. This means that, for the aforementioned datasets, CG and WLS almost never find a better solution than SOC. While for the DynTex dataset the differences are not as pronounced, SOC still computes the best error the majority of the time and about 20% more often than each other method. Second, SOC has orders-of-magnitude lower average relative error across all dimensions and datasets. Last, in terms of the execution time, SOC is slower than CG and WLS for low dimensions ($r{<}20$). However, it scales better than the other two methods, such that it becomes faster than CG for $r{>}20$. For the UCSD dataset, SOC and WLS become comparable in terms of average execution time near $n{=}30$. This observation is in line with the fact that CG and WLS are high space-complexity algorithms that may even fail to perform at high dimensions due to memory limitations.

| | SOC | CG | WLS | SOC | CG | WLS |
|---|---|---|---|---|---|---|
| | `steam` (r = 20) | | | `steam` (r = 40) | | |
| $\lvert\lambda_1\rvert$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $\sigma_1$ | 1.06 | 1.03 | 1.07 | 1.10 | 1.03 | 1.10 |
| $e(\hat{A})$ | **12.32** | 28.05 | 24.94 | **5.59** | 24.90 | 21.27 |
| time | 0.36 | **0.22** | 0.53 | **1.48** | 9.76 | 15.82 |
| | `fountain` (r = 20) | | | `fountain` (r = 40) | | |
| $\lvert\lambda_1\rvert$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $\sigma_1$ | 1.11 | 1.00 | 1.11 | 1.43 | 1.01 | 1.43 |
| $e(\hat{A})$ | **0.001** | 1.07 | 0.004 | **0.0005** | 2.97 | 0.0007 |
| time | 1.52 | 0.48 | **0.18** | 3.18 | 15.40 | **0.84** |

Table 9.1. Performance on `steam` and `fountain` sequences from the MIT database.

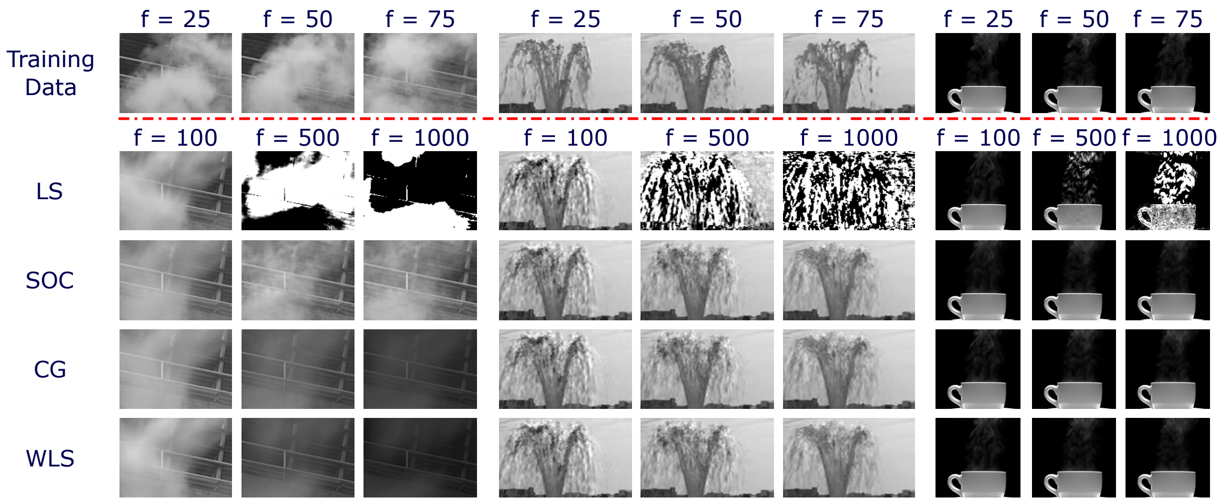


Figure 9.3. Synthesized sequences generated by LS, SOC, CG, and WLS for $r$=40.

Next, I compare the three methods on the `steam` sequence (composed of 120×170 pixel images) and the `fountain` sequence (composed of 150×90 pixel images) from the MIT temporal texture database [**219**]. Results are shown in Table 9.1. To show the effect on the predictive quality of the solutions, I plot the frames reconstructed from the learned LDS for each method in Figure 9.3. Next to the `steam` and `fountain` frame sequences, I add the `coffee cup` sequence used in Figure 9.1.

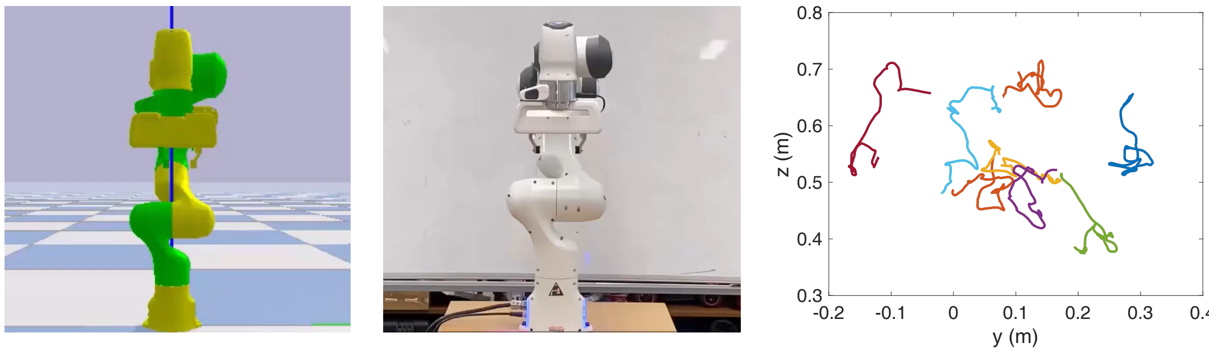


Figure 9.4. From left to right: simulation environment, physical robot, and experimental training data.

### 9.3.3. Control

In this section, I demonstrate the superior performance of SOC in control systems. Using experimental data gathered from the robotic arm Franka Emika Panda, I illustrate the improvement in both the reconstruction error of the learned model and the control performance. To use CG and WLS to compute a stable $\hat{A}$, I use the LS solution for the control matrix and modify the objective to

$$\hat{A} = \inf_{A \in \mathbb{S}} \frac{1}{2} \|Y' - AX\|_F^2, \tag{9.11}$$

where $Y' = Y - B_{ls}U$. The learning performance of the algorithms is then measured as the % error increase when compared to the LS solution $(A_{ls}, B_{ls})$. Note that this error depends both on $\hat{A}$ and $\hat{B}$; for WLS and CG, I use the LS solution for the control matrix $B = B_{ls}$, whereas SOC computes both $A$ and $B$.

I collected training data on the experimental platform at 50 Hz, using a controller to manually move the robotic arm. I gathered 400 measurements (8 seconds) in eight separate runs. The training data, along with the experimental and simulation environments used

| Measurements | 50 | 75 | 100 | 150 | 200 | 300 | 500 |
|---|---|---|---|---|---|---|---|
| SOC | **0.01** | **0.56** | **0.0001** | **0.06** | **0.05** | **0.09** | **0.05** |
| CG | 50.14 | 32.77 | 11.66 | - | - | 1.40 | 0.23 |
| WLS | 17.00 | - | - | 124.01 | 36.63 | 25.17 | 42.01 |

Table 9.2. Errors of stable LDS using experimental data from the Franka Emika Panda manipulator.

in this section are shown in Figure 9.4. Table 9.2 compares the performance of the SOC, CG, and WLS algorithms on learning stable models for the Franka Emika Panda robotic manipulator using experimental data. The performance is compared for different numbers of measurements $p$. As the data show, SOC is the only algorithm that never fails to find stable solutions, regardless of the amount of training data. As more measurements are used, the LS solution itself becomes more stable and CG and WLS are both able to converge to stable solutions. Further, the quality of CG solutions improves with more training measurements; the performance of SOC remains robust throughout the testing cases.

In Figure 9.5, I plot the reconstruction error for the three methods for different training data sizes. In this setting, however, measurement sets $(x_t, y_t, u_t)$ are randomly drawn from the training data such that the matrices $Y$ and $X$ have discontiguous measurements. Note how such a choice worsens the performance of WLS that assumes continuity in the observation matrices. On the other hand, CG and SOC are similar in learning performance.

With regard to controlling the system, I use LQR control computed using the models from each algorithm and simulate tracking a figure-8 pattern. The states are the $x$,$y$,$z$ coordinates of the end effector, the 7 joint angles of the arm, and the 7 joint angular velocities. The trajectory is generated in the $y{-}z$ plane for the end effector; the desired angle

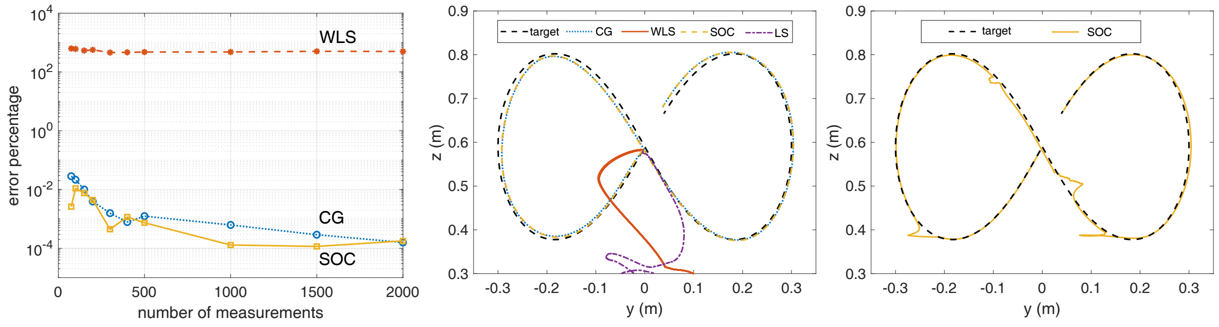


Figure 9.5. Control performance in simulation using experimental measurements from the Franka Emika Panda robotic arm. The left figure shows the reconstruction error of the learned models for a varying number of measurements sampled randomly from the training set; the middle figure shows the performance of the controllers after training with 100 measurements sampled randomly (2 seconds worth of data); the right figure shows the control performance of SOC after manually introducing disturbances to the position of the end effector.

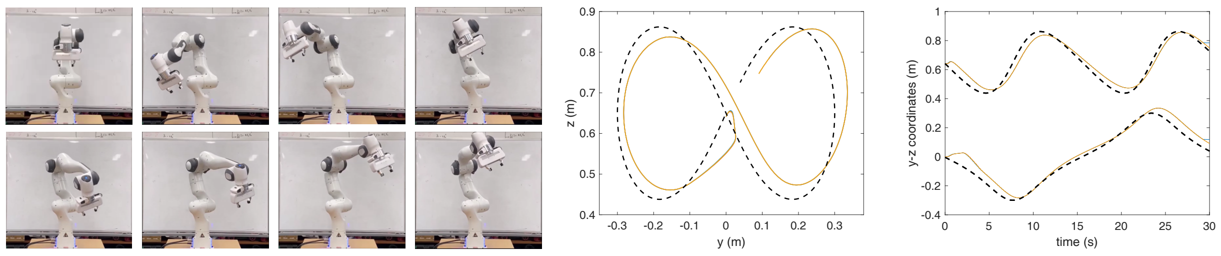


Figure 9.6. Experimental tracking of a figure-8 pattern using the Franka Emika Panda robotic manipulator. The left figure shows, from top to bottom, snapshots of the control maneuver; the rest figures show the trajectories of three trials. The three trials are almost identical, showing the robustness of the method. The applied control is computed with an LQR policy using the stable LDS system obtained from the SOC algorithm. The training data are obtained using 600 measurements.

configurations of the robotic arm are solved offline using inverse kinematics; the desired angular joint velocities are set to 0. LQR control is generated using $Q{=}\text{diag}([c_i]){\in}\mathbb{R}^{17\times 17}$, where $c_i{=}1$ for $i{\in}\{1{,}10\}$ and 0 elsewhere and $R{=}0.1{\times}I_{7\times 7}$.

The LS model is unstable and fails at the task. Similarly, WLS—despite the stable model—performs poorly, highlighting the need for both stability and fidelity of the learned representation. On the other hand, CG and SOC are similar in performance.

To measure robustness across the initial conditions, I run 50 trials, varying both the $y$ and $z$ initial positions with displacements sampled uniformly in $\mathcal{U}(-0.1,0.1)$. Across all trials, LS has an average error of 7556, WLS scores 38.73, CG scores 0.0810 and SOC scores 0.0799.

Then, I test LQR control computed on the LDS obtained from the SOC algorithm in an experiment to demonstrate that the simulation results are indicative of the performance in a physical experiment. Figure 9.6 shows the control performance of three trials tracking a figure-8 pattern. Due to COVID-19 limitations, I were unable to extend the experimental tests. However, these results serve primarily to experimentally validate the SOC algorithm and illustrate that the simulation results are an accurate prediction of the experimental behavior as well.

## 9.4. Discussion

In this work, I introduce a novel algorithm for computing stable LDSs. Compared to the current top-performing alternatives, the proposed scheme is significantly more memory-efficient and, as a result, scales better for high-dimensional systems often encountered in image processing and robotic applications. Further, the suggested method outperforms the alternatives in terms of the error and control performance, as demonstrated on three benchmark datasets and the Franka Emika Panda robotic arm experiments. These features make it a promising tool for compression and data-driven system identification tasks. Coupled with the ongoing research around Koopman-operator-based nonlinear control, this algorithm can be a promising candidate for high-dimensional nonlinear control.

The proposed method can improve robotic tasks that are safety-critical, particularly those that include a human-in-the-loop (such as rehabilitation devices and prosthetics) where the human-robot interaction dynamics are not known ahead of time. For such tasks, a robotic platform prioritizes stability and safety during operation. Unstable data-driven models may lead to catastrophic robotic behavior, as I demonstrate in simulations with the Franka Emika Panda robotic arm. This work provides a mechanism for online learning of models that satisfy stability constraints, improving the safety and reliability of closed-loop control of those systems.

CHAPTER 10

# Learning Stable Models for Prediction and Control Using Koopman Operators

This chapter demonstrates the benefits of imposing stability on data-driven Koopman operators. The data-driven identification of stable Koopman operators (DISKO) is implemented using the algorithm derived in Chapter 9 that computes the nearest *stable* matrix solution to a least-squares reconstruction error. As a first result, I derive a formula that describes the prediction error of Koopman representations for an arbitrary number of time steps, and which shows that stability constraints can improve the predictive accuracy over long horizons. As a second result, I determine formal conditions on basis functions of Koopman operators needed to satisfy the stability properties of an underlying nonlinear system. As a third result, I derive formal conditions for constructing Lyapunov functions for nonlinear systems out of stable data-driven Koopman operators, which I use to verify stabilizing control from data. Lastly, I demonstrate the benefits of DISKO in prediction and control with simulations using a pendulum and a quadrotor and experiments with a pusher-slider system. The chapter is complemented with a video: `https://sites.google.com/view/learning-stable-koopman`.

## 10.1. Stable Koopman Operators

In this section, I derive the error induced by approximate Koopman operators over an arbitrary number of time steps into the future. I then use the error expression to motivate

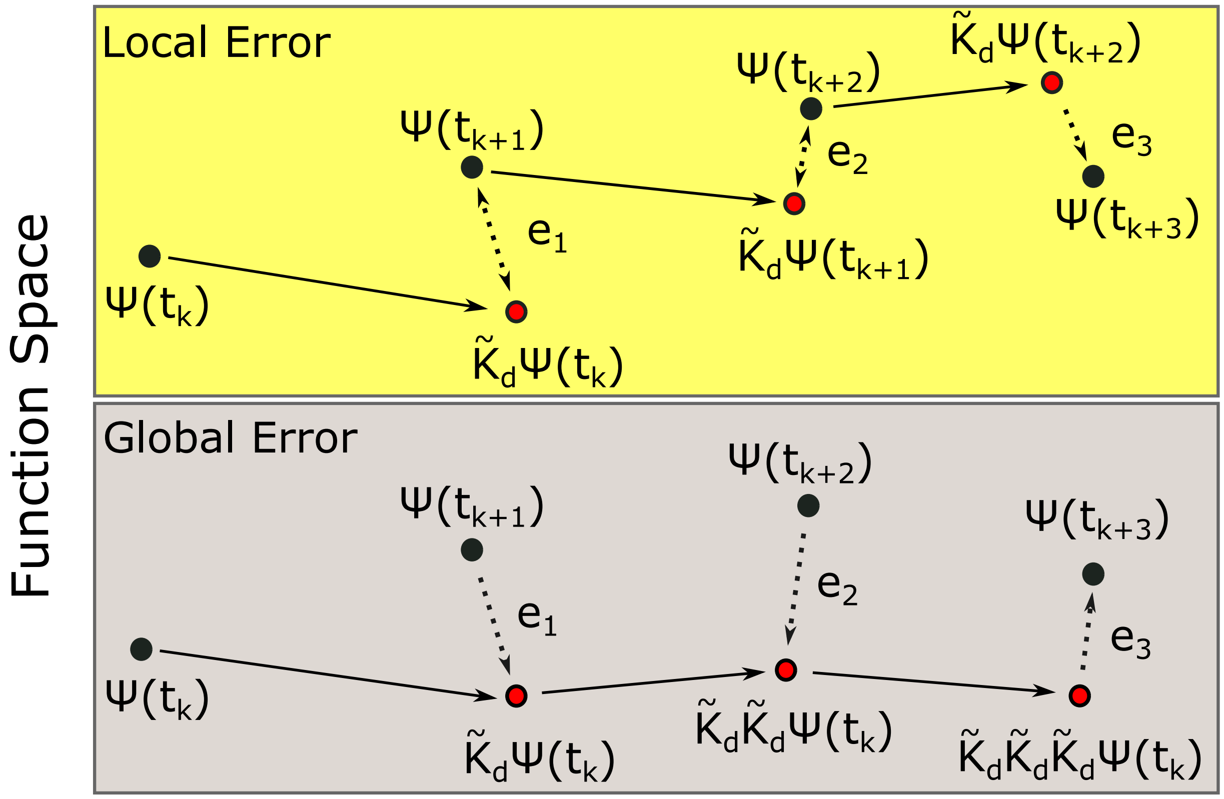


Figure 10.1. Local and global errors (in time) induced by approximate Koopman operators. The local error considers the accuracy of the model across a single time step; the global error considers the accuracy of the model across all time steps and is a more representative metric of long-term accuracy, assuming states are not updated in every time step.

imposing stability on the operators, sometimes even in cases when the underlying system is unstable. Then, I present conditions on the basis functions that are consistent with stable Koopman operators. Last, I demonstrate how to construct Lyapunov functions using stable data-driven Koopman operators.

### 10.1.1. Global Error of Approximate Koopman Operators

**10.1.1.1. Notation.** At time $t_0{+}n\Delta t$, I use $\Psi(s(t_0{+}n\Delta t))$ to indicate the true value of the basis function evaluated with the true state value $s(t_0{+}n\Delta t)$ and $\tilde{\Psi}_n$ to indicate the approximate solution, where $n{\in}\mathbb{Z}^+$ indicates the number of time steps into the future. I use $e_n$ to be the local error at the $n$th time step induced by the approximate Koopman operator $\tilde{\mathcal{K}}_d$, that is

$$e_n{\equiv}\Psi(s(t_0{+}n\Delta t)){-}\tilde{\mathcal{K}}_d\Psi(s(t_0{+}(n{-}1)\Delta t)), \tag{10.1}$$

which assumes that the Koopman operator propagates the true value of the basis functions from the previous time step. Similarly, I use $E_n$ to refer to the global error at the $n$th time step:

$$E_n{\equiv}\Psi(s(t_0{+}n\Delta t)){-}\tilde{\mathcal{K}}_d^n\Psi(s(t_0)), \tag{10.2}$$

which, for $n{=}1$, matches the local error (10.1). However, note that the global error (10.2) is not an accumulation of the local errors (10.1): $E_n{\neq}\sum_i^n e_i$. The difference between local and global error is illustrated in Fig. 10.1.

**10.1.1.2. Derivation of Global Error.** Consider the true solution at some time step $t_0{+}n\Delta t$, which can be written using (10.1) as

$$\Psi(s(t_0{+}n\Delta t)){=}\tilde{\mathcal{K}}_d\Psi(s(t_0{+}(n{-}1)\Delta t)){+}e_n. \tag{10.3}$$

Similarly,

$$\Psi(s(t_0+(n-1)\Delta t))=\tilde{\mathcal{K}}_d\Psi(s(t_0+(n-2)\Delta t))+e_{n-1}. \tag{10.4}$$

Plugging (10.4) into (10.3)

$$\Psi(s(t_0+n\Delta t))= \tilde{\mathcal{K}}_d\Big(\tilde{\mathcal{K}}_d\Psi(s(t_0+(n-2)\Delta t))+e_{n-1}\Big)+e_n \tag{10.5}$$

$$= \tilde{\mathcal{K}}_d^2\Psi(s(t_0+(n-2)\Delta t))+\tilde{\mathcal{K}}_d e_{n-1}+e_n. \tag{10.6}$$

Recursively expressing the solution in terms of the true values of the basis functions in the previous steps and the corresponding local error yields

$$\Psi(s(t_0+n\Delta t))=\tilde{\mathcal{K}}_d^n\Psi(s(t_0))+\sum_{i=0}^{n-1}\tilde{\mathcal{K}}_d^i e_{n-i}.$$

Therefore, using (10.2), the global error at $t_0+n\Delta t$ is

$$E_n=\sum_{i=0}^{n-1}\tilde{\mathcal{K}}_d^i e_{n-i}.$$

**10.1.1.3. Global Error Bound.** Next, I compute a bound for the global error bound. I use an induced norm that satisfies the properties of triangle inequality ($\|A+B\|\leq \|A\|+\|B\|$), subordinance ($\|Ax\|\leq\|A\|\|x\|$) and submultiplicativity ($\|AB\|\leq\|A\|\cdot\|B\|$),

such that

$$\begin{aligned}
& \|E_n\|= \|\sum_{i=0}^{n-1}\tilde{\mathcal{K}}_d^i e_{n-i}\| \\
\text{(triangle inequality)} \quad & \leq \sum_{i=0}^{n-1}\|\tilde{\mathcal{K}}_d^i e_{n-i}\| \\
\text{(subordinance)} \quad & \leq \sum_{i=0}^{n-1}\|\tilde{\mathcal{K}}_d^i\|\cdot\|e_{n-i}\| \\
\text{(submultiplicativity)} \quad & \|E_n\|\leq \sum_{i=0}^{n-1}\|\tilde{\mathcal{K}}_d\|^i\cdot\|e_{n-i}\|.
\end{aligned}$$

Assuming that the local error is bounded, that is there exists $\|e_{max}\|$ such that $\|e_i\|\leq \|e_{max}\|\ \forall\ i\in[1,n]$, I can simplify the upper bound to the global error to

$$\|E_n\|\leq\|e_{max}\|\cdot\sum_{i=0}^{n-1}\|\tilde{\mathcal{K}}_d\|^i. \tag{10.7}$$

Note that

$$\|e_{max}\|=0\Longleftrightarrow\|E_{k+n}\|=0,$$

which is true for invariant subspaces.

From (10.7), if $\tilde{\mathcal{K}}_d$ is unstable, then the power of the matrix norm diverges as the number of time steps increases. This means that an unstable $\tilde{\mathcal{K}}_d$ amplifies exponentially even small errors and thus renders the long-term prediction of the Koopman representation impractical. *Instead, I propose that one should use a stable operator that generates similar local errors, but which also has the additional benefit of numerical stability over long-term predictions.* Note that it is possible that such a stable operator can generate similar local errors even for unstable dynamics. In fact, later in Section 10.3, I show an example where a stable operator improves stabilization of a quadrotor that is unstable.

Note that the eigenvalue profile of the Koopman model can be at times misleading in terms of bounding the power of the operator, because the upper bound can be itself large, as is pointed out in [**220**–**222**]. Such scenarios are shown to occur for moderately large dimensions of the operator ($W{\approx}100$) and do not apply in the examples of this chapter. Exploiting the conditions that prevent large error growth in the transient response of a system, which is related to the strong stability property [**223**], for high-dimensional operators is left for future work.

### 10.1.2. Stability-Based Conditions for Koopman Basis Functions

Given the importance of the spectral properties of the Koopman operator on the accuracy of long-term predictions, I next consider the implications of stability on the choice of basis functions. Specifically, I relate the stability properties of original nonlinear dynamics to the stability properties of the Koopman representation and present necessary conditions for the basis functions so that they are consistent with dynamics that have a stable (or an asymptotically stable) equilibrium. Although in practice one can impose stability on arbitrary Koopman models, choosing basis functions that are not consistent with stable dynamics could lead to unstable representations and worse training errors. I present the analysis for continuous-time dynamics, which are arguably the default expression, but equivalent relationships can be extended to the discrete-time case.

Consider a nonlinear dynamical system with states $s(t) \in \mathcal{S}$ and dynamics of the form

$$\frac{d}{dt} s(t) = \mathrm{f}(s(t)) \tag{10.8}$$

with associated Koopman dynamics given by

$$\frac{d}{dt} \Psi(s(t)) = \mathcal{K} \Psi(s(t)). \tag{10.9}$$

If the two representations (10.8) and (10.9) are equivalent throughout the state space, that is, they evolve the dynamics in identically the same way, then I argue that certain properties must be true.

**Definition 3.** *For a nonlinear dynamical system* (10.8) *and an equivalent Koopman representation* (10.9)*, the following are true:*

*(1) A trajectory $s^*(t)$ is an equilibrium for the nonlinear dynamical system* (10.8) *if and only if it is an equilibrium for the equivalent Koopman dynamics* (10.9)*. That is,*

$$\mathrm{f}(s^*(t)) = 0 \Longleftrightarrow f(\Psi(s^*(t))) = 0.$$

*(2) A trajectory $s^*(t)$ is Lyapunov stable for the equivalent nonlinear dynamical system* (10.8) *if and only if it is Lyapunov stable for the Koopman dynamics* (10.9)*. That is, there exist $\epsilon_s > 0$ and $\epsilon_\Psi > 0$ such that*

$$\|s(t)\| < \epsilon_s \Longleftrightarrow \|\Psi_i(s(t))\| < \epsilon_\Psi \ \forall \ i \in \mathbb{Z}^{+1}, t \geq 0.$$

*(3) A trajectory $s(t)$ is asymptotically stable in a region of the state space $\mathcal{D}\subseteq\mathcal{S}$ for the nonlinear dynamical system* (10.8) *if and only if the equivalent trajectory $\Psi(s(t))$ is also asymptotically stable in $\mathcal{D}$ for the Koopman dynamics* (10.9)*. That is, given $s(0)\in\mathcal{D}$,*

$$\lim_{t\to\infty} s(t)=0 \Longleftrightarrow \lim_{t\to\infty}\Psi(s(t))=0.$$

Note that the conditions on the Koopman basis functions in Definition 3 consider equivalent Koopman representations and nonlinear dynamics. These conditions need to be satisfied even for approximate Koopman operators in order for the Koopman representation to be consistent with the stability properties of the original system. Next, I use Definition 3 to relate the stability properties of the original nonlinear dynamics to the properties of a Koopman operator and its associated basis functions. Using this relationship, I derive necessary conditions for both the operator and admissible basis functions associated with stable nonlinear systems.

**10.1.2.1. Conditions on Koopman Operators for Stable Systems.** First, I derive stability properties for a Koopman representation that is consistent with its associated original stable, in the sense of Lyapunov, system. The analysis rests on Assumption 8.

**Assumption 8.** *All states $s(t)\in\mathbb{R}^N$ of a nonlinear dynamical system* (10.8) *remain bounded for all $t$. That is, for some $\epsilon\geq 0$, $\|s(t)\|\leq\epsilon$ for all $t\geq 0$.*

[1]I only constraint the norms of each individual basis function, instead of the norm of $\Psi$, to be bounded due to the fact that, for an infinite-dimensional Koopman operator, the sum of norms of basis functions that are individually bounded could still be infinite.

Next, I prove that only a stable Koopman operator can accurately represent nonlinear dynamics that are Lyapunov stable.

**Definition 4.** *Consider a nonlinear dynamical system* (10.8) *with bounded states* $s(t){\in}\mathbb{R}^N$*. I define* $\mathcal{D}_\epsilon{\subseteq}\mathcal{S}$ *as a region of the state space such that, if* $\|s(t)\|{\leq}\epsilon$*, then* $s(t){\in}\mathcal{D}_\epsilon$ *for all* $t{\geq}0$*.*

**Theorem 4** (Lyapunov Stability)**.** *Consider a nonlinear dynamical system* (10.8) *and an equivalent Koopman representation* (10.9)*. The solution* $s(t){=}0$ *is a Lyapunov stable equilibrium if and only if the Koopman operator* $\mathcal{K}$ *is stable.*

Proof. From Definition 3, $s(t){=}0$ is a Lyapunov stable solution for the nonlinear dynamics if and only if it is also a Lyapunov stable solution for the Koopman dynamics (10.9):

$$\|s(t)\|{<}\epsilon_s{\Longleftrightarrow}\|\Psi_i(s(t))\|{<}\epsilon_\Psi, \quad t{\geq}0. \tag{10.10}$$

From the definition of stability for linear state space models, $\Psi(s(t)){=}0$ is a Lyapunov stable equilibrium for the Koopman dynamics (10.9) if and only if $\mathcal{K}$ is stable, specifically $\mathrm{Re}[\lambda_i]{\leq}0$, for each eigenvalue $\lambda_i$ of the operator $\mathcal{K}$. That is,

$$\|\Psi_i(s(t))\|{<}\epsilon_\Psi{\Longleftrightarrow}\mathrm{Re}[\lambda_i]{\leq}0, \quad t{\geq}0 \tag{10.11}$$

Then, from (10.10) and (10.11), $s(t){=}0$ is a Lyapunov stable solution for the nonlinear dynamics if and only if $\mathcal{K}$ is stable:

$$\|s(t)\|{<}\epsilon_s{\Longleftrightarrow}\mathrm{Re}[\lambda_i]{\leq}0, \quad t{\geq}0.$$

□

Theorem 4 proves that a Koopman operator that accurately represents a Lyapunov stable nonlinear system must itself be stable. Therefore, even approximate Koopman operators must satisfy this condition in order to satisfy the stability properties. Given a part of the state space $\mathcal{D}_\epsilon$ that bounds all trajectories $s(t)$ of the nonlinear system that start within $\mathcal{D}_\epsilon$, a Koopman operator trained with measurements from $\mathcal{D}_\epsilon$ should be stable (see Theorem 4).

Even when no finite-dimensional Koopman representation can capture the nonlinear dynamics with full fidelity, the approximate operator should be stable such that the system states do not grow unbounded and exit $\mathcal{D}_\epsilon$. Therefore, it is a necessary and sufficient condition that the Koopman operator is stable.[2]

**10.1.2.2. Conditions on Koopman Operators for Asymptotically Stable Systems.** Next, I derive stability conditions for a Koopman representation that is consistent with its associated *asymptotically* stable system. The analysis rests on Assumption 9 and further assumes that all states of a nonlinear dynamical system (10.8) lie inside a domain of attraction $D_0 \subseteq \mathbb{R}^N$, formally defined in Definition 5.

**Assumption 9.** *The nonlinear dynamical system* (10.8) *has a single asymptotic equilibrium.*

[2]Note that the notion of global stability is considered only in the sense of asymptotic stability and not Lyapunov stability. The system states, even when unbounded, always lie inside $\mathbb{R}^N$. Considering the entire (infinite) state space as a region of stability, that is $\mathcal{D}_\epsilon = \mathbb{R}^N$, all nonlinear dynamical systems can be thought of as globally Lyapunov stable, a property that caries no meaning.

**Definition 5.** *Given a nonlinear dynamical system* (10.8) *and an asymptotically stable solution* $s(t)=0$*, the domain of attraction is*

$$\mathcal{D}_0 \triangleq \{s_0 \in \mathcal{D} : \text{if } s(0)=s_0, \text{ then } \lim_{t\to\infty} s(t)=0\}.$$

Note that for multiple asymptotic equilibria, there need to be separate regions of attractions, which must in turn be represented by separate Koopman operators. In this work, I focus on obtaining a single Koopman operator consistent with a single asymptotic equilibrium. For many systems, due to the presence of friction, this can often be represented as the zero-velocity state. For nonlinear systems with multiple equilibria points, one can use work in [**224**] to obtain multiple local Koopman representations.

Next, I prove that only a Koopman operator that is Hurwitz (all eigenvalues are strictly in the left half-plane) satisfies the stability properties of nonlinear dynamics that are asymptotically stable.

**Theorem 5** (Asymptotic Stability)**.** *Consider a nonlinear dynamical system* (10.8) *and an equivalent Koopman representation* (10.9)*. The solution* $s(t)=0$ *is an asymptotically stable equilibrium if and only if the Koopman operator* $\mathcal{K}$ *is Hurwitz.*

Proof. From Definition 3, $s(t)=0$ is an asymptotically stable solution for the nonlinear dynamical if and only if it is also an asymptotically stable solution for the Koopman dynamics (10.9). That is,

$$\lim_{t\to\infty} s(t)=0 \Longleftrightarrow \lim_{t\to\infty} \Psi(s(t))=0.$$

From the stability properties of linear systems, $\Psi(s(t))=0$ is an asymptotically stable solution for the Koopman dynamics (10.9), if and only if $\mathcal{K}$ must be Hurwitz. That is,

$$\lim_{t\to\infty}\Psi(s(t))=0 \Longleftrightarrow \mathrm{Re}[\lambda_i]<0, \quad t\geq 0$$

for each eigenvalue $\lambda_i$ of the operator $\mathcal{K}$. Then, $s(t)=0$ is an asymptotically stable solution for the nonlinear dynamics if and only if the Koopman operator $\mathcal{K}$ of the associated Koopman dynamics is Hurwitz. That is,

$$\lim_{t\to\infty}s(t)=0 \Longleftrightarrow \mathrm{Re}[\lambda_i]<0, \quad t\geq 0.$$

□

Definition 3 and Theorems 4 and 5 present conditions for a Koopman representation, specifically on admissible basis functions and the operator itself, that is consistent with the stability properties of the associated underlying nonlinear system. In practice, one can compute a Koopman operator for any choice of basis functions. However, the stability-related constraints on the admissible basis functions can indicate which basis functions are consistent with stable Koopman operators. I argue that using basis functions that violate these conditions and are inconsistent with stable dynamics would generate unstable Koopman operator solutions or, if stability is enforced on the operator during learning as is the focus of this work, would lead to higher training error. As I show next, this becomes problematic as bounding the training error is of central importance in the construction of Lyapunov functions, which I show next.

### 10.1.3. Lyapunov Functions Using Stable Koopman Operators

Given a stable operator, it is possible to design Lyapunov functions for nonlinear systems using the data-driven Koopman matrix. Consider an approximate finite-dimensional Koopman representation that is equivalent, based on Definition 3, to general dynamical systems (10.8) such that

$$\frac{d}{dt}\Psi(s(t)) = \tilde{\mathcal{K}}\Psi(s(t)) + \epsilon(\Psi(s(t))), \tag{10.12}$$

where

$$\epsilon(\Psi(s(t))) \triangleq f(\Psi(s(t)) - \tilde{\mathcal{K}}\Psi(s(t)) \tag{10.13}$$

is the residual error. If $\tilde{\mathcal{K}}$ in (10.12) is stable, I can design Lyapunov functions for the nonlinear dynamics, as I prove in Theorem 6.

**Theorem 6.** *Consider a Koopman representation* (10.12) *of a nonlinear dynamical system. Further assume $\tilde{\mathcal{K}}$ is stable. Let*

$$\alpha \le \lambda_{min}(Q) - 2\lambda_{max}(P),$$

*where $Q \succ 0$ and $P \succ 0$ are a solution to the Lyapunov equation*

$$\tilde{\mathcal{K}}^T P + P\tilde{\mathcal{K}} + Q = 0. \tag{10.14}$$

*If*

(10.15) $$\|\epsilon(\Psi(0))\|_2=0$$

*and*

(10.16) $$\|\epsilon(\Psi(s(t)))\|_2<\alpha\|\Psi(s(t))\|_2 \ \forall \ \Psi(s(t))\in\mathcal{D}_\alpha\subseteq\mathcal{S},$$

*then, the zero solution $\Psi(s(t))=0$ to the nonlinear dynamics is asymptotically stable in $\mathcal{D}_\alpha$. Further, $V=\Psi(s(t))^T P\Psi(s(t))$ is a Lyapunov function in $\mathcal{D}_\alpha$.*

Proof. Consider a candidate Lyapunov function $V=\Psi^T(s(t))P\Psi(s(t))$. Taking the time derivative,

$$\begin{aligned}\frac{d}{dt}V(\Psi(s(t)))&=\frac{dV(\Psi(s(t))}{d\Psi(s(t))}\frac{d\Psi(s(t))}{dt}\\&=\Psi(s(t))^T Pf(\Psi(s(t)))+f(\Psi(s(t)))^T P\Psi(s(t))\\&=\Psi(s(t))^T P[\mathcal{K}\Psi(s(t))+\epsilon(\Psi(s(t)))]+[\Psi^T\tilde{\mathcal{K}}^T+\epsilon(\Psi(s(t)))^T]P\Psi(s(t))\\&=\Psi(s(t))^T(P\tilde{\mathcal{K}}+\tilde{\mathcal{K}}^T P)\Psi(s(t))+2\Psi(s(t))^T P\epsilon(\Psi(s(t))).\end{aligned}$$

Given that $\tilde{\mathcal{K}}$ is stable and $Q\succ 0$, then there always exists a (unique) solution $P\succ 0$ to the Lyapunov equation. Thus,

$$P\tilde{\mathcal{K}}+\tilde{\mathcal{K}}^T P=-Q$$

such that

$$\frac{d}{dt}V(\mathbf{\Psi}(s(t)))=-\mathbf{\Psi}(s(t))^TQ\mathbf{\Psi}(s(t))+2\mathbf{\Psi}(s(t))^TP\epsilon(\mathbf{\Psi}(s(t))).$$

Note that $-\mathbf{\Psi}(s(t))^TQ\mathbf{\Psi}(s(t))\leq-\lambda_{min}(Q)\|\mathbf{\Psi}(s(t))\|_2^2$ and using the Cauchy-Schwartz inequality [**225**], it follows that

$$\frac{d}{dt}V(\mathbf{\Psi}(s(t)))\leq-\lambda_{min}(Q)\|\mathbf{\Psi}(s(t))\|_2^2+2\lambda_{max}(P)\|\mathbf{\Psi}(s(t))\|_2\|\epsilon(\mathbf{\Psi}(s(t)))\|_2.$$

Then, using (10.16), I can rewrite

$$\frac{d}{dt}V(\mathbf{\Psi}(s(t)))\leq-(\lambda_{min}(Q)-2\alpha\lambda_{max}(P))\|)\|\mathbf{\Psi}(s(t))\|_2^2\ \forall\ \mathbf{\Psi}(s(t))\in\mathcal{D}_\alpha\subseteq\mathcal{S}.$$

Choosing $\alpha\leq\frac{\lambda_{min}(Q)}{2\lambda_{max}(P)}$, then

$$\frac{d}{dt}V(\mathbf{\Psi}(s(t)))\leq 0\ \forall\ \mathbf{\Psi}(s(t))\in\mathcal{D}_\alpha$$

such that the system (10.12) is stable in $\mathcal{D}_\alpha$ about $\mathbf{\Psi}(s(t))=0$. Further, for $\alpha<\frac{\lambda_{min}(Q)}{2\lambda_{max}(P)}$, $\frac{d}{dt}V(\mathbf{\Psi}(s(t)))<0\in\mathcal{D}_\alpha$ and the system is asymptotically stable in $\mathcal{D}_\alpha$ about $\mathbf{\Psi}(s(t))=0$. $\square$

Theorem 6 makes it is possible to design Lyapunov functions for nonlinear systems that are valid in a region $\mathcal{D}_\alpha$ of the state space where the modeling error of the Koopman operator is bounded (10.16). This highlights the importance of choosing appropriate basis functions that satisfy the stability conditions outlined in Definition 3 and the conditions in (10.15) and (10.16). Further, Theorem 6 makes it possible to design control-Lyapunov functions that can be used to verify stabilizing feedback laws from controlled measurements in a region $\mathcal{D}_\alpha$ (10.16) for data-driven systems. In Section 10.3, I present a few examples of verifying the stability of controlled systems using Lyapunov functions constructed from stable data-driven Koopman operators, but leave more sophisticated analysis for future work.

Section 10.1 shows that stable Koopman operators can improve long-term predictions and help construct Lyapunov functions. I also present conditions on admissible basis functions for stable Koopman models to match the stability properties of the operator and improve the training error. However, in practice, data-driven Koopman operators can be unstable even if the modeled dynamics are stable and even if appropriate basis functions (as described in Definition 3) are used. For this reason, next in Section 10.2 I next present the first framework for data-driven identification of stable Koopman operators (DISKO).

## 10.2. Synthesis of Stable Koopman Operators

This section presents the methodology used for the Data-driven Identification of Stable Koopman Operators (DISKO). There are several candidate algorithms that can compute a stable solution to the optimization in (10.17). In this chapter, I use the gradient-descent algorithm presented in Chapter 9 (referred to as SOC) to find locally optimal stable solutions $\tilde{\mathcal{K}}_d$. The SOC algorithm builds upon the work in [**212**, **226**], where the nearest stable matrix to an unstable one is computed by minimizing the Frobenius norm.

I choose the SOC algorithm because it is shown to outperform the top-alternative existing algorithms for stable linear dynamical systems in terms of model accuracy, memory efficiency, and scalability such that it can be used for feedback control of higher-dimensional systems when the other methods fail.

### 10.2.1. Stable Least-Squares Koopman Operators

To compute a stable Koopman operator, I first convert the optimization (10.17) to a different formulation that is suitable for the SOC algorithm.

**Proposition 10.** *Consider $P$ measurements of states $s \in \mathbb{R}^N$ and basis functions $\Psi(s(t)) \in \mathbb{R}^W$. Given $X$ and $Y$ such that*

$$X = \begin{bmatrix} \Psi(s(t_1),u(t_1))^T \\ \vdots \\ \Psi(s(t_P),u(t_P))^T \end{bmatrix}^T \quad \textit{and} \quad Y = \begin{bmatrix} \Psi(s(t_1+\Delta t),u(t_1+\Delta t))^T \\ \vdots \\ \Psi(s(t_P+\Delta t),u(t_P+\Delta t))^T \end{bmatrix}^T .$$

*Then, the expression*

$$\sum_{k=1}^{P} \frac{1}{2} \|\Psi(s(t_k+\Delta t),u(t_k+\Delta t)) - \tilde{\mathcal{K}}_d \Psi(s(t_k),u(t_k))\|^2$$

*is equivalent to*

$$\frac{1}{2} \|Y - \tilde{\mathcal{K}}_d X\|_F^2,$$

*where $X,Y \in \mathbb{R}^{W \times P}$, $\tilde{\mathcal{K}}_d \in \mathbb{R}^{W \times W}$, $\|\cdot\|_F$ is the Frobenius norm of a matrix and $\mathbb{S}_d^{W,W}$ is the set of all stable matrices of size $W \times W$.*

Proof. See Appendix A.8. □

From Proposition 10, seeking stable Koopman operators for

$$\inf_{\tilde{\mathcal{K}}_d \in \mathbb{S}_d^{W,W}} \frac{1}{2} \|Y - \tilde{\mathcal{K}}_d X\|_F^2, \tag{10.17}$$

is equivalent to seeking stable solutions for (7.4). Note that solving (10.17) is not equivalent to projecting the unconstrained Koopman solution (7.4) to the stable set of matrices. That is,

$$\inf_{\tilde{\mathcal{K}}_d \in \mathbb{S}_d^{W,W}} \frac{1}{2}\|Y-\tilde{\mathcal{K}}_d X\|_F^2 \neq \inf_{\tilde{\mathcal{K}}_d \in \mathbb{S}_d^{W,W}} \frac{1}{2}\|\tilde{\mathcal{K}}_d^* - \tilde{\mathcal{K}}_d\|_F^2. \tag{10.18}$$

Projecting an unstable solution of (7.4) to the stable set results in a matrix that is stable but often with much greater fitness error than the solution to (10.17), as I demonstrate with examples later in Section 10.3.

The SOC algorithm uses the property that a matrix $A$ is stable if and only if it can be written as $A = S^{-1}OCS$ [**226**], where $S$ is invertible, $O$ is orthogonal, and $C$ is a positive semidefinite contraction; its singular values are less than or equal to 1. Then, I reformulate the optimization (10.17) such that

$$\inf_{K_d \in \mathbb{S}_d^{W,W}} \frac{1}{2}\|Y-\tilde{\mathcal{K}}_d X\|_F^2 = \inf_{S \succ 0, U \text{ orthogonal}, C \succeq 0, \|C\| \leq 1, B} \frac{1}{2}\|Y - S^{-1}OCSX\|_F^2.$$

**Algorithm I** Data-driven Identification of Stable Koopman Operators (DISKO)

```
Input: $\Psi_s(t), \Psi_u(t)$                                  ▷ Choice of basis functions
Output: Stable Koopman model
1: while $k < k_{max}$ do
2:     Collect new state and control measurements $s(t_k), s(t_k+\Delta t)$, $u(t_k)$
3:     Evaluate basis functions $\Psi_s(s(t_k)), \Psi_s(s(t_k+\Delta t)), \Psi_u(t_k)$
4:     Update $\mathcal{G}, \mathcal{A}, X_U, Y_U, U_U$                 ▷ Preserve memory space
5:     Run SOC algorithm                    ▷ Compute stable Koopman dynamics (7.9)
6: end while
```

When considering systems with inputs, instead of imposing stability on the Koopman operator that propagates both state- and input- dependent basis functions, I use the

Koopman dynamics shown in (7.9) and impose stability only the state transition matrix $A$. Then, the optimization problem becomes

$$[A,B]=\inf_{S\succ 0, U \text{ orthogonal}, C\succeq 0, \|C\|\leq 1, B} \frac{1}{2}\|Y-S^{-1}OCSX-BU\|_F^2, \tag{10.19}$$

where $X,Y\in\mathbb{R}^{W_s\times P}$ include the measuremets of the state-dependent Koopman basis functions $\Psi_s(s(t))$ and $U\in\mathbb{R}^{W_u\times P}$ include the measurements of the control-dependent ones $\Psi_u(u(t))$, similar to the form shown in proposition 10. Note that for linear systems with inputs, the stability properties described in Section 10.1 apply to the state-transition matrix $A$ shown in (7.9). In the rest of the analysis, one can simply set $B=0$ for systems without inputs. For details on how the SOC algorithm can solve either (10.17) or (10.19), I refer the reader to Chapter 9 and the publicly available code[3].

Let $f(S,O,C,B)=\frac{1}{2}\|Y-S^{-1}OCSX-BU\|_F^2$. The gradients with respect to $S$,$O$, and $C$ are derived in Chapter 9 and rewritten here in a more compact form as

$$\begin{aligned}
\nabla_S f(S,O,C,B) &= S^{-T}[\mathcal{V}A^T-A^T\mathcal{V}]\\
\nabla_O f(S,O,C,B) &= -S^{-T}\mathcal{V}S^TC^T\\
\nabla_C f(S,O,C,B) &= -O^TS^{-T}\mathcal{V}S^T\\
\nabla_B f(S,O,C,B) &= -(Y-AX-BU)U^T
\end{aligned} \tag{10.20}$$

where $A=S^{-1}OCS\in\mathbb{R}^{W_s\times W_s}$ and $\mathcal{V}=(Y-AX-BU)X^T\in\mathbb{R}^{W_s\times W_s}$. The gradients (10.20) depend on $X$, $Y$, and $U$, which contain a history of all the basis functions measurements and can slow down the computation over time, as increasingly more data are collected.

[3]https://github.com/MurpheyLab/MemoryEfficientStableLDS

To speed up the computation as well as preserve memory space, I use the relationships $XX^T{=}\mathcal{G}{\in}\mathbb{R}^{W_s\times W_s}$, $YX^T{=}\mathcal{A}{\in}\mathbb{R}^{W_s\times W_s}$, $XU^T{=}X_U{\in}\mathbb{R}^{W_s\times W_u}$, $YU^T{=}Y_U{\in}\mathbb{R}^{W_s\times W_u}$, and $UU^T{=}U_U{\in}\mathbb{R}^{W_u\times W_u}$ (derived in Appendix A.9) such that $\mathcal{V}{=}\mathcal{A}{-}A\mathcal{G}{-}BX_U^T$ and $\nabla_B f(S{,}O{,}C{,}B){=}{-}Y_U{+}AX_U{+}BU_U$. Then, the gradient directions can be incrementally updated with new measurements and preserve memory space. The algorithmic steps of the DISKO framework are presented in Algorithm I.

## 10.3. Results

In this section, I demonstrate the benefits of DISKO in prediction and control. First, I consider systems without inputs and show the effect of imposing stability on the prediction accuracy. I first compare the solutions of the proposed SOC algorithm to the method [**212**] that does not consider the least-squares fitness error when projecting the matrix to the stable set of solutions. Then, I compare the evolution of nonlinear dynamics using the unconstrained and stable Koopman models, shown in (7.4) and (10.17), respectively.

### 10.3.1. Least-Squares vs Nearest Stabilization

I demonstrate the difference between projecting an unstable matrix to the nearest stable solution (nearest stabilization) and solving for a stable matrix while also optimizing for the reconstruction error (least-squares stabilization).

Consider the randomly generated matrices

$$X = \begin{bmatrix} 0.1419 & 0.4218 & 0.9157 & 0.7922 & 0.9595 \\ 0.6557 & 0.0357 & 0.8491 & 0.9340 & 0.6787 \\ 0.7577 & 0.7431 & 0.3922 & 0.6555 & 0.1712 \end{bmatrix}$$

$$Y = \begin{bmatrix} 8.1472 & 9.0579 & 1.2699 & 9.1338 & 6.3236 \\ 0.9754 & 2.7850 & 5.4688 & 9.5751 & 9.6489 \\ 1.5761 & 9.7059 & 9.5717 & 4.8538 & 8.0028 \end{bmatrix}.$$

Computing the least-squares solution using (7.4) and then projecting it to the stable set of matrices using [**212**] yields

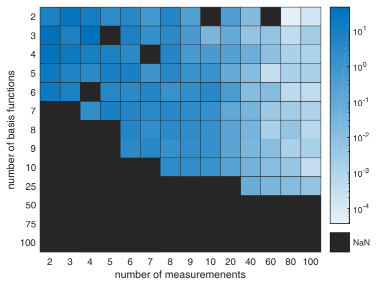
$$\tilde{\mathcal{K}}_d = \begin{bmatrix} 0.0041 & -6.6031 & 5.1709 \\ 10.3449 & -1.9480 & -0.0590 \\ 11.7192 & -6.7149 & 3.4609 \end{bmatrix},$$

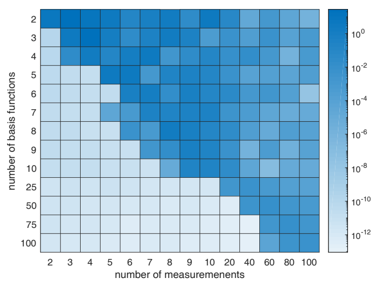
with eigenvalues $\Lambda = \{0.87, 0.87, -0.22\}$. The least-squares error using the stable matrix is

$$\frac{1}{2}\|Y - \tilde{\mathcal{K}}_d X\|_F^2 = 203.04$$

and the Frobenius norm from the least squares solution is

$$\frac{1}{2}\|\tilde{\mathcal{K}}_d^* - \tilde{\mathcal{K}}_d\|_F^2 = 45.98.$$


(a) CG


(b) DISKO

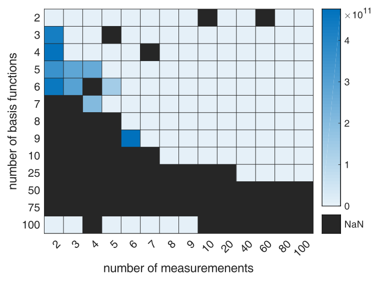

(c) Difference

Figure 10.2. Comparison of the SOC (Algorithm I) and CG [**4**] algorithms as a function of the total number of random measurements used for training and the total number of basis functions. The error is normalized by the product of the number of measurements and functions. In 10.2c, the difference is calculated as the percent difference of the error between the two algorithms and is calculated as $\frac{e_{CG}-e_{SOC}}{e_{SOC}}$.

On the other hand, directly solving (10.17) generates a different solution

$$\tilde{\mathcal{K}}_d=\begin{bmatrix} 5.6337 & -8.2334 & 11.5883 \\ 14.4877 & -5.0863 & 1.9636 \\ 8.3346 & -2.8916 & 1.0662 \end{bmatrix}$$

with eigenvalues $\Lambda=\{0.98,0.98,-0.35\}$. The least-squares error using the stable matrix is

$$\frac{1}{2}\|Y-\tilde{\mathcal{K}}_dX\|_F^2=79.47$$

and the Frobenius norm from the least squares solution is

$$\frac{1}{2}\|\tilde{\mathcal{K}}_d^*-\tilde{\mathcal{K}}_d\|_F^2=108.53.$$

As expected, projecting the unstable solution to the stable set and ignoring the least-squares error fitness generates a solution that is closer, in the Frobenius norm sense, to the original unstable matrix, but also with greater error compared to the solution of (10.17).

### 10.3.2. Comparison to alternative schemes for DISKO

As mentioned earlier, there are many candidate algorithms that can be implemented for DISKO. To motivate using the SOC algorithm over alternative choices, I compare it to the constraint generation (CG) approach [**4**], which has been shown to outperform competing alternative algorithms.[4] The two algorithms are compared also in Chapter 9, but not on randomly generated matrices and not up to such high dimensions.

I compare SOC and CG on finding stable operators that minimize the error in (10.17) for varying number of basis functions ($W \in [2{,}100]$ and number of measurements ($P \in [2{,}100]$). Data in $X$ and $Y$ are sampled from the uniform distributions $U(0{,}10)$ and $U(0{,}20)$, respectively, where $U(a{,}b)$ is a uniform distribution and $a$ and $b$ are the minimum and maximum values. The results are presented in Fig. 10.2. SOC outperforms CG in all cases, for any number of measurements and functions used. Further, CG does not always converge to a solution in the allotted time (10 minutes per minimization). Besides the superior performance in terms of the reconstruction error, the SOC algorithm is also more memory efficient. In light of these results, I use the SOC algorithm to implement DISKO in the remaining of this work.

### 10.3.3. Comparisons of Reconstruction and Prediction Error

Next, I compare the evolution of the nonlinear dynamics of a pendulum (without control) using the unconstrained solution (7.4) and the stable one (10.17). The states, dynamics,

[4]The algorithm in [**107**] does not work well for systems with inputs, as demonstrated in Chapter 9.

and basis functions used to train the Koopman operator are given by

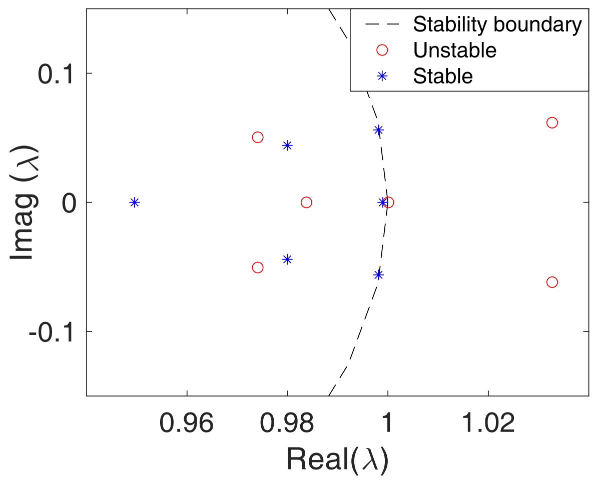
$$s=[\theta,\dot{\theta}]^T, \qquad \frac{d}{dt}s=[\dot{\theta},9.81\sin(\theta)+\beta\dot{\theta}]^T$$
$$\Psi(s)=[\theta,\dot{\theta},\sin(\theta),\cos(\theta)\dot{\theta},\sin(\theta)\cos(\theta),\sin(\theta)\dot{\theta}^2]^T,$$

where $\theta$,$\dot{\theta}$, and $\beta$ are the angle, angular velocity and damping coefficient, respectively. The time spacing between samples is $\Delta t$=0.02 s. For simplicity, the time dependencies of the variables are dropped.

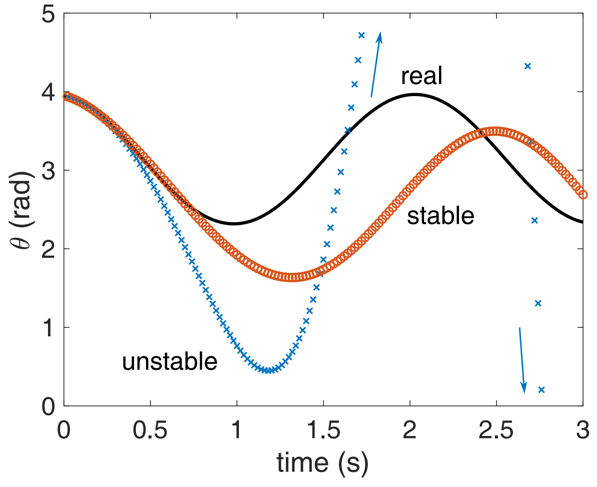
Fig. 10.3 shows the eigenvalues of the two methods and the prediction of the two solutions for the undamped pendulum ($\beta$=0). The prediction associated with the unconstrained least-squares Koopman operator diverges away, whereas the states predicted with DISKO remain bounded and close to the true evolution of the nonlinear dynamics for a longer amount of time.

Next, I investigate the effect of unstable eigenvalues on the prediction accuracy, as well as the data-efficiency of the algorithms. Specifically, for 300 randomly sampled initial conditions, I compare the prediction error for the pendulum angle as a function of the number of training measurements used to compute a Koopman model. I show the results in Fig. 10.4 for an undamped and a damped pendulum system.

For both systems, the stable Koopman operator leads to smaller average prediction error for any number of measurements used for training, as well as lower error variance than the least-squares, unconstrained solution (7.4), exhibiting better predictive accuracy and robustness. Further, the large error spikes associated with the unstable Koopman model around 5 measurements indicate that the unstable solution can be very inaccurate when trained with few measurements; on the other hand, the predictive accuracy of the



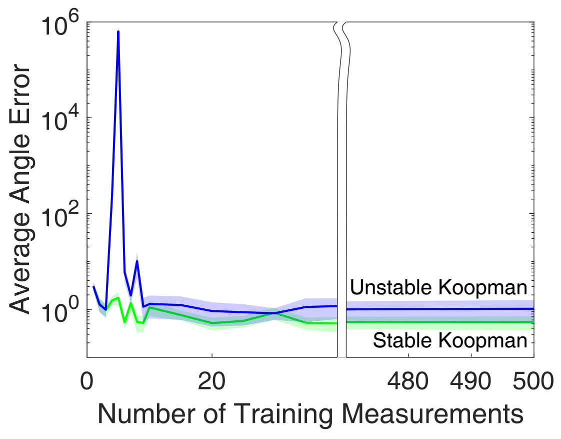
(a)



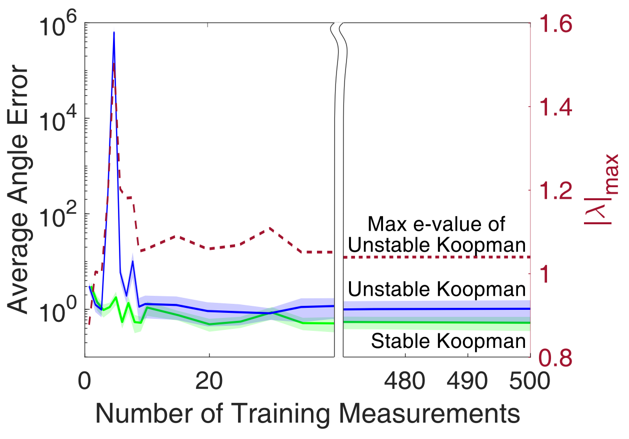
(b)

Figure 10.3. Figure 10.3a shows the eigenvalues of the unconstrained (7.4) and constrained (10.17) Koopman operator for the nonlinear dynamics of a pendulum. The constrained operator pushes the unstable eigenvalues to the stability boundary. The stable eigenvalues are also appropriately modified so that constrained Koopman solution locally minimizes the prediction error (10.17). Figure 10.3b shows the prediction of the angle of the pendulum system using the unconstrained Koopman solution (7.4) and the constrained-stable Koopman operator. The predictions of the unconstrained Koopman operator start to diverge away from the pendulum states after 1 second.

stable Koopman operator remains almost the same regardless of the size of the training sample, demonstrating data-efficiency and emphasizing the benefit of DISKO in the low-sample limit. When few measurements are available, the least-squares solution is prone to misidentifying the system, but the stability constraints help make the learning process less sensitive to the amount of training data. The envelope of the standard deviation error of the unconstrained Koopman is slightly lower than that of the stable solution around 10 training measurements, suggesting that it is possible that the unstable Koopman generates at times a smaller error and is more accurate. I argue that this is a result of the unstable Koopman overfitting to certain initial conditions and accurately predicting the evolution



(a) Undamped pendulum



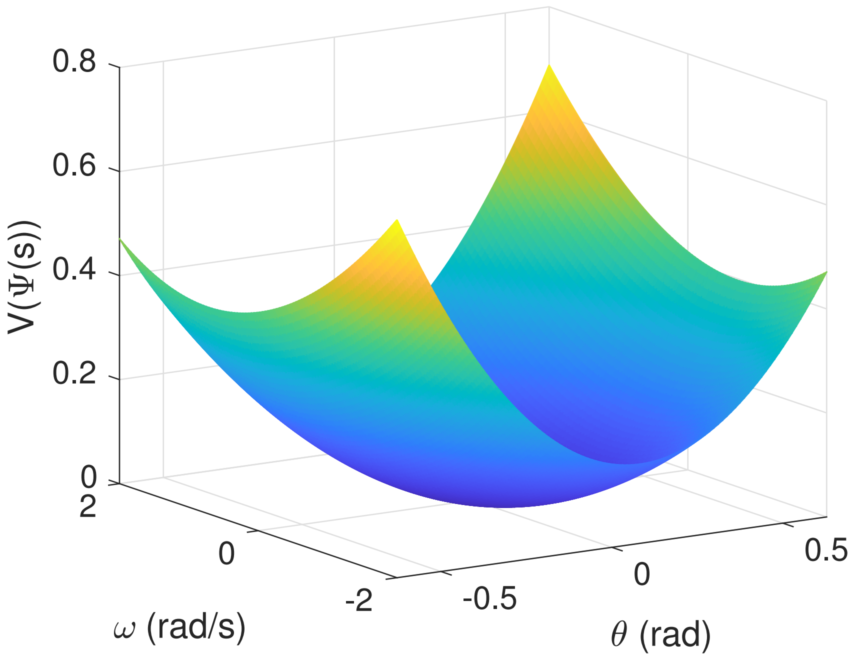
(b) Damped pendulum

Figure 10.4. Average angle error, with one-half standard deviation shading, for the undamped (Fig. 10.4a) and damped (Fig. 10.4b) pendulum dynamics, as predicted by the unconstrained and the constrained-stable Koopman operator solutions. For each number of measurements used to compute a Koopman operator, the average angle error is the average absolute difference of the true system state (evolved using the nonlinear dynamics) and the system state as predicted by either Koopman operator over 1 second over 300 initial conditions.

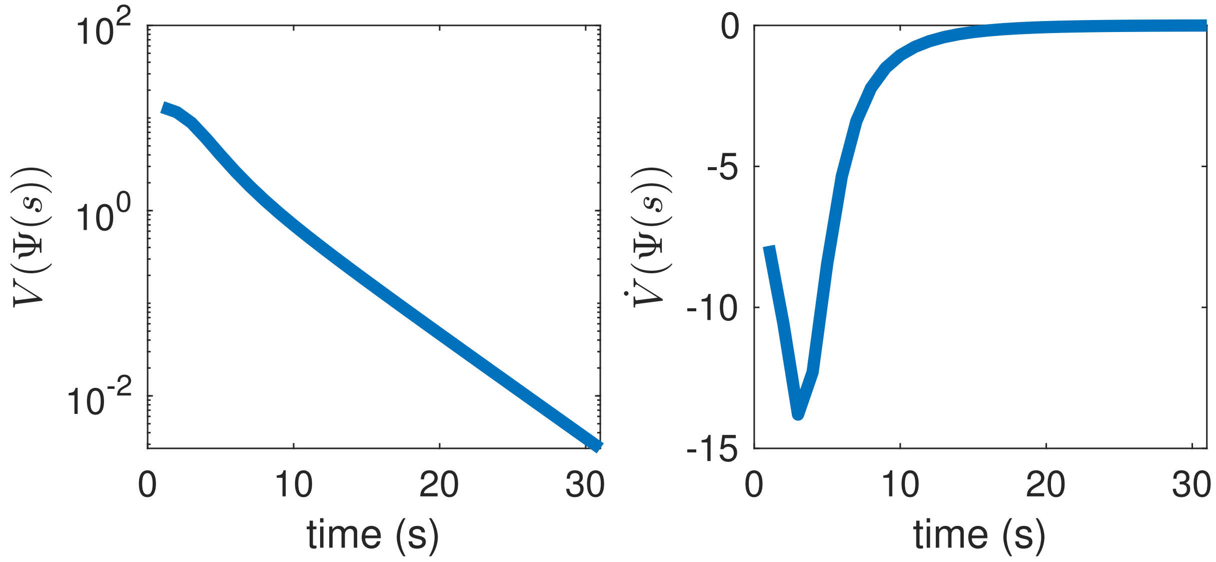
of very few initial states. Last, the unconstrained least-squares solution for the damped pendulum is always unstable, misidentifying in all cases the true properties of the system.

Next, I use the undamped pendulum system to illustrate how stable Koopman operators can be used to construct Lyapunov functions and verify the stability of a controller. Using LQR feedback, I generate a trajectory and use the state measurements and the DISKO algorithm to compute a stable Koopman operator. I then use the Koopman operator to construct a Lyapunov function, which I evaluate with the measurements from the controlled trajectory.

To generate LQR control, I use $Q$=diag[1,1] and $R$=diag[0.01], initial conditions $[\theta_0,\dot{\theta}_0]=[\pi,5]$ and collect measurements every $\Delta t$=0.1 s. I train a Koopman operator



(a) Candidate control-Lyapunov function



(b) Evaluation with state measurements from LQR-controlled trajectory.

Figure 10.5. Candidate control-Lyapunov function constructed from stable Koopman operators and evaluated on a LQR-controlled pendulum. The candidate control-Lyapunov function is used to verify the stability of the controlled trajectory.

using $\Psi(s)=[\theta,\dot{\theta},\theta^2,\dot{\theta}^2,\sin(\theta),\sin(\dot{\theta}),\sin(\theta)\dot{\theta},\sin(\dot{\theta})\theta]^T$, which satisfy the stability conditions presented in Section 10.1. I then solve the Lyapunov equation (10.14) using $Q_{\tilde{\mathcal{K}}}=$ diag[1,1,0,0,0,0,0,0] and construct the candidate control-Lyapunov function as $V(\Psi((s))= \Psi(s)^T P\Psi(s)$. The constructed Lyapunov function, rounded to two decimal places, is $V(\Psi(s))=1.16\theta^2+0.04\omega^2-0.16\theta\sin(\theta)+0.02\theta\sin(\omega)+0.04\omega\sin(\theta)+0.06\sin(\theta)^2+0.04\theta\omega$. I show the results in Fig. 10.5. The candidate control-Lyapunov function evaluated with the controlled trajectory of the pendulum satisfies the properties of a Lyapunov function and shows that the specific trajectory is converging to the equilibrium.

Note that the control-Lyapunov function shown in Fig. 10.5 is used to verify the stability of the applied controller for the particular trajectory and may not be a Lyapunov function everywhere in the state space. Rather, one can compute the region of validity based on Theorem 4 and the modeling errors of the data-driven Koopman operator [**227**, **228**]. I leave further analysis of Lyapunov functions generated with stable Koopman operators to future work.

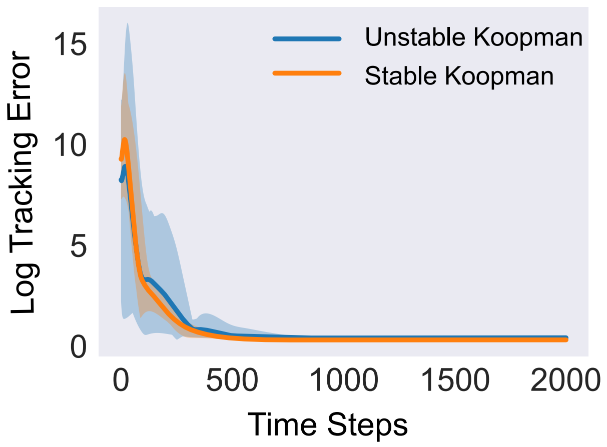
### 10.3.4. Nonlinear Control Using Stable Koopman Operators

In this subsection, I demonstrate the benefit of using stable Koopman operators for nonlinear control. By improving the robustness and modeling accuracy of data-driven models, I argue that stability-constrained models would also lead to improved control performance.

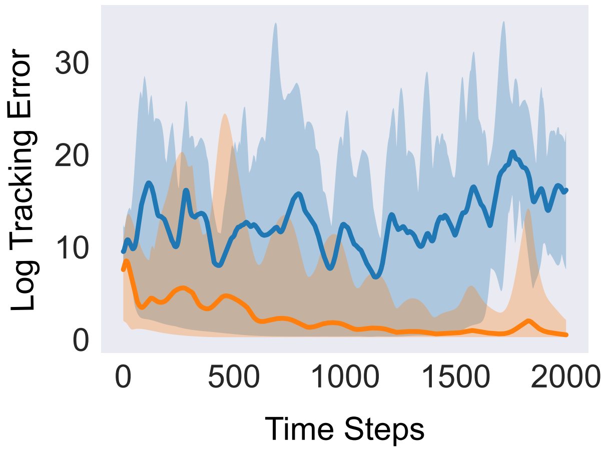
**10.3.4.1. Quadrotor.** I first consider stabilizing a falling quadrotor. Using active learning, which has been shown to enhance learning and the accuracy of identified dynamics [**229**], I collect training data within the first second of the free-fall. Then, using the same training sample, I compute a Koopman model using the least-squares solution (7.4) and DISKO and develop an LQR policy to stabilize the quadrotor. The system feedback rate is 200 Hz and the state is partially observed containing only the measured body-relative gravity vector and the body linear and angular velocities (see [**229**] for more detail). The quadrotor dynamics, LQR parameters, and Koopman basis functions are the same as in [**229**].

I present the stabilizing performance of the stable and unstable Koopman models in Fig. 10.6. I consider two prediction horizons (30 and 40 time steps) used in computing the finite-horizon LQR control and test both approaches using the same 10 uniformly sampled initial conditions as done in [**229**]. I use the median score (in log scale) as a performance metric of the two approaches, because it is not as biased in a case of failure (when states diverge). Using DISKO, the control is robust and stabilizes the dynamics for both choices of the prediction horizon, contrary to the unconstrained model that fails when using a longer prediction window. Even in the 30 time-step horizon, however, where both models succeed, the stable model leads to a lower median error than the unconstrained model and has smaller variance. These results show that active learning techniques that enhance learning are not sufficient to address the challenges of unconstrained data-driven models and can be further improved through the use of DISKO.

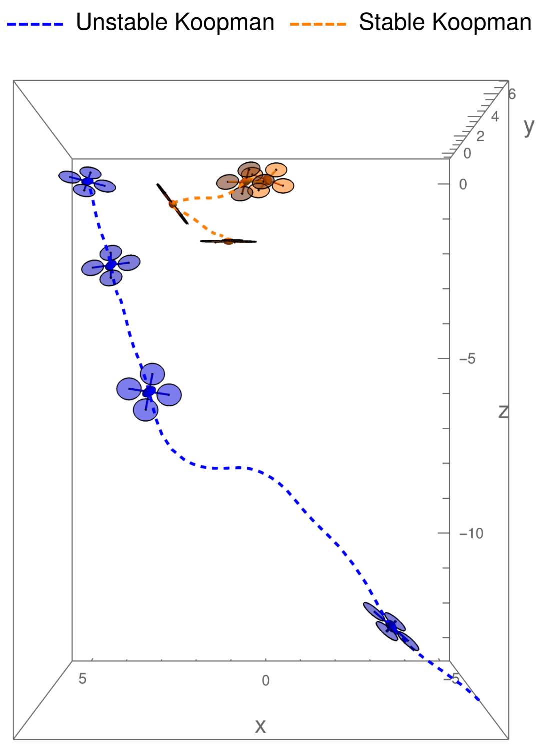
**10.3.4.2. Pusher-slider system.** Next, I demonstrate the benefits of DISKO using a pusher-slider system [**230**–**232**]. The pusher is a steel rod held tightly by the end effector of the Franka Emika Panda robot [**106**] and the slider is a rectangular block with dimensions 15.3×13 cm. The states of the system are recorded at 10 Hz using an overhead camera and QR codes on the slider. The experimental setup is shown in Fig. 10.7.



(a) 30 time-steps horizon.



(b) 40 time-steps horizon.



(c) LQR-controlled trajectories.

Figure 10.6. Performance of LQR control derived from the stable (10.17) and unstable (7.4) Koopman operators for the quadrotor dynamics. Both models use the same training measurements that are collected with active learning. At the end of the learning phase, the stable Koopman is computed and the LQR gains from both models are derived. Figures 10.6a and 10.6b show the log error of the tracking cost for 10 trajectories with the same uniformly sampled initial conditions. The solid line represents the median score of each approach and the shaded envelope the lowest and highest cost. Figure 10.6c shows a trajectory using the 40 time-step horizon control. The initial conditions are the same, but shifted in the x-axis for better visibility.

To train a model, I collected data using a controller to push the block with the end-effector of the robot. I collected six training sets (200 measurements each) and used the data to compute a least-squares unconstrained and a stable Koopman model. The basis functions used are

$$\Psi(s)=[x,y,\theta,py,v_n,v_p,\sin(\theta)v_n,\cos(\theta)v_n,\sin(\theta)v_p,\cos(\theta)v_p,pyv_n,v_p,v_n,u]^T,$$

where $x$,$y$, and $\theta$ are the world-frame coordinates and orientation of the block, $py$ is the distance of the slider (end effector of the Franka Emika robot) away from center of the block and along its pushing side (the slider is assumed to be always in contact with the block), $v_n$ and $v_p$ are the normal and parallel velocity of the slider in the body-frame of the block, respectively, and $u\in\mathbb{R}^2$ is the acceleration input for the body-frame normal and parallel velocity of the slider. To compare the unconstrained and stable least-squares models, I forward predict the system with zero inputs (see Fig. 10.8). Given no movement from the pusher, the block should stay in place. However, the states propagated with the unconstrained model diverge, as expected for an unstable linear model. On the other hand, the simulated prediction of the DISKO model barely shows any motion and is consistent with the expected behavior of the system.

Next, I test the predictive accuracy of the learned DISKO representation against the training data. Specifically, I use the stable Koopman model to forward-simulate the pusher-slider system with the controls applied during the experiments and compare it to the actual trajectories. The results are shown in Fig. 10.9. In all cases, the model obtained using DISKO generates qualitatively similar trajectories compared to the actual experiments.

Last, I test the control performance of the DISKO approach. I apply infinite-horizon LQR control, the gains of which are calculated offline once. The results are shown in Fig.

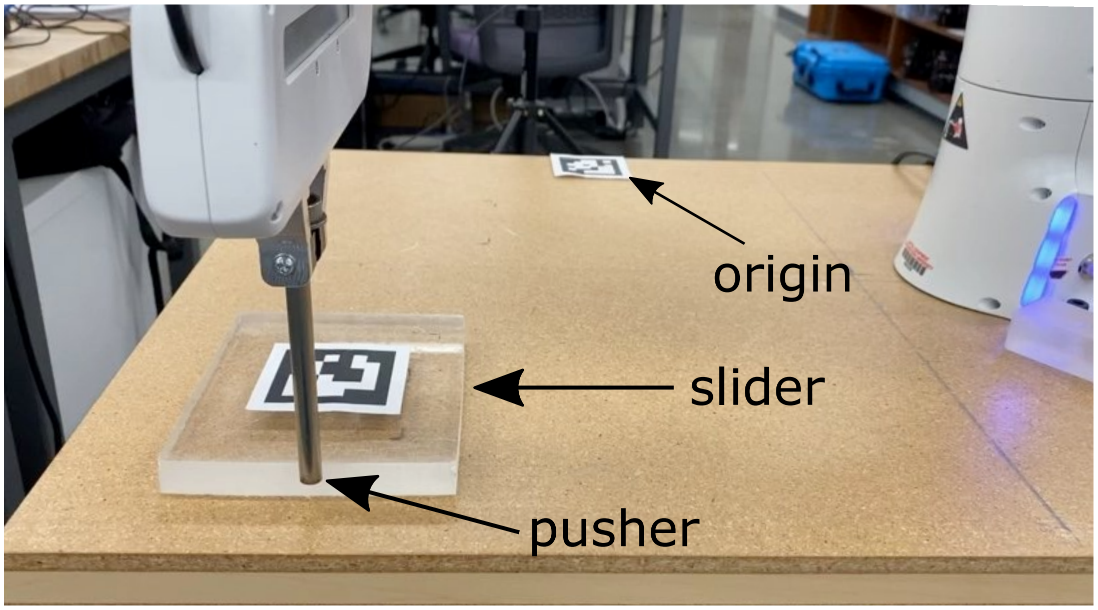


Figure 10.7. Experimental setup of the pusher-slider system. The states of the pusher and the slider are recorded with an overhead camera. The block configuration is identified using QR labels.

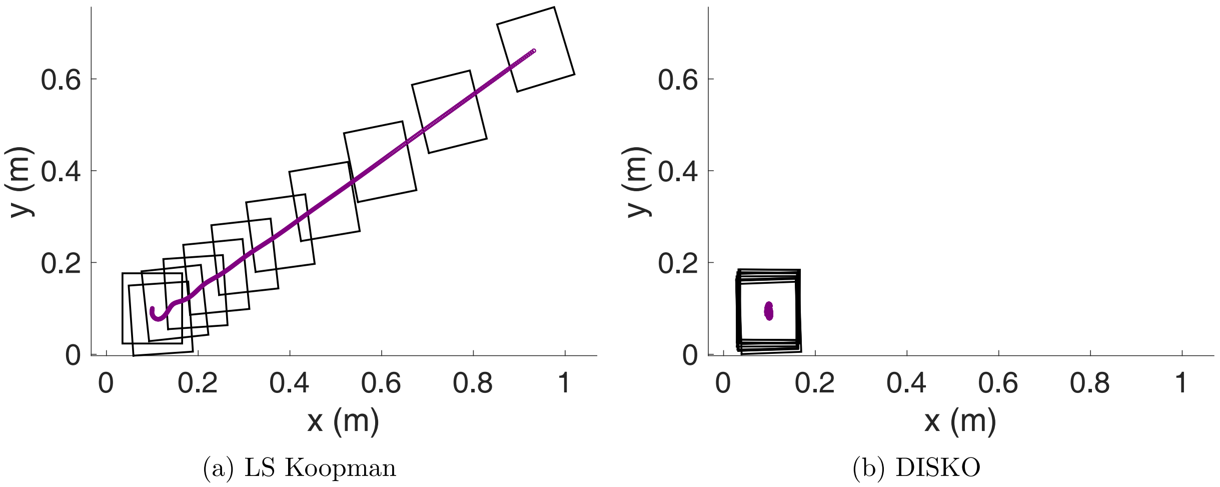


Figure 10.8. Simulation of the pusher-slider system over 500 time steps ($dt$=0.1) with zero control inputs. The least-squares Koopman model is unstable and drifts away, despite the fact that there should be no motion in the absence of control.

10.10. The Franka Emika Robot successfully pushes the block to the desired orientations.

In the last of the three experiments, I applied a low-pass filter to increase the controller

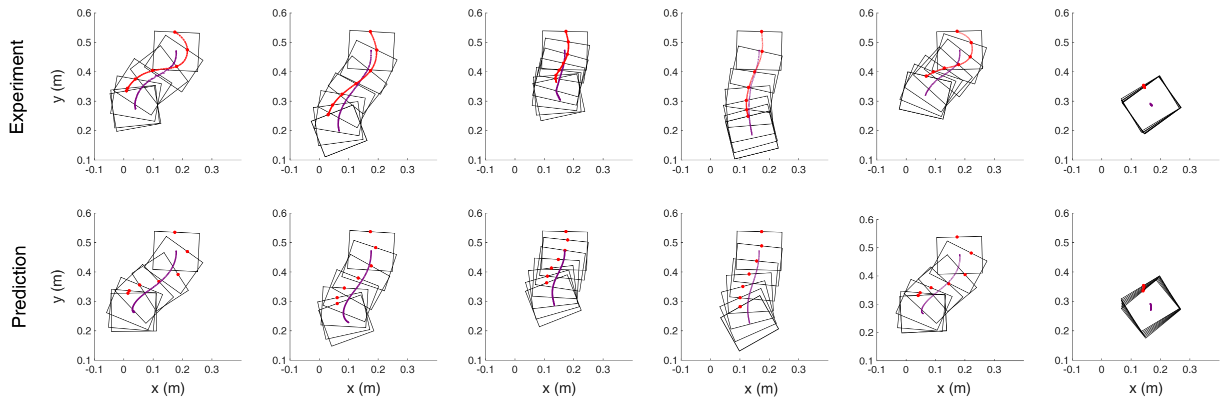


Figure 10.9. Comparison of the experimental trajectories from the training data to those obtained in simulation using DISKO. Instances are shown every 2 seconds. The simulated trajectories are initiated with the starting states of the experiments and forward-propagated with the same control inputs that were applied in each run. The position of the pusher is indicated with red and the center of the slider with purple. In the predictions, the location of the pusher is known only as $py$, the distance away from the center of the block along its pushing side; the pusher is always assumed to be in contact with the block. To highlight this difference, I do not plot a trajectory of the pusher, but show its location only at the instants the block is drawn.

lag time and slow down the experiment in order to capture more of the dynamic response of the robot experiment. The LQR objective is given in terms of $x$,$y$,$\theta$, and $py$. The weights used for the states are respectively given by $Q$=diag[800,800,50,10] and the control weights are given by $R$=diag[$10^4$,$10^4$]. In all three experiments, the pusher successfully moves the block close to the desired configuration. I verify that the controlled system converges to the target by constructing a candidate control-Lyapunov function with the stable Koopman operator and evaluating it with the state measurements. Note that the candidate control-Lyapunov function is reconstructed using Theorem 4, using data from each individual trajectory. This is a candidate control-Lyapunov function for a basis of attraction surrounding the equilibrium, but the robot would need to collect data verifying

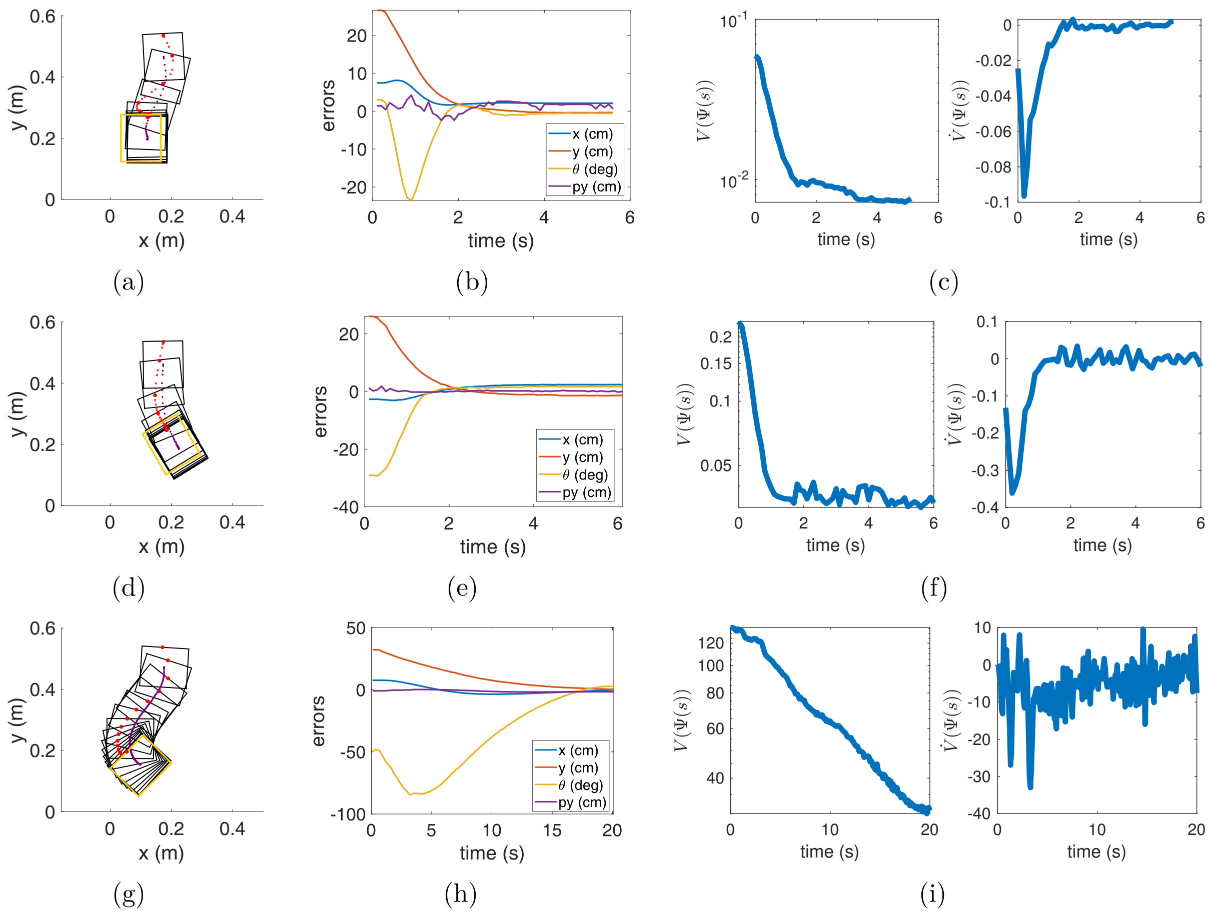


Figure 10.10. Control of the pusher-slider system using DISKO. Infinite-horizon LQR control is used to push the block to the desired orientation, marked with yellow border. The pusher and the center of the block are marked with red and purple dots, respectively. Each row corresponds to one experimental run and shows the trajectory, the tracking errors, and the constructed candidate control-Lyapunov function that verifies that the controlled system converges to the target. The candidate control-Lyapunov function is $V(\Psi(s))=\Psi(s)^T P\Psi(s)$, where $P$ is the solution to the Lyapunov equation (10.14) using the stable Koopman operator.

the candidate control-Lyapunov function. This active phase of stability verification will be considered in future work.

## 10.4. Discussion

This chapter demonstrates the benefits of stable Koopman operators in the prediction and control of data-driven nonlinear systems. I derive a formula for the prediction error associated with Koopman operators for an arbitrary number of time steps, which I use to show how unstable eigenvalues exponentially amplify modeling errors. I also derive properties for basis functions that are consistent with a stable Koopman operator and can improve the learning process. Appropriate basis functions can lead to low training error, which is in turn useful for the construction of Lyapunov functions, also shown in this work.

Using the examples of the pendulum and the quadrotor, I demonstrate how stability constraints on the Koopman models makes them more robust to limited training data (which often arises in time-urgent tasks such as stabilizing unknown dynamics), as well as the prediction horizon. In fact, stability constraints improve even models that are computed with rich data obtained through active learning methods. Last, using the pusher-slider system, I validate the performance of DISKO experimentally.

CHAPTER 11

# Conclusions and Future Work

In this thesis, I presented algorithms that improve both data-driven and model-based control. In Part 1, I presented the only nonlinear control scheme available that has formal performance guarantees that are dependent on the controllability properties of the system. Besides being the only nonlinear controller with formal convergence guarantees, it also has a computational advantage over alternative control methods, such as model predictive control, since it avoids the expensive calculation of controls over the entire horizon. In addition, it is shown to have superior convergence rates to top competing alternatives. These traits render the proposed algorithm a promising feedback scheme for underactuated and nonlinearly controllable dynamics.

In Part 2 of the thesis, I shifted the attention to control of dynamics whose model is not available. By leveraging Koopman operator theory, I presented a derivative-based methodology suitable for real-time data-driven identification and control. Besides the results presented in this thesis, the presented derivative-based methodology has also been successfully tested on predicting the state evolution of a soft robotic swimmer based on high-fidelity CFD simulations [**233**]. As shown in Chapter 8, combined with model predictive control, the performance of the proposed Koopman approach is comparable to state-of-the-art data-driven modeling methods, such as SINDy and NARX. Further, this modeling approach can even be used with known nonlinear dynamics to generate a linear representation and reap the benefits of linear tools for nonlinear feedback.

In addition, I demonstrated the benefits of using side information, and specifically stability, to constrain system identification methods to generate models with certain desirable properties. Imposing such properties on the derived models helps identify representations that better generalize beyond the training set and are remarkably more robust to the training size. Stability, in particular, is important for numerical reasons as well, as it ensures that the predictions generated by data-driven models are bounded and can be practically useful for control purposes. To impose stability on learned dynamics, I derived an optimization algorithm (SOC) to learn stable data-driven linear dynamical systems. SOC outperforms top-performing alternatives by orders of magnitude in terms of performance, which naturally leads to improved control as well. Further, SOC is significantly more memory efficient and scales better for high-dimensional systems often encountered in data-driven applications in robotics and other fields. After analyzing the prediction error based on Koopman models and explicitly showing the effect of stability, I used the SOC algorithm to present the benefits of data-driven identification of stable Koopman operators (DISKO). Stability constraints improve even models that are computed with rich data obtained through active learning methods.Using the pusher-slider system, I validated the performance of DISKO experimentally. I also derived properties for basis functions that are consistent with a stable Koopman operator and can improve the learning process. Appropriate basis functions can lead to low training error, which is in turn useful for the construction of candidate control Lyapunov functions, as illustrated in Chapter 10.

Advancements in robotics are happening at a fast pace, yet there are still a lot of challenges to be tackled in the near future. For applications where the dynamics model is available and accurate, it is important to develop feedback schemes that offer convergence guarantees for any controllable systems. Extending the second-order needle-variation-based algorithm presented in Part 1 of this thesis with higher-order information to always provide solutions for all controllable systems is a promising research area, if such an extension can scale well for high-dimensional robotics. In addition, with the increasing advancements in Koopman operator theory and linear embeddings of nonlinear systems, it is important to better understand the trade-offs between different modeling choices. Under what conditions and choices can linear embeddings of known nonlinear dynamics offer advantages in prediction and control? For example, using a linear approximation of nonlinear dynamics and modeling the error from the true model as state-dependent disturbance, as is shown in Chapter 10, can offer analysis tools of linear systems at the disposal of nonlinear control.

For applications with unknown dynamics, there is significant progress to be made in data collection, data-driven modeling, as well as data-driven control. For example, improving the data collection can help gather training data that are more representative and informative and enable better identification results. Promising active learning directions include collecting data that reduce the correlation of the existing model classifiers until system identification can be confidently attempted. Similarly, exploiting the connection between feedback linearizable and controllable dynamics and using control to cancel nonlinear terms that appear in the data-driven model until a linear model can be obtained is a promising direction for actively learning with guarantees for convergence to the true dynamics.

In addition, despite all the advancements seen in data-driven modeling in recent years, system identification has typically taken the form of black-box machine learning operations that generate unconstrained models that fit the data well, but do not generalize well outside the training set. Using side information and knowledge, general (e.g., a system is controllable) or more specific (e.g., a system has equilibria at known locations), about a system can help address the challenges of overfitting, as well as improve data-efficiency. In the future, there is a lot of room for growth in constraining optimization processes to generate models that satisfy a wider range of properties (e.g. symmetry, stability, periodicity, controllability) that transfer over to unseen data. Besides the task of the learning optimization itself, side information should be used for case-specific optimization of model parameters and classifiers a model is equipped with. For example, as Chapter 10 demonstrated, side-information such as knowledge of equilibria points of dynamics can be used to filter out basis functions that would violate the physical properties of the modeled system. In the same manner, system identification tools, such as SINDy [**198**], should use libraries with functions that are in accordance with the properties of the modeled system (e.g. basis functions that do not blow up at equilibria points). Other promising research directions include characterizing the fitness of basis functions based on other criteria, such as a Lipschitz constant for the lifted dynamics, given information about the range of system states (the domain of basis functions).

Last, promising directions for data-driven control include leveraging the training performance of system identification to generate control with performance guarantees. Treating the modeling error as state-dependent disturbance and implementing robust control on the identified model can be used to guarantee control safety. Further, improving the worst-case performance for a bundle of representations that lie within the uncertainty of the data-driven model can provide guarantees for the performance of data-driven control [**234**].

All in all, there are exciting research directions in robotics, control, and machine learning. Given the fast-evolving nature of these fields, employing on any system (known or unknown) autonomous methods that can run online, use limited computational resources, adapt with minimal data, and provide performance and safety guarantees seems like an attainable goal that can have a huge impact on applications such as self-driving cars, human-robot interactions, and underwater operations.

# Bibliography

[1] C. M. Postlethwaite, T. M. Psemeneki, J. Selimkhanov, M. Silber, and M. A. MacIver, "Optimal movement in the prey strikes of weakly electric fish: a case study of the interplay of body plan and movement capability," *Journal of The Royal Society Interface*, vol. 6, no. 34, pp. 417–433, 2009.

[2] A. R. Ansari and T. D. Murphey, "Sequential action control: closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1196–1214, 2016.

[3] E. Tzorakoleftherakis and T. Murphey, "Iterative sequential action control for stable, model-based control of nonlinear systems," *arXiv preprint arXiv:1706.08932*, 2017.

[4] B. Boots, G. J. Gordon, and S. M. Siddiqi, "A constraint generation approach to learning stable linear dynamical systems," in *Advances in neural information processing systems*, 2008, pp. 1329–1336.

[5] D. Mayne, *Nonlinear Model Predictive Control: Challenges and Opportunities*, F. Allgöwer and A. Zheng, Eds. Basel: Birkhäuser Basel, 2000, vol. 26.

[6] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[7] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[8] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming—The explicit solution," *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, 2002.

[9] F. Allgöwer and A. Zheng, *Nonlinear Model Predictive Control*. Basel: Birkhäuser, Basel, 2012, vol. 26.

[10] A. Isidori, *Nonlinear Control Systems*, 3rd ed. London, UK: Springer, 1995.

[11] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming.* New York: American Elsevier, 1970.

[12] E. Theodorou, Y. Tassa, and E. Todorov, "Stochastic differential dynamic programming," in *Proceedings of the American Control Conference*, 2010, pp. 1125–1132.

[13] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the American Control Conference*, 2005, pp. 300–306.

[14] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

[15] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation.* CRC press, 1994, ch. 18.

[16] P. V. Kokotovic, "The joy of feedback: nonlinear and adaptive," *IEEE Control systems*, vol. 12, no. 3, pp. 7–17, 1992.

[17] D. Seto and J. Baillieul, "Control problems in super-articulated mechanical systems," *IEEE Transactions on Automatic Control*, vol. 39, no. 12, pp. 2442–2453, 1994.

[18] J. L. Junkins and R. C. Thompson, "An asymptotic perturbation method for nonlinear optimal control problems," *Journal of Guidance, Control, and Dynamics*, vol. 9, no. 4, pp. 391–396, 1986.

[19] W. Perruquetti and J.-P. Barbot, *Sliding Mode Control in Engineering.* New York: Marcel Dekker, 2002.

[20] V. I. Utkin, *Sliding Modes in Control and Optimization.* Springer-Verlag, 1992.

[21] R. Xu and Ü. Özgüner, "Sliding mode control of a class of underactuated systems," *Automatica*, vol. 44, no. 1, pp. 233–241, 2008.

[22] S. C. Brown and K. M. Passino, "Intelligent control for an acrobot," *Journal of Intelligent & Robotic Systems*, vol. 18, no. 3, pp. 209–248, 1997.

[23] C. J. Harris, C. G. Moore, and M. Brown, *Intelligent Control: Aspects of Fuzzy Logic and Neural Nets.* Singapore: World Scientific, 1993, vol. 6.

[24] R. Fierro, F. L. Lewis, and A. Lowe, “Hybrid control for a class of underactuated mechanical systems,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 6, pp. 649–654, 1999.

[25] H. K. Khalil, *Nonlinear Systems.* Prentice-Hall, 2002.

[26] A. M. El-Nagar, M. El-Bardini, and N. M. EL-Rabaie, “Intelligent control for nonlinear inverted pendulum based on interval type-2 fuzzy PD controller,” *Alexandria Engineering Journal*, vol. 53, no. 1, pp. 23–32, 2014.

[27] I. Kolmanovsky and N. H. McClamroch, “Developments in nonholonomic control problems,” *IEEE Control Systems*, vol. 15, no. 6, pp. 20–36, 1995.

[28] H. J. Sussmann, “Two new methods for motion planning for controllable systems without drift,” in *European Control Conference (ECC)*, 1991, pp. 1501–1506.

[29] G. Lafferriere and H. J. Sussmann, “A differential geometric approach to motion planning,” in *Nonholonomic Motion Planning.* Springer, 1993, pp. 235–270.

[30] G. Lafferriere and H. Sussmann, “Motion planning for controllable systems without drift,” in *Robotics and Automation.* IEEE, 1991, pp. 1148–1153.

[31] R. S. Strichartz, “The Campbell-Baker-Hausdorff-Dynkin formula and solutions of differential equations,” *Journal of Functional Analysis*, vol. 72, no. 2, pp. 320–345, 1987.

[32] W. Rossmann, *Lie groups: An Introduction Through Linear Groups.* Oxford University Press, 2002, vol. 5.

[33] S. M. La Valle, “Motion planning,” *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 108–118, 2011.

[34] M. B. McMickell and B. Goodwine, “Motion planning for nonlinear symmetric distributed robotic formations,” *The International Journal of Robotics Research*, vol. 26, no. 10, pp. 1025–1041, 2007.

[35] R. W. Brockett, “Control theory and singular Riemannian geometry,” in *New Directions in Applied Mathematics.* Springer, 1982, pp. 11–27.

[36] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control.* Springer, 2013, vol. 10.

[37] A. R. Teel, R. M. Murray, and G. C. Walsh, "Non-holonomic control systems: from steering to stabilization with sinusoids," *International Journal of Control*, vol. 62, no. 4, pp. 849–870, 1995.

[38] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," *Robot Motion Planning and Control*, vol. 229, pp. 1–53, 1998.

[39] K. A. Morgansen, V. Duidam, R. J. Mason, J. W. Burdick, and R. M. Murray, "Nonlinear control methods for planar carangiform robot fish locomotion," in *International Conference on Robotics and Automation (ICRA)*, vol. 1, 2001, pp. 427–434.

[40] F. Lamiraux and J.-P. Laumond, "Flatness and small-time controllability of multibody mobile robots: Application to motion planning," *IEEE Transactions on Automatic Control*, vol. 45, no. 10, pp. 1878–1881, 2000.

[41] M. Rathinam and R. M. Murray, "Configuration flatness of Lagrangian systems underactuated by one control," *SIAM Journal on Control and Optimization*, vol. 36, no. 1, pp. 164–179, 1998.

[42] I. M. Ross and F. Fahroo, "Pseudospectral methods for optimal motion planning of differentially flat systems," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1410–1413, 2004.

[43] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[44] P. Rouchon, M. Fliess, J. Levine, and P. Martin, "Flatness and motion planning: The car with n trailers," in *European Control Conference (ECC)*, 1993, pp. 1518–1522.

[45] F. Bullo and K. M. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 402–412, 2001.

[46] K. M. Lynch, N. Shiroma, H. Arai, and K. Tanie, "Collision-free trajectory planning for a 3-DOF robot with a passive joint," *The International Journal of Robotics Research*, vol. 19, no. 12, pp. 1171–1184, 2000.

[47] T. D. Murphey and J. W. Burdick, "The power dissipation method and kinematic reducibility of multiple-model robotic systems," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 694–710, 2006.

[48] P. Choudhury and K. M. Lynch, "Trajectory planning for kinematically controllable underactuated mechanical systems," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 559–575.

[49] E. Theodorou and F. J. Valero-Cuevas, "Optimality in neuromuscular systems," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2010, pp. 4510–4516.

[50] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 377–382, 2002.

[51] T. Fan and T. Murphey, "Online feedback control for input-saturated robotic systems on Lie groups," in *Robotics: Science and Systems Conference (RSS)*, 2016.

[52] S. Aseev and V. Veliov, "Needle variations in infinite-horizon optimal control," *Variational and Optimal Control Problems on Unbounded Domains*, vol. 619, pp. 1–17, 2014.

[53] M. S. Shaikh and P. E. Caines, "On the hybrid optimal control problem: theory and algorithms," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1587–1603, 2007.

[54] A. Ansari, K. Flaßkamp, and T. D. Murphey, "Sequential action control for tracking of free invariant manifolds," in *Conference on Analysis and Design of Hybrid Systems*, In Press.

[55] A. D. Wilson, J. Schultz, A. Ansari, and T. D. Murphey, "Real-time trajectory synthesis for information maximization using sequential action control and least-squares estimation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, In Press.

[56] I. Abraham, G. De La Torre, and T. D. Murphey, "Model-based control using koopman operators," *arXiv preprint arXiv:1709.01568*, 2017.

[57] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the Koopman operator and model predictive control," in *Proceedings of Robotics: Science and Systems*, 2019.

[58] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2018, pp. 39–45.

[59] B. Klaassen, R. Linnemann, D. Spenneberg, and F. Kirchner, "Biomimetic walking robot scorpion: Control and modeling," *Robotics and autonomous systems*, vol. 41, no. 2-3, pp. 69–76, 2002.

[60] Y. F. Zheng, H. Wang, S. Li, Y. Liu, D. Orin, K. Sohn, Y. Jun, and P. Oh, "Humanoid robots walking on grass, sands and rocks," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013, pp. 1–6.

[61] G. Kan-feng and Z. Ming-yang, "Dynamic modeling and simulation of driving control for wheeled mobile robot on sand," *Journal of System Simulation*, vol. 20, no. 18, pp. 5035–5039, 2008.

[62] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: Recent advances and new challenges," in *IFAC Conference of Manoeuvering and Control of Marine Craft*, vol. 88, 2006, pp. 1–12.

[63] G. Mamakoukas, M. A. MacIver, and T. D. Murphey, "Feedback synthesis for underactuated systems using sequential second-order needle variations," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1826–1853, 2018.

[64] J. Yuh, "Modeling and control of underwater robotic vehicles," *IEEE Transactions on Systems, man, and Cybernetics*, vol. 20, no. 6, pp. 1475–1483, 1990.

[65] G. Mamakoukas, M. A. MacIver, and T. D. Murphey, "Sequential action control for models of underactuated underwater vehicles in a planar ideal fluid," in *Proceedings of the American Control Conference*, 2016, pp. 4500–4506.

[66] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[67] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.

[68] O. Ennasr, G. Mamakoukas, M. Castaño, D. Coleman, T. Murphey, and X. Tan, "Adaptive single action control policies for linearly parameterized systems," in *Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

[69] R. Johansson, A. Robertsson, K. Nilsson, and M. Verhaegen, "State-space system identification of robot manipulator dynamics," *Mechatronics*, vol. 10, no. 3, pp. 403–418, 2000.

[70] J. Swevers, W. Verdonck, and J. De Schutter, “Dynamic model identification for industrial robots,” *IEEE control systems magazine*, vol. 27, no. 5, pp. 58–71, 2007.

[71] W. He, W. Ge, Y. Li, Y.-J. Liu, C. Yang, and C. Sun, “Model identification and control design for a humanoid robot,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 45–57, 2016.

[72] B. O. Koopman, “Hamiltonian systems and transformation in hilbert space,” *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[73] J. L. Proctor, S. L. Brunton, and J. N. Kutz, “Generalizing Koopman theory to allow for inputs and control,” *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.

[74] I. Mezić, “On applications of the spectral theory of the Koopman operator in dynamical systems and control theory,” in *Proceedings of the Conference on Decision and Control*, 2015, pp. 7034–7041.

[75] M. Budišić, R. Mohr, and I. Mezić, “Applied Koopmanism,” *Chaos*, vol. 22, no. 4, p. 047510, 2012.

[76] M. Korda and I. Mezić, “Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control,” *Automatica*, vol. 93, pp. 149–160, 2018.

[77] J. P. Hespanha, *Linear systems theory.* Princeton university press, 2018.

[78] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, “Dynamic textures,” *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

[79] Y. Li, W. Li, V. Mahadevan, and N. Vasconcelos, “Vlad3: Encoding dynamics of deep features for action recognition,” in *Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1951–1960.

[80] H. Wang, C. Yuan, G. Luo, W. Hu, and C. Sun, “Action recognition using linear dynamic systems,” *Pattern Recognition*, vol. 46, no. 6, pp. 1710–1718, 2013.

[81] A. Mauroy and J. Goncalves, “Linear identification of nonlinear systems: A lifting technique based on the Koopman operator,” in *Proceedings of the Conference on Decision and Control*, 2016, pp. 6500–6505.

[82] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, “A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition,” *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[83] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven approximations of dynamical systems operators for control," *arXiv preprint arXiv:1902.10239*, 2019.

[84] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, "Extending data-driven Koopman analysis to actuated systems," in *Proceedings of the IFAC Symposium on Nonlinear Control Systems*, 2016, pp. 704–709.

[85] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PloS One*, vol. 11, no. 2, p. e0150171, 2016.

[86] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control," *https://arxiv.org/pdf/1707.01146*, 2017.

[87] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman invariant subspaces for dynamic mode decomposition," in *Proceedings of the Neural Information Processing Systems*, 2017, pp. 1130–1140.

[88] M. Haseli and J. Cortés, "Efficient identification of linear evolutions in nonlinear vector fields: Koopman invariant subspaces," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1746–1751.

[89] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[90] C. Folkestad, D. Pastor, I. Mezic, R. Mohr, M. Fonoberova, and J. Burdick, "Extended dynamic mode decomposition with learned koopman eigenfunctions for prediction and control," *arXiv preprint arXiv:1911.08751*, 2019.

[91] H. Arbabi and I. Mezic, "Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator," *SIAM Journal on Applied Dynamical Systems*, vol. 16, no. 4, pp. 2096–2126, 2017.

[92] G. Mamakoukas, M. L. Castaño, X. Tan, and T. D. Murphey, "Local Koopman operators for data-driven control of robotic systems," in *Proceedings of Robotics: Science and Systems*, 2019.

[93] A. Mauroy and J. Goncalves, "Koopman-based lifting techniques for nonlinear systems identification," *arXiv preprint arXiv:1709.02003*, 2017.

[94] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, no. 1, p. 4950, 2018.

[95] I. Abraham, G. De La Torre, and T. D. Murphey, "Model-based control using Koopman operators," in *Proceedings of Robotics: Science and Systems*, 2017.

[96] A. Salova, J. Emenheiser, A. Rupe, J. P. Crutchfield, and R. M. D'Souza, "Koopman operator and its approximations for systems with symmetries," *Chaos*, vol. 29, no. 9, p. 093128, 2019.

[97] S. Peitz and S. Klus, "Koopman operator-based model reduction for switched-system control of PDEs," *Automatica*, vol. 106, pp. 184–191, 2019.

[98] S. Sinha, U. Vaidya, and E. Yeung, "On computation of Koopman operator from sparse data," in *Proceedings of the American Control Conference*, 2019, pp. 5519–5524.

[99] B. Huang, X. Ma, and U. Vaidya, "Data-driven nonlinear stabilization using Koopman operator," *arXiv preprint arXiv:1901.07678*, 2019.

[100] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," in *Proceedings of the American Control Conference*, 2019, pp. 4832–4839.

[101] G. Mamakoukas, M. L. Castano, X. Tan, and T. D. Murphey, "Derivative-based Koopman operators for real-time control of robotic systems," *Transactions on Robotics (TRO)*, 2021.

[102] N. L. C. Chui and J. M. Maciejowski, "Realization of stable models with subspace methods," *Automatica*, vol. 32, no. 11, pp. 1587–1595, 1996.

[103] S. Sinha, B. Huang, and U. Vaidya, "Robust approximation of Koopman operator and prediction in random dynamical systems," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 5491–5496.

[104] S. Sinha, U. Vaidya, and E. Yeung, "On computation of Koopman operator from sparse data," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 5519–5524.

[105] S. Sinha, H. Bowen, and U. Vaidya, "On robust computation of koopman operator and prediction in random dynamical systems," *arXiv preprint arXiv:1803.08562*, 2018.

[106] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using

penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, 2019.

[107] W.-b. Huang, L. le Cao, F. Sun, D. Zhao, H. Liu, and S. Yu, "Learning stable linear dynamical systems with the weighted least square method." in *IJCAI*, 2016, pp. 1599–1605.

[108] M. Egerstedt, Y. Wardi, and H. Axelsson, "Transition-time optimization for switched-mode dynamical systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 110–115, 2006.

[109] G. Mamakoukas, M. A. MacIver, and T. D. Murphey, "Superlinear convergence using controls based on second-order needle variations," in *Conference on Decision and Control (CDC)*, 2018, pp. 4301–4308.

[110] Y. Wardi, M. Egerstedt, and P. Twu, "A controlled-precision algorithm for mode-switching optimization," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on.* IEEE, 2012, pp. 713–718.

[111] T. M. Caldwell and T. D. Murphey, "Projection-based optimal mode scheduling," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on.* IEEE, 2013, pp. 5307–5314.

[112] I. D. Neveln, Y. Bai, J. B. Snyder, J. R. Solberg, O. M. Curet, K. M. Lynch, and M. A. MacIver, "Biomimetic and bio-inspired robotics in electric fish research," *Journal of Experimental Biology*, vol. 216, no. Pt 13, pp. 2501–2514, Jul. 2013.

[113] O. M. Curet, N. A. Patankar, G. V. Lauder, and M. A. MacIver, "Mechanical properties of a bio-inspired robotic knifefish with an undulatory propulsor," *Bioinspir. Biomim.*, vol. 6, no. 2, 2011.

[114] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model," in *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, vol. 1. IEEE, 2006, pp. 781–786.

[115] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot motion planning and control.* Springer, 1998, pp. 171–253.

[116] D. Babineau, J. E. Lewis, and A. Longtin, "Spatial acuity and prey detection in weakly electric fish," *PLoS Comput Biol*, vol. 3, no. 3, p. e38, 2007.

[117] M. MacIver, E. Fontaine, and J. W. Burdick, "Designing future underwater vehicles: Principles and mechanisms of the weakly electric fish," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 3, pp. 651–659, 2004.

[118] M. Porez, V. Lebastard, A. J. Ijspeert, and F. Boyer, "Multi-physics model of an electric fish-like robot: Numerical aspects and application to obstacle avoidance," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on.* IEEE, 2011, pp. 1901–1906.

[119] J. R. Solberg, K. M. Lynch, and M. A. MacIver, "Active electrolocation for underwater target localization," *International Journal of Robotics Research*, vol. 27, no. 5, pp. 529–548, 2008.

[120] J. B. Snyder, M. E. Nelson, J. W. Burdick, and M. A. MacIver, "Omnidirectional sensory and motor volumes in an electric fish," *PLoS Biology*, vol. 5, no. 11, pp. 2671–2683, 2007.

[121] B. Pagurek and C. M. Woodside, "The conjugate gradient method for optimal control problems with bounded control variables," *Automatica (Journal of IFAC)*, vol. 4, no. 5-6, pp. 337–349, 1968.

[122] V. Quintana and E. Davison, "Clipping-off gradient algorithms to compute optimal controls with constrained magnitude," *International Journal of Control*, vol. 20, no. 2, pp. 243–255, 1974.

[123] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, 1944.

[124] L. Lasdon, S. Mitter, and A. Waren, "The conjugate gradient method for optimal control problems," *IEEE Transactions on Automatic Control*, vol. 12, no. 2, pp. 132–138, 1967.

[125] W. Sun, E. A. Theodorou, and P. Tsiotras, "Game theoretic continuous time differential dynamic programming," in *American Control Conference (ACC), 2015.* IEEE, 2015, pp. 5593–5598.

[126] ——, "Continuous-time differential dynamic programming with terminal constraints," in *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL).* IEEE, 2014, pp. 1–6.

[127] A. N. K. Nasir, M. A. Ahmad, and M. F. Rahmat, "Performance comparison between LQR and PID controllers for an inverted pendulum system," in *AIP Conference Proceedings*, vol. 1052, no. 1. AIP, 2008, pp. 124–128.

[128] E. Tzorakoleftherakis and T. D. Murphey, "Controllers as filters: Noise-driven swing-up control based on maxwell's demon," in *54th Annual Conference on Decision and Control (CDC).* IEEE, 2015, pp. 4368–4374.

[129] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1168–1175.

[130] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control.* Wiley New York, 2006, vol. 3.

[131] F. Plestan, J. W. Grizzle, E. R. Westervelt, and G. Abba, "Stable walking of a 7-dof biped robot," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, pp. 653–668, 2003.

[132] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback control of dynamic bipedal robot locomotion.* CRC press, 2007, vol. 28.

[133] J. H. Park and K. D. Kim, "Biped robot walking using gravity-compensated inverted pendulum mode and computed torque control," in *1998 IEEE International Conference on Robotics and Automation, 1998. Proceedings.*, vol. 4. IEEE, 1998, pp. 3528–3533.

[134] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE International Conference on Robotics and Automation, 2003. Proceedings.*, vol. 2. IEEE, 2003, pp. 1620–1626.

[135] Q. Huang, K. Kaneko, K. Yokoi, S. Kajita, T. Kotoku, N. Koyachi, H. Arai, N. Imamura, K. Komoriya, and K. Tanie, "Balance control of a biped robot combining off-line pattern with real-time modification," in *IEEE International Conference on Robotics and Automation, 2000. Proceedings.*, vol. 4. IEEE, 2000, pp. 3346–3352.

[136] P.-B. Wieber and C. Chevallereau, "Online adaptation of reference trajectories for the control of walking systems," *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 559–566, 2006.

[137] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 5134–5140.

[138] M. Rutschmann, B. Satzinger, M. Byl, and K. Byl, “Nonlinear model predictive control for rough-terrain robot hopping,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1859–1864.

[139] Y. Jian, D. A. Winter, M. G. Ishac, and L. Gilchrist, “Trajectory of the body cog and cop during initiation and termination of gait,” *Gait & Posture*, vol. 1, no. 1, pp. 9–22, 1993.

[140] B. G. Buss, K. A. Hamed, B. A. Griffin, and J. W. Grizzle, “Experimental results for 3d bipedal robot walking based on systematic optimization of virtual constraints,” in *American Control Conference (ACC)*, 2016, pp. 4785–4792.

[141] E. Tzorakoleftherakis, “Stable control synthesis for human-in-the-loop systems,” Ph.D. dissertation, Northwestern University, 2017, unpublished thesis.

[142] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, “Hybrid zero dynamics of planar biped walkers,” *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 42–56, 2003.

[143] D. Dimitrov, P.-B. Wieber, O. Stasse, H. J. Ferreau, and H. Diedam, “An optimized linear model predictive control solver for online walking motion generation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 1171–1176.

[144] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, “Online walking motion generation with automatic footstep placement,” *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[145] B. J. Stephens and C. G. Atkeson, “Dynamic balance force control for compliant humanoid robots,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 1248–1255.

[146] J. M. Wang, D. J. Fleet, and A. Hertzmann, “Optimizing walking controllers,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 168, 2009.

[147] G. Mamakoukas, M. A. MacIver, and T. D. Murphey, “Feedback synthesis for controllable underactuated systems using sequential second order actions,” in *Robotics: Science and Systems (RSS)*, 2017.

[148] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko, “The mathematical theory of optimal processes,” *New York: Interscience*, 1962.

[149] A. Dmitruk and N. Osmolovskii, “On the proof of Pontryagin’s maximum principle by means of needle variations,” *arXiv preprint arXiv:1412.2363*, 2014.

[150] M. Garavello and B. Piccoli, "Hybrid necessary principle," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1867–1887, 2005.

[151] T. M. Caldwell and T. D. Murphey, "Projection-based iterative mode scheduling for switched systems," *Nonlinear Analysis: Hybrid Systems*, vol. 21, pp. 59–83, 2016.

[152] ——, "Switching mode generation and optimal estimation with application to skid-steering," *Automatica*, vol. 47, no. 1, pp. 50–64, 2011.

[153] B. Jakubczyk, "Introduction to geometric nonlinear control; controllability and Lie bracket," *Mathematical Control Theory*, vol. 1, no. 2, pp. 107–168, 2001.

[154] P. Rashevsky, "About connecting two points of a completely nonholonomic space by admissible curve," *Uch. Zapiski Ped. Inst. Libknechta*, vol. 2, pp. 83–94, 1938.

[155] W. Chow, "Über Systeme von linearen partiellen Differentialgleichungen erster Ordnung." *Mathematische Annalen*, vol. 117, pp. 98–105, 1940/1941. [Online]. Available: http://eudml.org/doc/160043

[156] W. Murray, "Newton-type methods," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[157] R. B. Schnabel and E. Eskow, "A new modified Cholesky factorization," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 6, pp. 1136–1158, 1990.

[158] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[159] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006, ch. 3.

[160] D. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. IET, 2004.

[161] J. B. Kuipers, *Quaternions and Rotation Sequences*. Princeton, NJ: Princeton Univ. Press, 1999.

[162] A. E. C. da Cunha, "Benchmark: Quadrotor attitude control." in *Applied Verification for Continuous and Hybrid Systems (ARCH)*, 2015, pp. 57–72.

[163] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[164] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, “Randomized kinodynamic motion planning with moving obstacles,” *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[165] S. Prajna, A. Jadbabaie, and G. J. Pappas, “A framework for worst-case and stochastic safety verification using barrier certificates,” *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[166] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, “Control barrier certificates for safe swarm behavior,” *Analysis and Design of Hybrid Systems*, vol. 48, no. 27, pp. 68–73, 2015.

[167] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, “Robustness of control barrier functions for safety critical control,” *the IFAC Conference on Analysis and Design of Hybrid Systems*, vol. 48, no. 27, pp. 54–61, 2015.

[168] L. Wang, A. D. Ames, and M. Egerstedt, “Safety barrier certificates for collisions-free multirobot systems,” *IEEE Transactions on Robotics*, 2017.

[169] A. D. Ames, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs with application to adaptive cruise control,” in *IEEE Conference on Decision and Control (CDC)*, 2014, pp. 6271–6278.

[170] G. Wu and K. Sreenath, “Safety-critical geometric control for systems on manifolds subject to time-varying constraints”,” *IEEE Transactions on Automatic Control, in review*, 2016.

[171] M. Deng, A. Inoue, and K. Sekiguchi, “Lyapunov function-based obstacle avoidance scheme for a two-wheeled mobile robot,” *Journal of Control Theory and Applications*, vol. 6, no. 4, pp. 399–404, 2008.

[172] H. G. Tanner, S. Loizou, and K. J. Kyriakopoulos, “Nonholonomic stabilization with collision avoidance for mobile robots,” in *International Conference on Intelligent Robots and Systems*, vol. 3, 2001, pp. 1220–1225.

[173] E. Rimon and D. E. Koditschek, “Exact robot navigation using artificial potential functions,” *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[174] P. J. Antsaklis and A. N. Michel, *A Linear Systems Primer.* Springer Science & Business Media, 2006, ch. 3, pp. 116–119.

[175] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the Koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.

[176] Y. Lan and I. Mezić, "Linearization in the large of nonlinear systems and Koopman operator spectrum," *Physica D: Nonlinear Phenomena*, vol. 242, no. 1, pp. 42–53, 2013.

[177] S. P. Nandanoori, S. Sinha, and E. Yeung, "Data-driven operator theoretic methods for global phase space learning," *arXiv preprint arXiv:1910.03011*, 2019.

[178] M. Haseli and J. Cortés, "Learning Koopman Eigenfunctions and Invariant Subspaces from Data: Symmetric Subspace Decomposition," *arXiv preprint arXiv:1909.01419*, 2020.

[179] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning koopman invariant subspaces for dynamic mode decomposition," *arXiv preprint arXiv:1710.04340*, 2017.

[180] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control," *arXiv preprint arXiv:1707.01146*, 2017.

[181] ——, "Discovering conservation laws from data for control," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6415–6421.

[182] S. Pan, N. Arnold-Medabalimi, and K. Duraisamy, "Sparsity-promoting algorithms for the discovery of informative koopman invariant subspaces," *arXiv preprint arXiv:2002.10637*, 2020.

[183] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using Koopman operator theory," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6244–6250.

[184] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[185] K. P. Murphy, *Machine learning: A Probabilistic Perspective*. Cambridge, MA: The MIT Press, 2012.

[186] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York: Springer, 2013, vol. 112.

[187] L. Ljung, *System Identification*. Upper Saddle River, NJ: Prentice Hall, 1998.

[188] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models," *Nature Communications*, vol. 6, p. 8133, 2015.

[189] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos, "Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis," *Communications in Mathematical Sciences*, vol. 1, no. 4, pp. 715–762, 2003.

[190] Z. Hou and S. Jin, "A novel data-driven control approach for a class of discrete-time nonlinear systems," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 6, pp. 1549–1558, 2011.

[191] J. R. Cloutier, "State-dependent Riccati equation techniques: an overview," in *Proceedings of the 1997 American control conference (Cat. No. 97CH36041)*, vol. 2. IEEE, 1997, pp. 932–936.

[192] T. Çimen, "State-dependent Riccati equation (SDRE) control: A survey," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 3761–3775, 2008.

[193] T. Carleman, "Application de la théorie des équations intégrales linéaires aux systèmes d'équations différentielles non linéaires," *Acta Mathematica*, vol. 59, no. 1, pp. 63–87, 1932.

[194] K. Kowalski and W.-H. Steeb, *Nonlinear Dynamical Systems and Carleman Linearization*. Teaneck, NJ: World Scientific, 1991.

[195] S. Banks, "Infinite-dimensional Carleman linearization, the Lie series and optimal control of non-linear partial differential equations," *International Journal of Systems Science*, vol. 23, no. 5, pp. 663–675, 1992.

[196] G. V. Bard, "Numerically estimating derivatives during simulations," in *Proceedings of the International Conference on Modelling, Simulation, and Visualization Methods*, 2011, pp. 341–347.

[197] P. J. Olver, *Introduction to partial differential equations*. Springer, 2014.

[198] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[199] O. Nelles, "Nonlinear dynamic system identification," in *Nonlinear System Identification*. Springer, 2001, pp. 547–577.

[200] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, "Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data," *Journal of Open Source Software*, vol. 5, no. 49, p. 2104, 2020. [Online]. Available: https://doi.org/10.21105/joss.02104

[201] I. The MathWorks, *MATLAB Deep Learning Toolbox*, Natick, Massachusetts, United State, 2019b. [Online]. Available: https://www.mathworks.com/help/deeplearning

[202] J. Wang and X. Tan, "Averaging tail-actuated robotic fish dynamics through force and moment scaling," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 906–917, 2015.

[203] M. Castaño and X. Tan, "Model Predictive Control-Based Path-Following for Tail-Actuated Robotic Fish," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 7, pp. 1–11, 2019.

[204] M. L. Castaño and X. Tan, "Backstepping control-based trajectory tracking for tail-actuated robotic fish," in *Proceedings of the International Conference on Advanced Intelligent Mechatronics*, 2019, pp. 839–844.

[205] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear Koopman realizations for the modeling and control of systems with unknown dynamics," *arXiv preprint arXiv:2010.09961*, 2020.

[206] G. Mamakoukas, O. Xherija, and T. D. Murphey, "Memory-Efficient Learning of Stable Linear Dynamical Systems for Prediction and Control," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[207] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning stable models for prediction and control," *https://arxiv.org/abs/2005.04291*, 2020.

[208] J. Quevedo, "Digital control: past, present and future of pid control," in *Proceedings of IFAC Workshop*, 2000.

[209] J. Yu, M. Tan, S. Wang, and E. Chen, "Development of a biomimetic robotic fish and its control algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 4, pp. 1798–1810, 2006.

[210] Q. Ren, J. Xu, and X. Li, "A motion control approach for a robotic fish with iterative feedback tuning," in *Proceedings of the International Conference on Industrial Technology*, 2015, pp. 40–45.

[211] F. Haugen, *PID Control.* Tapir Academic Press, 2004, ch. 4, pp. 94–98.

[212] N. Gillis, M. Karow, and P. Sharma, “A note on approximating the nearest stable discrete-time descriptor systems with fixed rank,” *Applied Numerical Mathematics*, vol. 148, pp. 131–139, 2020.

[213] R. A. Horn and C. R. Johnson, *Matrix analysis.* Cambridge university press, 2012.

[214] S. Soatto, G. Doretto, and Y. N. Wu, “Dynamic textures,” in *International Conference on Computer Vision*, vol. 2, 2001, pp. 439–446.

[215] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory—Implementation—Applications.* Springer Science & Business Media, 2012.

[216] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto, “Dynamic texture recognition,” in *Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2001.

[217] A. B. Chan and N. Vasconcelos, “Probabilistic kernels for the classification of autoregressive visual processes,” in *Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 846–851.

[218] R. Péteri, S. Fazekas, and M. J. Huiskes, “Dyntex: A Comprehensive Database of Dynamic Textures,” *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1627–1632, 2010.

[219] M. Szummer and R. W. Picard, “Temporal texture modeling,” in *International Conference on Image Processing*, vol. 3, 1996, pp. 823–826.

[220] N. Borovykh and M. Spijker, “Resolvent conditions and bounds on the powers of matrices, with relevance to numerical stability of initial value problems,” *Journal of Computational and Applied Mathematics*, vol. 125, no. 1-2, pp. 41–56, 2000.

[221] P. Shcherbakov, “On peak effects in discrete time linear systems,” in *2017 25th Mediterranean Conference on Control and Automation (MED).* IEEE, 2017, pp. 376–381.

[222] U. Ahiyevich, S. E. Parsegov, and P. S. Shcherbakov, “Upper bounds on peaks in discrete-time linear systems,” *Automation and Remote Control*, vol. 79, no. 11, pp. 1976–1988, 2018.

[223] G. Halikias, L. Dritsas, A. Pantelous, and V. Tsoulkas, “Strong stability of discrete-time systems,” *Linear Algebra and its Applications*, vol. 436, no. 7, pp. 1890–1908, 2012.

[224] T. A. Berrueta, A. Pervan, K. Fitzsimons, and T. D. Murphey, “Dynamical system segmentation for information measures in motion,” *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 169–176, 2018.

[225] W. M. Haddad and V. Chellaboina, *Nonlinear dynamical systems and control: a Lyapunov-based approach.* Princeton university press, 2011.

[226] N. Gillis, M. Karow, and P. Sharma, “Approximating the nearest stable discrete-time system,” *Linear Algebra and its Applications*, vol. 573, pp. 37–53, 2019.

[227] B. D. Anderson, T. S. Brinsmead, F. De Bruyne, J. Hespanha, D. Liberzon, and A. S. Morse, “Multiple model adaptive control. part 1: Finite controller coverings,” *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 10, no. 11-12, pp. 909–929, 2000.

[228] J. Hespanha, D. Liberzon, A. Stephen Morse, B. D. Anderson, T. S. Brinsmead, and F. De Bruyne, “Multiple model adaptive control. part 2: switching,” *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 11, no. 5, pp. 479–496, 2001.

[229] I. Abraham and T. D. Murphey, “Active learning of dynamics for data-driven control using koopman operators,” *arXiv preprint arXiv:1906.05194*, 2019.

[230] F. R. Hogan and A. Rodriguez, “Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics,” in *Algorithmic Foundations of Robotics XII.* Springer, 2020, pp. 800–815.

[231] M. Bauza and A. Rodriguez, “A probabilistic data-driven model for planar pushing,” in *2017 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2017, pp. 3008–3015.

[232] M. Bauza, F. R. Hogan, and A. Rodriguez, “A data-efficient approach to precise and controlled pushing,” *arXiv preprint arXiv:1807.09904*, 2018.

[233] M. L. Castaño, A. Hess, G. Mamakoukas, T. Gao, T. Murphey, and X. Tan, “Control-oriented modeling of soft robotic swimmer with koopman operators,” in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 1679–1685.

[234] W. Edwards, G. Tang, G. Mamakoukas, T. Murphey, and K. Hauser, “Automatic tuning for data-driven model predictive control,” in *International Conference on Robotics and Automation (ICRA)*, 2021.

[235] K. MacMillan and J. Sondow, "Proofs of power sum and binomial coefficient congruences via Pascal's identity," *The American Mathematical Monthly*, vol. 118, no. 6, pp. 549–551, 2011.

APPENDIX A

# Derivations and Proofs

## A.1. Derivation of the Mode Insertion Hessian

Consider switched systems that are defined by dynamics

$$\dot{x}(t)=f\Big(x(t),\Lambda,t\Big)=\begin{cases} f_1\Big(x(t),t\Big) & , \quad T_0\leq t<\tau_1 \\ f_2\Big(x(t),t\Big) & , \quad \tau_1\leq t<\tau_1+\lambda_1 \\ f_1\Big(x(t),t\Big) & , \quad \tau_1+\lambda_1\leq t<\tau_2 \\ f_3\Big(x(t),t\Big) & , \quad \tau_2\leq t<\tau_2+\lambda_2 \\ f_1\Big(x(t),t\Big) & , \quad \tau_2+\lambda_2\leq t<\tau_3 \\ \vdots & \quad \vdots \qquad \vdots \\ f_1\Big(x(t),t\Big) & , \quad \tau_L+\lambda_L\leq t<T_F \end{cases} \tag{A.1}$$

$$\text{subject to: } x(T_0)=x_0,$$

where $T_0$ is the initial time, $T_F$ is the final time, $x_0{:}\mathbb{R}\mapsto\mathbb{R}^N$ is the initial state, $L$ is the number of injected dynamics, $\tau=\{\tau_1,\tau_2,\cdots,\tau_L\}\in\mathbb{R}^L$ is a monotonically increasing set of switching times, $\Lambda=\{\lambda_1,\lambda_2,\cdots,\lambda_L\}\in\mathbb{R}^L$ is a set of control durations, $f_1{:}\mathbb{R}\mapsto\mathbb{R}^N$ specify the default dynamics, and $f_i{:}\mathbb{R}\mapsto\mathbb{R}^N$ describe the $i^{\text{th}}$ injected dynamics. The switching times are assumed to be fixed.

Note that, while the system dynamics $f$ depend on the set of control durations $\Lambda$, the same is not true for the individual switch mode dynamics $f_i$. In addition, I refer to individual elements in the set $\Lambda$ as either $\Lambda_i$ or $\lambda_i$. I measure the performance of the system with the integral of the Lagrangian, $\ell(\cdot)$, and a terminal cost $m(\cdot)$, similar to (2.3).

$$J(\Lambda)=\int_{T_0}^{T_F}\ell(x(t))\mathrm{d}t+m(x(T_F)). \tag{A.2}$$

Re-writing the dynamics using step functions gives

$$\begin{aligned}
f(x(t),\Lambda,t) =&\Big[1(t-T_0)-1(t-\tau_1^-)\Big]f_1\Big(x(t),t\Big)\\
&+\Big[1(t-\tau_1^+)-1\Big(t-(\tau_1+\lambda_1)^-\Big)\Big]f_2\Big(x(t),t\Big)\\
&+\Big[1\Big(t-(\tau_1+\lambda_1)^+\Big)-1(t-\tau_2^-)\Big]f_1\Big(x(t),t\Big)\\
&+\ldots\\
&+\Big[1(t-(\tau_{L-1}+\lambda_{L-1})^+)-1(t-\tau_L^-)\Big]f_1\Big(x(t),t\Big)\\
&+\Big[1(t-\tau_L^+)-1\Big(t-(\tau_L+\lambda_L)^-\Big)\Big]f_L\Big(x(t),t\Big)\\
&+\Big[1\Big(t-(\tau_L+\lambda_L)^+\Big)-1(t-T_F)\Big]f_1\Big(x(t),t\Big).
\end{aligned}$$

The superscripts $+$ and $-$ help avoid ambiguity at the switching times. I use directional derivatives to differentiate, where the slot derivative $D_i\mathcal{F}(\cdot,\cdot)$ is the partial derivative of a function $\mathcal{F}$ with respect to its $i^{\text{th}}$ argument. That is, $D_x\mathcal{F}$ indicates the derivative of $\mathcal{F}$ with respect to $x$ and is the same as $\frac{\partial\mathcal{F}}{\partial x}$. Further, $D_{i,j}\mathcal{F}(\cdot,\cdot)$ denotes the second partial of a function $\mathcal{F}$ with respect to its first and second arguments. The step-function form of the dynamics makes it straightforward to compute the partial derivatives $D_1 f\Big(x(t),\Lambda,t\Big)$

and $D_2 f\Big(x(t),\Lambda,t\Big)$. Specifically,

$$\begin{aligned} D_1 f\Big(x(t),\Lambda,t\Big) =& \Big[1(t-T_0)-1(t-\tau_1^-)\Big] D_1 f_1\Big(x(t),t\Big) \\ &+\Big[1(t-\tau_1^+)-1\Big(t-(\tau_1+\lambda_1)^-\Big)\Big] D_1 f_2\Big(x(t),t\Big) \\ &+\Big[1\Big(t-(\tau_1+\lambda_1)^+\Big)-1(t-\tau_2^-)\Big] D_1 f_1\Big(x(t),t\Big) \\ &+\dots \\ &+\Big[1(t-(\tau_{L-1}+\lambda_{L-1})^+)-1(t-\tau_L^-)\Big] D_1 f_1\Big(x(t),t\Big) \\ &+\Big[1(t-\tau_L^+)-1\Big(t-(\tau_L+\lambda_L)^-\Big)\Big] D_1 f_L\Big(x(t),t\Big) \\ &+\Big[1\Big(t-(\tau_L+\lambda_L)^+\Big)-1(t-T_F)\Big] D_1 f_1\Big(x(t),t\Big) \end{aligned}$$

and

$$\text{(A.3)} \qquad D_2 f\Big(x(t),\Lambda,t\Big)=\Big\{\delta\Big(t-(\tau_k+\lambda_k)^-\Big) f_k(x(t),t)-\delta\Big(t-(\tau_k+\lambda_k)^+\Big) f_1(x(t),t)\Big\}_{k=1}^{L},$$

where $\delta(\cdot)$ is the Dirac delta functions. Using variational calculus,

$$\text{(A.4)} \qquad DJ(\Lambda)\cdot\theta=\int_{T_0}^{T_F} D\ell\Big(x(r)\Big)\cdot z(r)\mathrm{d}r+Dm\Big(x(T_F)\Big)\cdot z(T_F)$$

where $z(t)$:$\mathbb{R}\mapsto\mathbb{R}^{N\times 1}$ is the variation of $x(t)$ due to the variation, $\theta$, in $\Lambda$. Also,

$$\dot{z}(t)=\frac{\partial}{\partial t}\frac{\partial x(t)}{\partial \Lambda}=\frac{\partial}{\partial \Lambda}\frac{\partial x(t)}{\partial t}=\frac{\partial}{\partial \Lambda}\dot{x}(t)=\frac{\partial}{\partial \Lambda}f\Big(x(t),\Lambda,t\Big)$$
$$=D_1f\Big(x(t),\Lambda,t\Big)\cdot z(t)+D_2f\Big(x(t),\Lambda,t\Big)\cdot\theta,$$
$$\text{subject to: } z(0)=\frac{\partial}{\partial \Lambda}x(0)=0.$$

Define $A(t)\triangleq D_1f\Big(x(t),\Lambda,t\Big)$ and $B(t)\triangleq D_2f\Big(x(t),\Lambda,t\Big)$. Therefore, $\dot{z}$ is

$$\dot{z}(t)=A(t)\cdot z(t)+B(t)\cdot\theta$$
$$\text{subject to: } z(0)=0.$$

The above differential equation has the solution

$$z(t)=\int_{T_0}^{t}\mathbf{\Phi}(t,r)B(r)\cdot\theta\,\mathrm{d}r, \tag{A.5}$$

where $\mathbf{\Phi}(t,r)$ is the state transition matrix corresponding to $A(t)$. Substituting $z(\cdot)$ in $DJ(\Lambda)\cdot\theta$,

$$DJ(\Lambda)\cdot\theta=\int_{T_0}^{T_F}D\ell\Big(x(r)\Big)\int_{T_0}^{r}\mathbf{\Phi}(r,s)B(s)\cdot\theta\,\mathrm{d}s\,\mathrm{d}r+Dm\Big(x(T_F)\Big)\int_{T_0}^{T_F}\mathbf{\Phi}(T_F,s)B(s)\cdot\theta\,\mathrm{d}s.$$

Switching the order of integration in the first-integral,

$$\begin{aligned} DJ(\Lambda)\cdot\theta &= \int\limits_{T_0}^{T_F}\int\limits_{s}^{T_F} D\ell\Big(x(r)\Big)\boldsymbol{\Phi}(r,s)B(s)\cdot\theta \mathrm{d}r\mathrm{d}s + \int\limits_{T_0}^{T_F} Dm\Big(x(T_F)\Big)\boldsymbol{\Phi}(T_F,s)B(s)\cdot\theta\mathrm{d}s \\ &= \int\limits_{T_0}^{T_F} \underbrace{\Big[\int\limits_{s}^{T_F} D\ell\Big(x(r)\Big)\boldsymbol{\Phi}(r,s)\mathrm{d}r + \int\limits_{T_0}^{T_F} Dm\Big(x(T_F)\Big)\boldsymbol{\Phi}(T_F,s)\Big]}_{\rho(s)^T} B(s)\mathrm{d}s\cdot\theta. \end{aligned}$$

Then,

$$\rho(t) = \boldsymbol{\Phi}(T_F,t)^T Dm(x(T_F))^T + \int\limits_{t}^{T_F} \boldsymbol{\Phi}(r,t)^T D\ell(x(r))^T \mathrm{d}r,$$

where $\rho(t)$ is the solution to the backwards differential equation:

$$\begin{gathered} \dot{\rho}(t) = -D_1 f(x(t),\Lambda,t)^T \rho - D\ell(x(t)) \\ \text{subject to: } \rho(T_F) = Dm(x(T_F))^T. \end{gathered} \tag{A.6}$$

To avoid confusion, it is important to explain the notation used in the remaining of the derivation. I use $\theta$ to represent first-order and $\eta$ to represent second-order perturbations to control durations $\Lambda$, respectively. I use subscripts to refer to the perturbation acting on a specific (single) duration. For example, $\theta_i$ indicates the perturbation that takes place with respect to the $i^{\text{th}}$ control duration, $\lambda_i$. I index the order of perturbations with a superscript, so that $\theta^j$ indicates the $j^{\text{th}}$ (in order) perturbation to the set of control durations $\Lambda$. Therefore, $\theta_1^2$ indicates the perturbation that acts on the first control duration $\lambda_1$ and that is associated with the second perturbation.

I write

$$\frac{\partial}{\partial\Lambda}(DJ(\Lambda)\cdot\theta^1)=\frac{\partial}{\partial\Lambda}(\int_{T_0}^{T_N} D\ell(x(r))\cdot z^1(r)\mathrm{d}r+Dm(x(T_F))\cdot z^1(T_F)),$$

and, using the product rule, I compute

$$D^2J(\Lambda)\cdot(\theta^1,\theta^2)+DJ(\Lambda)\cdot\eta=\int_{T_0}^{T_F} D^2\ell(x(r))\cdot\Big(z^1(r),z^2(r)\Big)+D\ell(x(r))\cdot\zeta(r)\mathrm{d}r$$
$$+D^2m\Big(x(T_F)\Big)\cdot(z^1(T_F),z^2(T_F))+Dm\Big(x(T_F)\cdot\zeta(T_F),$$

where $\theta^1$ and $\theta^2$ are two first-order variations of $\Lambda$, $\eta$ is a second-order variation of $\Lambda$ and $\zeta(t)$ is the second-order variation of $x(t)$. Parameter $\dot{\zeta}(t)$ is found by taking the second-order switching time derivative of $\dot{x}$(t):

$$\begin{aligned}\dot{\zeta}(t)&=\frac{\partial^2}{\partial\Lambda^2}\dot{x}(t)=\frac{\partial}{\partial\Lambda}\dot{z}^1(t)\\&=\frac{\partial}{\partial\Lambda}\Big(D_1f(x(t),\Lambda,t\Big)\cdot z^1(t)+D_2f\Big(x(t),\Lambda,t)\cdot\theta^1\Big)\end{aligned}$$

such that

(A.7)
$$\dot{\zeta}(t)=A(t)\cdot\zeta(t)+B(t)\cdot\eta+\Big(z^1(t)^T\quad\theta^{1^T}\Big)\begin{pmatrix}D_1^2f\Big(x(t),\Lambda,t\Big) & D_{1,2}f(x(t),\Lambda,t\Big)\\ D_{2,1}f\Big(x(t),\Lambda,t\Big) & D_2^2f\Big(x(t),\Lambda,t\Big)\end{pmatrix}\begin{pmatrix}z^2(t)\\ \theta^2\end{pmatrix}$$
$$\text{subject to: }\zeta(0)=\frac{\partial^2}{\partial\Lambda^2}x(0)=0.$$

Define

$$C(t) \triangleq \begin{pmatrix} D_1^2 f\Big(x(t),\Lambda,t\Big) & D_{1,2} f\big(x(t),\Lambda,t\big) \\ D_{2,1} f\Big(x(t),\Lambda,t\Big) & D_2^2 f\Big(x(t),\Lambda,t\Big) \end{pmatrix},$$

and notice that $\dot{\zeta}(t)$ is linear with respect to $\zeta(t)$ and therefore $\dot{\zeta}(t)$ has solution

$$\zeta(t) = \int_{T_0}^{t} \mathbf{\Phi}(t,r)\Big[B(r)\cdot\eta + \Big(z^1(r)^T {\theta^1}^T\Big) C(r) \begin{pmatrix} z^2(r) \\ \theta^2 \end{pmatrix}\Big] \mathrm{d}t.$$

Substituting $\zeta(t)$ into (A.1) gives

$$\begin{aligned} D^2 J(\Lambda)\cdot(\theta^1,\theta^2) + DJ(\Lambda)\cdot\eta = & \int_{T_0}^{T_F} \Big[ z^1(r)^T D^2\ell(x(r)) z^2(r) \\ & + D\ell(x(r)) \int_{T_0}^{r} \mathbf{\Phi}(r,s)\Big[B(s)\cdot\eta + (z^1(s)^T {\theta^1}^T) C(s) \begin{pmatrix} z^2(s) \\ \theta^2 \end{pmatrix}\Big] \mathrm{d}s \Big] \mathrm{d}r \\ & + z^1(T_F)^T D^2 m(x(T_F)) z^2(T_F) + Dm(x(T_F))\cdot \\ & \int_{T_0}^{T_F} \mathbf{\Phi}(T_F,s)\Big[B(s)\cdot\eta + (z^1(s)^T {\theta^1}^T) C(s) \begin{pmatrix} z^2(s) \\ \theta^2 \end{pmatrix}\Big] \mathrm{d}s. \end{aligned}$$

Note that $DJ(\Lambda)\cdot\eta$ equals $\int_{T_0}^{T_F} D\ell(x(r))\int_{T_0}^{r}\Phi(r,s)B(s)\cdot\eta\mathrm{d}s\mathrm{d}r+Dm(x(T_F))\int_{T_0}^{T_F}\Phi(T_F,s)B(s)\cdot\eta\mathrm{d}s$, which is clear from (A.4) and (A.5). Therefore, this leaves

$$\begin{aligned}D^2J(\Lambda)\cdot(\theta^1,\theta^2)=\int_{T_0}^{T_F}&\Big[z^1(r)D^2\ell(x(r))z^2(r)\\&+D\ell(x(r))\int_{T_0}^{\tau}\Phi(r,s)(z^1(s)^T\theta^{1^T})C(s)\begin{pmatrix}z^2(s)\\\theta^2\end{pmatrix}\mathrm{d}s\Big]\mathrm{d}r\\&+z^1(T_F)^TD^2m(x(T_F))z^2(T_F)\\&+Dm(x(T_F))\int_{T_0}^{T_F}\Phi(T_F,s)\Big(z^1(s)^T\theta^{1^T}\Big)C(s)\begin{pmatrix}z^2(s)\\\theta^2\end{pmatrix}\mathrm{d}s.\end{aligned}$$

Split the integral over d$r$, move $D\ell\Big(x(r)\Big)$ and $Dm(x(T_F))$ into their respective integrals and switch the order of integration of the double integral:

$$\begin{aligned}&=\int_{T_0}^{T_F}z^1(r)^TD^2\ell(x(r))z^2(r)\mathrm{d}r+\int_{T_0}^{T_F}\int_{s}^{T_F}D\ell(x(r))\Phi(r,s)\Big(z^1(s)^T\theta^{1^T}\Big)C(s)\begin{pmatrix}z^2(s)\\\theta^2\end{pmatrix}\mathrm{d}r\mathrm{d}s\\&\quad+z^1(T_F)^TD^2m(x(T_F))z^2(T_F)+\int_{T_0}^{T_F}Dm\Big(x(T_F)\Big)\Phi(T_F,s)\Big(z^1(s)^T,\theta^{1^T}\Big)C(s)\begin{pmatrix}z^2(s)\\\theta^2\end{pmatrix}\mathrm{d}s.\end{aligned}$$

I combine the integrals over d$s$, and notice that $\rho(r)^T$, in (A.6), enters the equations. Furthermore, I switch the dummy variable $s$ to $r$ and put everything back under one integral:

$$\begin{aligned}&=\int_{T_0}^{T_F}z^1(r)^TD^2\ell(x(r))z^2(r)+\rho(r)^T\Big(z^1(r)^T\theta^{1^T}\Big)C(r)\begin{pmatrix}z^2(r)\\\theta^2\end{pmatrix}\mathrm{d}r\\&\quad+z^1(T_F)^TD^2m(x(T_F))z^2(T_F).\end{aligned}$$

Expand $C(\cdot)$ back out,

$$
\begin{aligned}
=&\int_{T_0}^{T_F} z^1(r)^T D^2\ell(x(r))z^2(r)+\rho(r)^T\left[z^1(r)^T D_1^2 f(x(r),\Lambda,r)z^2(r)\right]\\
&+\rho(r)^T\left[z^1(r)^T D_{1,2}f(x(r),\Lambda,r)\theta^2\right]+\rho(r)^T\left[{\theta^1}^T D_{2,1}f(x(r),\Lambda,r)z^2(r)\right]\\
&+\rho(r)^T\left[{\theta^1}^T D_2^2 f(x(r),\Lambda,r)\theta^2\right]\mathrm{d}r+z^1(T_F)^T D^2m(x(T_F))z^2(T_F).
\end{aligned}
$$

Switching to index notation, where $\rho_k(\cdot)$ is the $k^{\text{th}}$ component of $\rho(\cdot)$ and $f^k(\cdot,\cdot,\cdot)$ is the $k^{\text{th}}$ component of $f(\cdot,\cdot,\cdot)$,

$$
\begin{aligned}
=&\int_{T_0}^{T_F} z^1(r)^T D^2\ell(x(r))z^2(r)+z^1(r)^T\sum_{k=1}^{n}\rho_k(r)D_1^2 f^k(x(r,\Lambda,r)z^2(r)\\
&+z^1(r)^T\sum_{k=1}^{n}\rho_k(r)D_{1,2}f^k(x(r),\Lambda,r)\theta^2+{\theta^1}^T\sum_{k=1}^{n}\rho_k(r)D_{2,1}f^k(x(r),\Lambda,r)z^2(r)\\
&+{\theta^1}^T\sum_{k=1}^{n}\rho_k(r)D_2^2 f^k(x(r),\Lambda,r)\theta^2\mathrm{d}r+z^1(T_F)^T D^2(x(T_F))z^2(T_F).
\end{aligned}
$$

Rearrange the terms allows $D^2J(\Lambda)\cdot(\theta^1,\theta^2)$ to be partitioned into the summation of parts $P_1,P_2,P_3$ given by

$$P_1=\int_{T_0}^{T_F} z^1(r)^T\Big[D^2\ell(x(r))+\sum_{k=1}^{n}\rho_k(r)D_1^2f^k\Big(x(r),\Lambda,r\Big)\Big]z^2(r)\mathrm{d}r$$

$$+z^1(T_F)^T D^2m\Big(x(T_F)\Big)z^2(T_F),$$

$$P_2=\int_{T_0}^{T_F}\theta^{2^T}\sum_{k=1}^{n}\rho_k(r)D_{2,1}f^k\Big(x(r),\Lambda,r\Big)z^1(r)+\theta^1\sum_{k=1}^{n}\rho_k(r)D_{2,1}f^k\Big(x(r),\Lambda,r\Big)z^2(r)\mathrm{d}r,$$

$$P_3=\int_{T_0}^{T_F}\theta^{1^T}\sum_{k=1}^{n}\rho_k(r)D_2^2f^k\Big(x(r),\Lambda,r\Big)\theta^2\mathrm{d}r$$

Looking at $P_1$ first, let

$$g(r)=D^2\ell\Big(x(r)\Big)+\sum_{k=1}^{n}\rho_k(r)D_1^2f^k\Big(x(r),\Lambda,r\Big).$$

Then,

$$P_1=\int_{T_0}^{T_F}z^1(r)^Tg(r)z^2(r)\mathrm{d}r+z^1(T_F)D^2m\Big(x(T_F)\Big)z^2(T_F).$$

Substituting (A.5) for $z^{\cdot}(\cdot)$, results in

$$=\int_{T_0}^{T_F}\Big[\int_{T_0}^{r}\Phi(r,s)B(s)\theta^1\mathrm{d}s\Big]^Tg(r)\int_{T_0}^{r}\Phi(r,\omega)B(\omega)\theta^2\mathrm{d}w\mathrm{d}r$$

$$+\Big[\int_{T_0}^{T_F}\Phi(T_F,s)B(s)\theta^1\mathrm{d}s\Big]^TD^2m\Big(x(T_F)\Big)\int_{T_0}^{T_F}\Phi(T_F,w)B(w)\theta^2\mathrm{d}w.$$

The integrals may be specified as follows:

$$=\int_{T_0}^{T_F}\int_{T_0}^{r}\int_{T_0}^{r}\theta^{1^T}B(s)^T\mathbf{\Phi}(r,s)^Tg(r)\mathbf{\Phi}(r,w)B(w)\theta^2\mathrm{d}s\mathrm{d}w\mathrm{d}r$$
$$+\int_{T_0}^{T_F}\int_{T_0}^{T_F}\theta^1B(s)^T\mathbf{\Phi}(T_F,s)^TD^2m\Big(x(T_F)\Big)\mathbf{\Phi}(T_F,w)B(w)\theta^2\mathrm{d}s\mathrm{d}w.$$

Note that the volume of the triple integral is given by $r{=}max(s,w)$. Therefore, the order of integration may be switched to:

$$=\int_{T_0}^{T_F}\int_{T_0}^{T_F}\int_{max(s,w)}^{T_F}\theta^{1^T}B(s)^T\mathbf{\Phi}(r,s)^Tg(r)\mathbf{\Phi}(r,w)B(w)\theta^2\mathrm{d}r\mathrm{d}s\mathrm{d}w$$
$$+\int_{T_0}^{T_F}\int_{T_0}^{T_F}\theta^1B(s)^T\mathbf{\Phi}(T_F,s)^TD^2m\Big(x(T_F)\Big)\mathbf{\Phi}(T_F,w)B(w)\theta^2\mathrm{d}s\mathrm{d}w.$$

I combine the double integral with the triple integral and rearrange the terms so that only the ones depending on $r$ are inside the internal integral:

$$=\int_{T_0}^{T_F}\int_{T_0}^{T_F}B(s)^T\Big[\int_{max(s,w)}^{T}\mathbf{\Phi}(r,s)^Tg(r)\mathbf{\Phi}(r,w)\mathrm{d}r+\mathbf{\Phi}(T_F,s)^TD^2m\Big(x(T_F)\Big)\mathbf{\Phi}(T_F,w)\Big]B(w)\mathrm{d}s\mathrm{d}w$$
$$\cdot(\theta^1,\theta^2).$$

Let

$$\mathbf{\Omega}(t)=\int_{t}^{T_F}\mathbf{\Phi}(r,t)g(r)\mathbf{\Phi}(r,t)\mathrm{d}r+\mathbf{\Phi}(T_F,t)^TD^2m\Big(x(T_F)\Big)\mathbf{\Phi}(T_F,t)$$

where $\Omega(t) \in \mathbb{R}^{n\times n}$ is the integral curve to the following differential equation

$$\begin{aligned}\dot{\Omega}(t) &= -A(t)^T\Omega(t) - \Omega(t)A(t) - g(t)\\ &= -A(t)^T\Omega(t) - \Omega(t)A(t) - D^2\ell\Big(x(t)\Big) - \sum_{k=1}^{n}\rho_k(t)D_1^2 f^k\Big(x(t),\Lambda,t\Big)\\ &\quad \text{subject to: } \Omega(T_F) = D^2 m\Big(x(T_F)\Big).\end{aligned}$$

Then, depending on the relationship between $s$ and $w$, $P_1$ becomes

$$P_1 = \begin{cases} \int_{T_0}^{T_F}\int_{T_0}^{T_F} B(s)^T\Omega(s)\Phi(s,w)B(w)\mathrm{d}s\mathrm{d}w\cdot(\theta^1,\theta^2) & s>w\\ \int_{T_0}^{T_F}\int_{T_0}^{T_F} B(s)^T\Phi(w,s)^T\Omega(w)B(w)\mathrm{d}s\mathrm{d}w\cdot(\theta^1,\theta^2) & s<w\\ \int_{T_0}^{T_F}\int_{T_0}^{T_F} B(s)^T\Omega(s)B(w)\mathrm{d}s\mathrm{d}w\cdot(\theta^1,\theta^2) & s=w,\end{cases}$$

$P_1$ is a scalar and equal to its transpose, therefore

$$P_1 \overset{s\leq w}{=} \int_{T_0}^{T_F}\int_{T_0}^{T_F} B(w)^T\Omega(w)\Phi(w,s)B(s)\mathrm{d}s\mathrm{d}w\cdot(\theta^2,\theta^1)$$

Use $i$ and $j$ to index $\theta^1$ and $\theta^2$ respectively, where $\theta$ indicate the variations of $\Lambda$ and $i,j=1,...,L$. Integrating the $\delta$-functions in $B(s)$ and $B(w)$ will pick out times $s=\tau_i+\lambda_i$

and $w{=}\tau_j{+}\lambda_j$ such that $P_{1_{ij}}$ is given by

$$P_{1_{ij}}\begin{cases} \overset{i>j}{=} & \Big[f_i\Big(x(\tau_i{+}\lambda_i),t\Big){-}f_1\Big(x(\tau_i{+}\lambda_i),t\Big)\Big]^T \\ & \Omega(\tau_i{+}\lambda_i)\Phi(\tau_i{+}\lambda_i,\tau_j{+}\lambda_j) \\ & \Big[f_j\Big(x(\tau_j{+}\lambda_j),t\Big){-}f_1\Big(x(\tau_j{+}\lambda_j),t\Big)\Big]\cdot(\theta_i^1,\theta_j^2) \\ \overset{i<j}{=} & \Big[f_j\Big(x(\tau_j{+}\lambda_j),t\Big){-}f_1\Big(x(\tau_j{+}\lambda_j),t\Big)\Big]^T \\ & \Omega(\tau_j{+}\lambda_j)\Phi(\tau_j{+}\lambda_j,\tau_i{+}\lambda_i) \\ & \Big[f_i\Big(x(\tau_i{+}\lambda_i),t\Big){-}f_1\Big(x(\tau_i{+}\lambda_i),t\Big)\Big]\cdot(\theta_i^1,\theta_j^2) \\ \overset{i=j}{=} & \Big[f_i\Big(x(\tau_i{+}\lambda_i),t\Big){-}f_1\Big(x(\tau_i{+}\lambda_i),t\Big)\Big]^T\Omega(\tau_i{+}\lambda_i) \\ & \Big[f_i\Big(x(\tau_i{+}\lambda_i),t\Big){-}f_1\Big(x(\tau_i{+}\lambda_i),t\Big)\Big]\cdot(\theta_i^1,\theta_i^2) \end{cases}$$

Taking the limit $\Lambda{\to}0$, $\lim_{\Lambda\to 0}P_{1_{ij}}$ becomes

$$\lim_{\Lambda\to 0}P_{1_{ij}}\begin{cases} \overset{i>j}{=} & [f_i(x(\tau_i),t){-}f_1(x(\tau_i),t)]^T\Omega(\tau_i)\Phi(\tau_i,\tau_j) \\ & [f_j(x(\tau_j),t){-}f_1(x(\tau_j),t)]\cdot(\theta_i^1,\theta_j^2) \\ \overset{i<j}{=} & [f_j(x(\tau_j),t){-}f_1(x(\tau_j),t)]^T\Omega(\tau_j)\Phi(\tau_j,\tau_i) \\ & [f_i(x(\tau_i),t){-}f_1(x(\tau_i),t)]\cdot(\theta_i^1,\theta_j^2) \\ \overset{i=j}{=} & [f_i(x(\tau_i),t){-}f_1(x(\tau_i),t)]\Omega(\tau_i) \\ & [f_i(x(\tau_i),t){-}f_1(x(\tau_i),t)]\cdot(\theta_i^1,\theta_i^2). \end{cases}$$

Now consider $P_2$, where

$$D_{2,1}f^k(x(t),\Lambda,t){=}\Big\{\delta\Big(t{-}(\tau_a{+}\lambda_a)^-\Big)D_1f_a^k(x(t),t)^T-\delta\Big(t{-}(\tau_a{+}\lambda_a)^+\Big)D_1f_1^k(x(t),t)^T\Big\}_{a=1}^L.$$

Choose again the $i^{\text{th}}$ index of $\theta^1$ and the $j^{\text{th}}$ index of $\theta^2$, where $i,j{=}1,...,L$. This corresponds to the $i^{\text{th}}$ index of $z^1(t)$ and the $j^{\text{th}}$ index of $z^2(t)$, where the $k^{\text{th}}$ index of $z^{\cdot}(\cdot)$ is

$$\text{(A.8)}\quad z_k^{\cdot}(t){=}\int_{T_0}^{t}\Phi(t,r)\Big[\delta\Big(r{-}(\tau_k{+}\lambda_k)^{-}\Big)f_k\Big(x(r),r\Big){-}\delta\Big(r{-}(\tau_k{+}\lambda_k)^{+}\Big)f_1\Big(x(r),r\Big)\Big]\mathrm{d}r\theta_k^{\cdot},$$

Specifying these indexes allows us to revert back to matrix representation for $\rho(\cdot)$ and $f(\cdot,\cdot,\cdot)$. Thus,

$$P_{2_{ij}}{=}\int_{T_0}^{T_F}\theta_j^2\rho(r)^T\Big[\delta\Big(r{-}(\tau_j{+}\lambda_j)^{-}\Big)D_1f_j(x(r),r)\ {-}\delta\Big(r{-}(\tau_j{+}\lambda_j)^{+}\Big)D_1f_1(x(r),r)\Big]z_i^1(r)$$

$$+\theta_i^1\rho(r)^T\Big[\delta\Big(r{-}(\tau_i{+}\lambda_i)^{-}\Big)D_1f_i(x(r),r)\ {-}\delta\Big(r{-}(\tau_i{+}\lambda_i)^{+}\Big)D_1f_1(x(r),r)\Big]z_j^2(r)\mathrm{d}r.$$

Integrating over the $\delta$-functions picks out the times for which the $\delta$-functions' arguments are zero:

$$\begin{aligned}&{=}\theta_j^2\rho\Big((\tau_j{+}\lambda_j)^{-}\Big)D_1f_j\Big(x(\tau_j{+}\lambda_j)^{-},(\tau_j{+}\lambda_j)^{-}\Big)z_i^1\Big((\tau_j{+}\lambda_j)^{-}\Big)\\&\quad{-}\theta_j^2\rho\Big((\tau_j{+}\lambda_j)^{+}\Big)D_1f_1\Big(x(\tau_j{+}\lambda_j)^{+},(\tau_j{+}\lambda_j)^{+}\Big)z_i^1\Big((\tau_j{+}\lambda_j)^{+}\Big)\\&\quad{+}\theta_i^1\rho\Big((\tau_i{+}\lambda_i)^{-}\Big)^TD_1f_i\Big(x(\tau_i{+}\lambda_i)^{-},(\tau_i{+}\lambda_i)^{-}\Big)z_j^2\Big((\tau_i{+}\lambda_i)^{-}\Big)\\&\quad{-}\theta_i^1\rho\Big((\tau_i{+}\lambda_i)^{+}\Big)^TD_1f_1\Big(x(\tau_i{+}\lambda_i)^{+},(\tau_i{+}\lambda_i)^{+}\Big)z_j^2\Big((\tau_i{+}\lambda_i)^{+}\Big).\end{aligned}$$

The indexes $i$ and $j$ relate in three possible ways: $i{<}j$, $i{=}j$, or $i{>}j$. The first and last case are the same, which is based on the fact that partial derivatives (with respect to perturbations indexed with $i$ and $j$) commute.

Recall that $\tau$ is a set of monotonically increasing times. Therefore, if $i{>}j$, then $\tau_i{+}\lambda_i{>}\tau_j{+}\lambda_j$. Given (A.8), $z_k^{\cdot}(t)$ is non-zero only after time $t{=}(\tau_k{+}\lambda_k)^{-}$. In other words, the state does not change up until the first injected control and so the state perturbation $z$ will be zero for all times prior to the perturbations to the control duration. Consequently, because $t_j{+}\lambda_j{<}t_i{+}\lambda_i$, given that $i{>}j$ and so $z_i^1(\tau_j{+}\lambda_j){=}0$. Therefore, the first two terms of $P_{2_{ij}}$ are zero and

$$\begin{aligned}
&\overset{i\geq j}{=}\theta_i^1\rho\Big((\tau_i{+}\lambda_i)^-\Big)^T D_1 f_i\Big(x(\tau_i{+}\lambda_i)^-,(\tau_i{+}\lambda_i)^-\Big)\\
&\quad\Phi\Big((\tau_i{+}\lambda_i)^-,(\tau_j{+}\lambda_j)^-\Big)\Big[f_j\Big(x(\tau_j{+}\lambda_j)^-,(\tau_j{+}\lambda_j)^-\Big){-}f_1\Big(x(\tau_j{+}\lambda_j)^+,(\tau_j{+}\lambda_j)^+\Big)\Big]\theta_j^2\\
&\quad-\theta_i^1\rho\Big((\tau_i{+}\lambda_i)^+\Big)^T D_1 f_1\Big(x(\tau_i{+}\lambda_i)^+,(\tau_i{+}\lambda_i)^+\Big)\\
&\quad\Phi\Big((\tau_i{+}\lambda_i)^-,(\tau_j{+}\lambda_j)^-\Big)\Big[f_j\Big(x(\tau_j{+}\lambda_j)^-,(\tau_j{+}\lambda_j)^-\Big){-}f_1\Big(x(\tau_j{+}\lambda_j)^+,(\tau_j{+}\lambda_j)^+\Big)\Big]\theta_j^2.
\end{aligned}$$

Omitting the no longer useful superscripts $+$ and $-$ gives

$$\begin{aligned}
&\overset{i\geq j}{=}\theta_i^1\rho\Big(\tau_i{+}\lambda_i\Big)^T D_1 f_i\Big(x(\tau_i{+}\lambda_i),\tau_i{+}\lambda_i\Big)\\
&\quad\Phi\Big(\tau_i{+}\lambda_i,\tau_j{+}\lambda_j\Big)\Big[f_j\Big(x(\tau_j{+}\lambda_j),\tau_j{+}\lambda_j\Big){-}f_1\Big(x(\tau_j{+}\lambda_j),\tau_j{+}\lambda_j\Big)\Big]\theta_j^2\\
&\quad-\theta_i^1\rho\Big(\tau_i{+}\lambda_i\Big)^T D_1 f_1\Big(x(\tau_i{+}\lambda_i),\tau_i{+}\lambda_i\Big)\\
&\quad\Phi\Big(\tau_i{+}\lambda_i,\tau_j{+}\lambda_j\Big)\Big[f_j\Big(x(\tau_j{+}\lambda_j),\tau_j{+}\lambda_j\Big){-}f_1\Big(x(\tau_j{+}\lambda_j),\tau_j{+}\lambda_j\Big)\Big]\theta_j^2\\
&\overset{i\geq j}{=}\theta_i^1\rho\Big(\tau_i{+}\lambda_i\Big)^T\Big[D_1 f_i\Big(x(\tau_i{+}\lambda_i),\tau_i{+}\lambda_i\Big){-}D_1 f_1\Big(x(\tau_i{+}\lambda_i),\tau_i{+}\lambda_i\Big)\Big]\\
&\quad\Phi\Big(\tau_i{+}\lambda_i,\tau_j{+}\lambda_j\Big)\Big[f_j\Big(x(\tau_j{+}\lambda_j),\tau_j{+}\lambda_j\Big){-}f_1\Big(x(\tau_j{+}\lambda_j),\tau_j{+}\lambda_j\Big)\Big]\theta_j^2.
\end{aligned}$$

Taking the limit $\Lambda\to 0$,

$$\lim_{\Lambda\to 0} P_2 \overset{i\geq j}{=} \rho(\tau_i)^T [D_1 f_i(x(\tau_i),\tau_i) - D_1 f_1(x(\tau_i),\tau_i)] \boldsymbol{\Phi}(\tau_i,\tau_j) [f_j(x(\tau_j),\tau_j) - f_1(x(\tau_j),\tau_j)] \cdot(\theta_i^1,\theta_j^2).$$

Now consider the $i=j$ case. Because $i=j$, the perturbations $\theta^1$ and $\theta^2$ are equivalent—in the sense that they are both perturbations to the same control duration $\lambda_i$—and therefore $z^1(t)$ and $z^2(t)$ are also equivalent. So,

$$P_{2_{ij}} \overset{i=j}{=} 2\theta_i^2 \rho\Big((\tau_i+\lambda_i)^-\Big)^T D_1 f_i\Big(x(\tau_i+\lambda_i)^-,(\tau_i+\lambda_i)^-\Big) z_i^1\Big((\tau_i+\lambda_i)^-\Big) - 2\theta_i^2 \rho\Big((\tau_i+\lambda_i)^+\Big)^T D_1 f_1\Big(x(\tau_i+\lambda_i)^+,(\tau_i+\lambda_i)^+\Big) z_i^1\Big((\tau_i+\lambda_i)^+\Big)$$

Substituting in for $z_i^1(\cdot)$,

$$\begin{aligned}
&= 2\theta_i^2 \rho\Big((\tau_i+\lambda_i)^-\Big)^T D_1 f_i\Big(x(\tau_i+\lambda_i)^-,(\tau_i+\lambda_i)^-\Big) \\
&\quad \int_{T_0}^{(\tau_i+\lambda_i)^-} \boldsymbol{\Phi}\Big((\tau_i+\lambda_i)^-,r\Big)\cdot\Big[\delta\Big(r-(\tau_i+\lambda_i)^-\Big) f_i(x(r),r) - \delta\Big(r-(\tau_i+\lambda_i)^+\Big) f_1(x(r),r)\Big] \mathrm{d}r\, \theta_i^1 \\
&\quad - 2\theta_i^2 \rho\Big((\tau_i+\lambda_i)^+\Big)^T D_1 f_1\Big(x(\tau_i+\lambda_i)^+,(\tau_i+\lambda_i)^+\Big) \\
&\quad \int_{T_0}^{(\tau_i+\lambda_i)^+} \boldsymbol{\Phi}\Big((\tau_i+\lambda_i)^+,r\Big)\cdot\Big[\delta\Big(r-(\tau_i+\lambda_i)^-\Big) f_i(x(r),r) - \delta\Big(r-(\tau_i+\lambda_i)^+\Big) f_1(x(r),r)\Big] \mathrm{d}r\, \theta_i^1.
\end{aligned}$$

This time the arguments of the $\delta$-functions are zero at the upper bounds of their integrals. So,

$$\begin{aligned}
&=2\theta_i^2\rho\Big((\tau_i{+}\lambda_i)^-\Big)^T D_1 f_i\Big(x(\tau_i{+}\lambda_i)^-,(\tau_i{+}\lambda_i)^-\Big)\cdot \\
&\quad\frac{1}{2}\Phi\Big((\tau_i{+}\lambda_i)^-,(\tau_i{+}\lambda_i)^-\Big) f_i\Big(x(\tau_i{+}\lambda_i)^-,(\tau_i{+}\lambda_i)^-\Big)\theta_i^1 \\
&\quad-2\theta_i^2\rho\Big((\tau_i{+}\lambda_i)^+\Big)^T D_1 f_1\Big(x(\tau_i{+}\lambda_i)^+,(\tau_i{+}\lambda_i)^+\Big)\cdot \\
&\quad\Big[\Phi\Big((\tau_i{+}\lambda_i)^+,(\tau_i{+}\lambda_i)^-\Big) f_i\Big(x(\tau_i{+}\lambda_i)^-,(\tau_i{+}\lambda_i)^-\Big) \\
&\quad-\Phi\Big((\tau_i{+}\lambda_i)^+,(\tau_i{+}\lambda_i)^+\Big)\frac{1}{2} f_1\Big(x(\tau_i{+}\lambda_i)^+,(\tau_i{+}\lambda_i)^+\Big)\Big]\theta_i^1.
\end{aligned}$$

Recall that $\Phi((\tau_i{+}\lambda_i)^-,(\tau_i{+}\lambda_i)^-){=}\Phi((\tau_i{+}\lambda_i)^+,(\tau_i{+}\lambda_i)^+){=}I$ and that $\Phi(\cdot,\cdot)$ is a continuous operator, such that $\Phi((\tau_i{+}\lambda_i)^+,(\tau_i{+}\lambda_i)^-){=}I$. Therefore, omitting the no longer helpful $-$ and $+$ super-scripts,

$$\begin{aligned}
&=\rho(\tau_i{+}\lambda_i)^T\Big[D_1 f_i\Big(x(\tau_i{+}\lambda_i),\tau_i{+}\lambda_i\Big) f_i\Big(x(\tau_i{+}\lambda_i),\tau_i{+}\lambda_i\Big) \\
&\quad-2D_1 f_1\Big(x(\tau_i{+}\lambda_i),(\tau_i{+}\lambda_i)\Big) f_i\Big(x(\tau_i{+}\lambda_i),(\tau_i{+}\lambda_i)\Big) \\
&\quad+D_1 f_1\Big(x(\tau_i{+}\lambda_i),(\tau_i{+}\lambda_i)\Big) f_1\Big(x(\tau_i{+}\lambda_i),(\tau_i{+}\lambda_i)\Big)\Big]\cdot(\theta_i^1,\theta_i^2).
\end{aligned}$$

Taking the limit $\Lambda{\to}0$,

$$\begin{aligned}
\lim_{\Lambda\to 0} P_2 &\overset{i=j}{=}\rho(\tau_i)^T[D_1 f_i(x(\tau_i),\tau_i) f_i(x(\tau_i),\tau_i)\ -2D_1 f_1(x(\tau_i),(\tau_i)) f_i(x(\tau_i),(\tau_i)) \\
&\quad+D_1 f_1(x(\tau_i),(\tau_i)) f_1(x(\tau_i),(\tau_i))]\cdot(\theta_i^1,\theta_i^2).
\end{aligned}$$

Finally, $P_3$. Start with $D_2^2 f^k\Big(x(r),\lambda,r\Big)$. For $i=j$,

$$D_2^2 f^k(x(r),\lambda,r)_{ij}=\left(\frac{\partial}{\partial \Lambda_i}\delta\Big(r-(\tau_i+\lambda_i)^-\Big)\right)f_i^k(x(r),r)-\left(\frac{\partial}{\partial \Lambda_i}\delta\Big(r-(\tau_i+\lambda_i)^+\Big)f_1^k(x(r),r)\right),$$

and, for $i\neq j$, $D_2^2 f^k\Big(x(r),\lambda,r\Big)_{ij}=0$. Revert back to matrix representation of $\rho(\cdot)$ and $f(\cdot,\cdot)$. For $i=j$, using chair rule on $D_2^2 f^k\Big(x(r),\Lambda,r\Big)_{ij}$ results in:

$$D_2^2 f^k(x(r),\Lambda,r)_{ij}=-\dot{\delta}\Big(r-(\tau_i+\lambda_i)^-\Big)f_i^k(x(r),r)+\dot{\delta}\Big(r-(\tau_i+\lambda_i)^+\Big)f_1^k(x(r),r).$$

Then,

$$P_3=\int_{T_0}^{T_F}\Big[-\rho(r)^T\dot{\delta}\Big(r-(\tau_i+\lambda_i)^-\Big)f_i(x(r),r)\ +\rho(r)^T\dot{\delta}\Big(r-(\tau_i+\lambda_i)^+\Big)f_1(x(r),r)\Big]\mathrm{d}r\cdot(\theta_i^1,\theta_i^2).$$

Using integration by parts,

$$=\Bigg[-\rho(r)^T\delta\Big(r-(\tau_i+\lambda_i)^-\Big)f_i(x(r),r)\Bigg|_{T_0}^{T_F}+\rho(r)^T\delta\Big(r-(\tau_i+\lambda_i)^+\Big)f_1(x(r),r)\Bigg|_{T_0}^{T_F}$$

$$\int_{T_0}^{T_F}\Big[\dot{\rho}(r)^Tf_i(x(r),r)\ +\rho(r)^TD_1f_i(x(r),r)\dot{x}(t)+\rho(r)^TD_2f_i(x(r),r)\Big]\cdot\delta\Big(r-(\tau_i+\lambda_i)^-\Big)\mathrm{d}r$$

$$-\int_{T_0}^{T_F}\Big[\dot{\rho}(r)^Tf_1(x(r),r)+\rho(r)^TD_1f_1(x(r),r)\dot{x}(t)+\rho(r)^TD_2f_1\Big(x(r),r\Big)\Big]$$

$$\delta\Big(r-(\tau_i+\lambda_i)^+\Big)\mathrm{d}r\Bigg]\cdot(\theta_i^1,\theta_i^2).$$

Integrating over the $\delta$-functions picks out the times for which the $\delta$-functions' arguments are zero:

$$
\begin{aligned}
&=\Big[\dot{\rho}\Big((\tau_i+\lambda_i)^-\Big)f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)-\dot{\rho}\Big((\tau_i+\lambda_i)^+\Big)f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)\\
&\quad+\rho\Big((\tau_i+\lambda_i)^-\Big)^T D_1 f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)\dot{x}\Big((\tau_i+\lambda_i)^-\Big)\\
&\quad-\rho\Big((\tau_i+\lambda_i)^+\Big)^T D_1 f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)\dot{x}\Big((\tau_i+\lambda_i)^+\Big)\\
&\quad+\rho\Big((\tau_i+\lambda_i)^-\Big)^T D_2 f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)\\
&\quad-\rho\Big((\tau_i+\lambda_i)^+\Big)^T D_2 f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)\Big]\cdot(\theta_i^1,\theta_i^2).
\end{aligned}
$$

Using (A.6),

$$
\begin{aligned}
&=\Big[\Big[-\rho\Big((\tau_i+\lambda_i)^-\Big)^T D_1 f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)-D\ell\Big(x\Big((\tau_i+\lambda_i)^-\Big)\Big)\Big]\\
&\quad\cdot f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)\\
&\quad-\Big[-\rho\Big((\tau_i+\lambda_i)^+\Big)^T D_1 f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)-D\ell\Big(x\Big((\tau_i+\lambda_i)^+\Big)\Big)\Big]\\
&\quad\cdot f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),\lambda_i,(\tau_i+\lambda_i)^+\Big)\\
&\quad+\rho\Big((\tau_i+\lambda_i)^-\Big)^T D_1 f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)\\
&\quad-\rho\Big((\tau_i+\lambda_i)^+\Big)^T D_1 f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)\\
&\quad+\rho\Big((\tau_i+\lambda_i)^-\Big)^T D_2 f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)\\
&\quad-\rho\Big((\tau_i+\lambda_i)^+\Big)^T D_2 f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)\Big]\cdot(\theta_i^1,\theta_i^2).
\end{aligned}
$$

Canceling out terms,

$$=\Big[-D\ell\Big(x\Big((\tau_i+\lambda_i)^-\Big)\Big)\Big(f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)\ -f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)\Big)$$

$$+\rho\Big((\tau_i+\lambda_i)^-\Big)^T\Big(D_2f_i\Big(x\Big((\tau_i+\lambda_i)^-\Big),(\tau_i+\lambda_i)^-\Big)\ -D_2f_1\Big(x\Big((\tau_i+\lambda_i)^+\Big),(\tau_i+\lambda_i)^+\Big)\Big)\Big]$$

$$\cdot\Big(\theta_i^1,\theta_i^2\Big).$$

Then, taking $\Lambda\to0$ and omitting the superscripts,

$$\lim_{\Lambda\to0}P_3=[D\ell(x(\tau_i))(f_i(x(\tau_i),\tau_i)-f_1(x(\tau_i),\tau_i))$$

$$+\rho(\tau_i)^T(D_2f_i(x(\tau_i),\tau_i)-D_2f_1(x(\tau_i),\tau_i))\Big]\cdot(\theta_i^1,\theta_i^2).$$

Therefore, for $i\neq j$,

$$\lim_{\Lambda\to0}D^2J=\Big[[f_i(x(\tau_i),\tau_i)-f_1(x(\tau_i),\tau_i)]^T\boldsymbol{\Omega}(\tau_i)$$

$$+\rho(\tau_i)^T[D_1f_i(x(\tau_i),\tau_i)-D_1f_1(x(\tau_i),\tau_i)]\Big]\cdot\boldsymbol{\Phi}(\tau_i,\tau_j)[f_j(x(\tau_j),\tau_j)-f_1(x(\tau_j),\tau_j)]$$

and, for $i=j$,

$$\lim_{\Lambda\to0}D^2J=\Big[f_i\Big(x(\tau_i),\tau_i\Big)-f_1\Big(x(\tau_i),\tau_i\Big)\Big]^T\boldsymbol{\Omega}(\tau_i)\Big[f_i\Big(x(\tau_i),\tau_i\Big)-f_1\Big(x(\tau_i),\tau_i\Big)\Big]$$

$$+\rho(\tau_i)^T\Big[D_1f_i\Big(x(\tau_i),\tau_i\Big)f_i\Big(x(\tau_i),\tau_i\Big)-2D_1f_1\Big(x(\tau_i),\tau_i\Big)f_i\Big(x(\tau_i),\tau_i\Big)$$

$$+D_1f_1\Big(x(\tau_i),\tau_i\Big)f_1\Big(x(\tau_i),\tau_i\Big)+D_2f_i\Big(x(\tau_i),\tau_i\Big)-D_2f_1\Big(x(\tau_i),\tau_i\Big)\Big]$$

$$-D\ell(x(\tau_i))(f_i(x(\tau_i),\tau_i)-f_1(x(\tau_i),\tau_i)).$$

Given dynamics of the form (2.1), the MIH (for $i=j$) takes the form in (5.1).

## A.2. Dependence of the Mode Insertion Hessian on First-Order Lie-Brackets

Proof. The following analysis shows the algebraic dependence of the MIH expression on the first-order Lie brackets $[h_i,h_j]$ and $[g,h_i]$ and proves Proposition 5 if either: 1) $\rho^T[h_i,h_j] \neq 0$ or 2) $\rho^T[g,h_i] \neq 0$, as guaranteed by Proposition 4.

Consider controls such that $u_j = v_i \; \forall \; j,i{\neq}k$ and $v_k{=}0$ and expresses the MIH expression (5.1) as

$$\frac{d^2J}{d\lambda_+^2} = u^T\mathcal{G}u - u_k((D_x l_1)h_k - \rho^T[g,h_k]),$$

where $\mathcal{G}_{ij} = 0 \; \forall \; i,j{\in}[1,M]\backslash\{k\}$, $\mathcal{G}_{ik} = \mathcal{G}_{ki} = \frac{1}{2}[h_i,h_k]$, and $\mathcal{G}_{kk} = h_k^T\Omega h_k + \rho^T D_x h_k{\cdot}h_k$. The matrix $\mathcal{G}$ is shown to be either indefinite or negative semidefinite if there exists a Lie bracket term $[h_i,h_k]$ such that $\rho^T[h_i,h_k] \neq 0$. From Proposition (8), there exist $i,\, j \in [1,M]$ such that either $\rho^T[h_i,h_j] \neq 0$ or $\rho^T[g,h_i] \neq 0$. Let $k \in [1,M]$ be an index chosen such that either $\rho^T[h_i,h_k] \neq 0$ or $\rho^T[g,h_i] \neq 0$ for some $i \in [1,M] \setminus \{k\}$.

I use summation notation to express the MIH as

$$\begin{aligned}
\frac{d^2J}{d\lambda^2} = {} & \Big(\sum_{i=1}^{M} h_i(u_i - v_i)\Big)^T \Omega \sum_{j=1}^{M} h_j(u_j - v_j) \\
& + \rho^T\Big[\frac{\partial g}{\partial x}g + \frac{\partial g}{\partial x}\sum_j^M h_j u_j + \sum_j^M \frac{\partial h_j}{\partial x}u_j g + \sum_j^M \frac{\partial h_j}{\partial x}u_j \sum_j^M h_j u_j + \frac{\partial g}{\partial x}g + \frac{\partial g}{\partial x}\sum_i^M h_i v_i \\
& + \sum_i^M \frac{\partial h_i}{\partial x}v_i g + \sum_i^M \frac{\partial h_i}{\partial x}v_i \sum_i^M h_i v_i - 2\frac{\partial g}{\partial x}g - 2\frac{\partial g}{\partial x}\sum_j^M h_j u_j - 2\sum_i^M \frac{\partial h_i}{\partial x}v_i g - 2\sum_i^M \frac{\partial h_i}{\partial x}v_i \sum_j^M h_j u_j\Big] \\
& - \frac{\partial \ell}{\partial x}\Big(\sum_{i=1}^{M} h_i(u_i - v_i)\Big),
\end{aligned}$$

which can be simplified to

$$= \Big(\sum_{i=1}^{M} h_i(u_i - v_i)\Big)^T \Omega \sum_{j=1}^{M} h_j(u_j - v_j) + \rho^T \Big( -\frac{\partial g}{\partial x}\sum_{j}^{M} h_j u_j + \sum_{j}^{M}\frac{\partial h_j}{\partial x} u_j g + \sum_{j}^{M}\frac{\partial h_j}{\partial x} u_j \sum_{j}^{M} h_j u_j$$
$$+ \frac{\partial g}{\partial x}\sum_{i}^{M} h_i v_i - \sum_{i}^{M}\frac{\partial h_i}{\partial x} v_i g + \sum_{i}^{M}\frac{\partial h_i}{\partial x} v_i \sum_{i}^{M} h_i v_i - 2\sum_{i}^{M}\frac{\partial h_i}{\partial x} v_i \sum_{j}^{M} h_j u_j \Big) - \frac{\partial \ell}{\partial x}\Big(\sum_{i=1}^{M} h_i(u_i - v_i)\Big).$$

Rearranging the expression into quadratic and linear terms in the control input, I rewrite the MIH expression (5.1) as

$$= \Big(\sum_{i=1}^{M} h_i(u_i - v_i)\Big)^T \Omega \sum_{j=1}^{M} h_j(u_j - v_j)$$
$$+ \rho^T \Big( \sum_{j}^{M}\frac{\partial h_j}{\partial x} u_j \sum_{j}^{M} h_j u_j + \sum_{i}^{M}\frac{\partial h_i}{\partial x} v_i \sum_{i}^{M} h_i v_i - 2\sum_{i}^{M}\frac{\partial h_i}{\partial x} v_i \sum_{j}^{M} h_j u_j \Big) - \frac{\partial \ell}{\partial x}\Big(\sum_{i=1}^{M} h_i(u_i - v_i)\Big)$$
$$+ \rho^T \Big( -\frac{\partial g}{\partial x}\sum_{j}^{M} h_j u_j + \sum_{j}^{M}\frac{\partial h_j}{\partial x} u_j g + \frac{\partial g}{\partial x}\sum_{i}^{M} h_i v_i - \sum_{i}^{M}\frac{\partial h_i}{\partial x} v_i g \Big).$$

Further considering controls such that $u_j = v_i \ \forall\ j,\ i \neq k$ and $v_k = 0$,

$$= (h_k u_k)^T \Omega (h_k u_k) + \rho^T \mathcal{D} - \frac{\partial \ell}{\partial x}(h_k u_k) + \rho^T \Big( -\frac{\partial g}{\partial x} h_k u_k + \frac{\partial h_k}{\partial x} u_k g \Big),$$

where $u_k$ is the $k^{\text{th}}$ control input and $\mathcal{D}$ is

$$\begin{aligned}
\mathcal{D} &= \sum_{j}^{M}\frac{\partial h_j}{\partial x}u_j\sum_{j}^{M}h_j u_j+\sum_{i}^{M}\frac{\partial h_i}{\partial x}v_i\sum_{i}^{M}h_i v_i-2\sum_{i}^{M}\frac{\partial h_i}{\partial x}v_i\sum_{j}^{M}h_j u_j\\
&= (\sum_{j\neq k}^{M}\frac{\partial h_j}{\partial x}u_j)\sum_{j\neq k}^{M}h_j u_j+(\frac{\partial h_k}{\partial x}u_k)\sum_{j\neq k}^{M}h_j u_j+(\sum_{j\neq k}^{M}\frac{\partial h_j}{\partial x}u_j)h_k u_k\\
&\quad+\frac{\partial h_k}{\partial x}u_k h_k u_k+(\sum_{i\neq k}^{M}\frac{\partial h_i}{\partial x}v_i)\sum_{i\neq k}^{M}h_i v_i-2(\sum_{i\neq k}^{M}\frac{\partial h_i}{\partial x}v_i)\sum_{j\neq k}^{M}h_j u_j-2(\sum_{i\neq k}^{M}\frac{\partial h_i}{\partial x}v_i)h_k u_k\\
&= u_k\sum_{j\neq k}^{M}\frac{\partial h_k}{\partial x}h_j u_j-u_k(\sum_{j\neq k}^{M}\frac{\partial h_j}{\partial x}u_j h_k)+\frac{\partial h_k}{\partial x}u_k h_k u_k\\
&= u_k\Big[\sum_{j\neq k}^{M}u_j\Big(\frac{\partial h_k}{\partial x}h_j-\frac{\partial h_j}{\partial x}h_k\Big)\Big]+\frac{\partial h_k}{\partial x}u_k h_k u_k\\
&= u_k\Big[\sum_{j\neq k}^{M}u_j[h_j,h_k]\Big]+\frac{\partial h_k}{\partial x}u_k h_k u_k,
\end{aligned}$$

where terms cancel because $u_j = v_i \; \forall \; j, i \neq k.$ I use the property $x^T Ax = x^T\Big(\frac{1}{2}(A+A^T)\Big)x$ and I write $\mathcal{D}$ in a matrix form,

$$=u^T\begin{pmatrix}
0 & \dots & \frac{1}{2}[h_1,h_k] & \dots & 0\\
\vdots & \ddots & \frac{1}{2}[h_2,h_k] & \iddots & \vdots\\
\frac{1}{2}[h_1,h_k] & \frac{1}{2}[h_2,h_k] & \frac{\partial h_k}{\partial x}h_k & \frac{1}{2}[h_{M-1},h_k] & \frac{1}{2}[h_M,h_k]\\
\vdots & \iddots & \frac{1}{2}[h_{M-1},h_k] & \ddots & \vdots\\
0 & \dots & \frac{1}{2}[h_M,h_k] & \dots & 0
\end{pmatrix}u.$$

The dotted entries in the matrix represent zero terms. Combining all terms, the MIH can be written as

$$\frac{d^2J}{d\lambda^2} = u^T\mathcal{G}u - \frac{\partial \ell}{\partial x}(h_k u_k) + \rho^T\Big(-\frac{\partial g}{\partial x}h_k u_k + \frac{\partial h_k}{\partial x}u_k g\Big) \tag{A.9}$$

$$= u^T\mathcal{G}u - u_k\Big(\frac{\partial \ell}{\partial x}h_k - \rho^T[g,h_k]\Big), \tag{A.10}$$

where

$$\mathcal{G} = \begin{pmatrix} 0 & \cdots & \frac{1}{2}\rho^T[h_1,h_k] & \cdots & 0 \\ \vdots & \ddots & \frac{1}{2}\rho^T[h_2,h_k] & \iddots & \vdots \\ \frac{1}{2}\rho^T[h_1,h_k] & \frac{1}{2}\rho^T[h_2,h_k] & \mathcal{C}_1 & \frac{1}{2}\rho^T[h_{M-1},h_k] & \frac{1}{2}\rho^T[h_M,h_k] \\ \vdots & \iddots & \frac{1}{2}\rho^T[h_{M-1},h_k] & \ddots & \vdots \\ 0 & \cdots & \frac{1}{2}\rho^T[h_M,h_k] & \cdots & 0 \end{pmatrix},$$

and $\mathcal{C}_1 = h_k^T\Omega h_k + \rho^T\frac{\partial h_k}{\partial x}h_k$.

Given that $\frac{dJ}{d\lambda_+} = 0$, then, by Proposition 7, $\rho^T h_i = 0\ \forall\ i \in [1,M]$. In addition, by Proposition 5, $\rho \neq 0$ and, by Proposition 8, there exist $i$, $j \in [1,M]$ such that $\rho^T[h_i,h_j] \neq 0$ or $\rho^T[g,h_i] \neq 0$. It is more convenient to consider two cases that capture all possible scenarios: 1) $\rho^T[h_i,h_j] \neq 0$ and 2) $\rho^T[h_i,h_j] = 0$ (which implies $\rho^T[g,h_i] \neq 0$, by Proposition 3).

**Case 1.** $\rho^T[h_i,h_j] \neq 0$.

Let $\mathcal{G}_{[i,j]}$ denote a 2×2 matrix obtained from $\mathcal{G}$ by deleting all but its $i^{\text{th}}$ and $j^{\text{th}}$ row and $i^{\text{th}}$ and $j^{\text{th}}$ column

$$\mathcal{G}_{[i,j]} = \begin{bmatrix} \mathcal{G}_{ii} & \mathcal{G}_{ij} \\ \mathcal{G}_{ji} & \mathcal{G}_{jj} \end{bmatrix},$$

where $\mathcal{G}_{ij} = \mathcal{G}_{ji}$ because $\mathcal{G}$ is symmetric. The principal minors of $\mathcal{G}$ of order 2 are given by $\Delta_2 = \det(\mathcal{G}_{[i,j]}) = \mathcal{G}_{ii}\mathcal{G}_{jj} - \mathcal{G}_{ij}^2\mathcal{G}_{ji}\ \forall\ i \neq j \in [1{,}M]$.

Consider first the diagonal terms of $\mathcal{G}_{[i,j]}$. Note that, because $i \neq j$ and $\mathcal{G}_{ii} = 0\ \forall\ i \neq k$, then either $\mathcal{G}_{ii} = 0$ or $\mathcal{G}_{jj} = 0$. Therefore, $\Delta_2 = -\ \mathcal{G}_{ij}\mathcal{G}_{ji}\ \forall\ i \neq j \in [1{,}M]$. Next, consider the off-diagonal elements. Also note that $\mathcal{G}_{ij} = 0\ \forall\ i,\ j \in [1{,}M] \setminus\{k\}$. Given that $\mathcal{G}_{ik} = \mathcal{G}_{ki} = \frac{1}{2}\rho^T[h_i{,}h_k]\ \forall\ i \in [1{,}M] \setminus \{k\}$, I have $\Delta_2 = 0\ \forall\ i \in [1{,}M] \setminus \{k\}$ and $\Delta_2 = -\ \frac{1}{4}(\rho^T[h_i{,}h_k])^2$, otherwise. I summarize these cases as follows

$$\Delta_2 = \begin{cases} 0 & \forall\ i,\ j \in [1{,}M] \setminus \{k\} \\ -\mathcal{G}_{ij}^2 = -\frac{1}{4}(\rho^T[h_i{,}h_k])^2 \leq 0 & \text{otherwise.} \end{cases}$$

If there exists $i \in [1{,}M]$ such that $\rho^T[h_i{,}h_j] \neq 0$, there is at least one negative second-order principal minor. Therefore, $\mathcal{G}$ is indefinite and so there exist controls $u \in \mathbb{R}^M$ such that $u^T\mathcal{G}u < 0$.

Choose $u \in \mathbb{R}^{\mathbb{M}}$ such that $u^T\mathcal{G}u < 0$ and let $u_k \in \mathbb{R}$ represent the $k^{th}$ element of $u$. If $u_k\Big(\rho^T[g{,}h_k]{-}D_xlh_k\Big) \leq 0$, then

$$u^T\mathcal{G}u < 0 \Longrightarrow u^T\mathcal{G}u{+}u_k\Big(\rho^T[g{,}h_k]{-}D_xlh_k\Big) < 0.$$

Else, if $u_k\Big(\rho^T[g,h_k]-D_xlh_k\Big) > 0$, choose $u' = -u$ so that

$$u'_k\Big(\rho^T[g,h_k]-D_xlh_k\Big)=-u_k\Big(\rho^T[g,h_k]-D_xlh_k\Big) < 0$$

and

$$u'^T\mathcal{G}u'+u'_k\Big(\rho^T[g,h_k]-D_xlh_k\Big)=u^T\mathcal{G}u-u_k\Big(\rho^T[g,h_k]-D_xlh_k\Big) < 0.$$

Therefore, if $\rho^T[h_i,h_j] \neq 0$, there always exists $u \in \mathbb{R}^M$ such that $\frac{d^2J}{d\lambda^2} < 0$.

**Case 2.** $\rho^T[h_i,h_j] = 0$.

If $\rho^T[h_i,h_j] = 0 \;\forall\; i,\, j \in [1,M]$, then, shown in Proposition 8, there exists $i \in [1,M]$ such that $\rho^T[g,h_i] \neq 0$. For $\rho^T[h_i,h_j] = 0$, the MIH becomes

$$u_k^2\Big(h_k^T\Omega h_k+\rho^TD_xh_k\Big)+u_k\Big(\rho^T[g,h_k]-D_xlh_k\Big), \tag{A.11}$$

which is a quadratic expression of the form $ax^2+bx+c$. Quadratic expressions become negative if and only if $a \leq 0$ or $b^2-4ac > 0$. Therefore, (A.11) takes negative values if and only if, for some time $t \in [t_o,t_o+T]$,

(1) $h_k^T\Omega h_k+\rho^TD_xh_k \leq 0$ OR
(2) $\rho^T[g,h_k]-D_xlh_k \neq 0$.

I consider the second condition: $\rho^T[g,h_k]-D_x l h_k \neq 0$. Because $\frac{dJ}{d\lambda_+} = 0 \; \forall \; u \in \mathbb{R}^M$ and $\forall \; t \in [t_o,t_o+T]$, then

$$\begin{aligned}
\frac{dJ}{d\lambda_+} = 0 &\Longrightarrow \rho^T h_i = 0 \; \forall \; i \in [1,M], \; \forall \; t \in [t_o, \; t_o+T] \\
&\Longrightarrow \rho^T h_k = 0 \; \forall \; t \in [t_o, \; t_o+T] \\
&\Longrightarrow \rho^T h_k = 0 \text{ for } t = t_o+T \text{ AND } \dot{\rho}^T h_k = 0 \; \forall \; t \in [t_o, \; t_o+T] \\
&\Longrightarrow D_x m h_k = 0 \text{ for } t = t_o+T \text{ AND } (-D_x l - \rho^T D_x f_2) h_k = 0 \; \forall \; t \in [t_o, \; t_o+T] \\
&\Longrightarrow (x-x_d)^T P_1 h_k = 0 \text{ for } t = t_o+T \\
&\qquad \text{AND } (-(x-x_d)^T Q - \rho^T D_x f_2) h_k = 0 \; \forall \; t \in [t_o, \; t_o+T].
\end{aligned}$$

Consider positive-definite weight matrices $Q = \delta P_1 \succ 0$, where $\delta$ is a scale factor. Then,

$$\begin{aligned}
(x-x_d)^T P_1 h_k = 0|_{t_o+T} &\Leftrightarrow (x-x_d)^T \delta P_1 h_k = 0|_{t_o+T} \\
&\Leftrightarrow (x-x_d)^T Q h_k = 0|_{t_o+T},
\end{aligned}$$

and

$$\begin{aligned}
\frac{dJ}{d\lambda_+} = 0 &\Longrightarrow D_x m h_k = 0 \\
&\Leftrightarrow D_x l h_k = 0 \text{ AND } \rho^T D_x f_2 h_k = 0.
\end{aligned}$$

Then, $\rho^T[g,h_k]-D_x l h_k = \rho^T[g,h_k] \neq 0$. Therefore, there exist control solutions $u \in \mathbb{R}^M$ such that the MIH expression becomes negative. $\square$

## A.3. Control Solutions based on the Mode Insertion Gradient and Hessian

In the following derivation, I treat the first- and second-order mode insertion gradient terms separately.

Associate $f_1$ with default control $v$ and $f_2$ with injected control $u$, such that

$$f_1 \triangleq f(x(t),v(t)) = g(x(t)) + h(x(t))v(t)$$

$$f_2 \triangleq f(x(t),u(t)) = g(x(t)) + h(x(t))u(t)$$

For simplicity, I drop the arguments as necessary. For the mode insertion gradient, the update step is straightforward

$$\frac{\partial}{\partial u}\frac{dJ}{d\lambda_+} = \frac{\partial}{\partial u}\rho^T(f_2 - f_1) = \frac{\partial}{\partial u}\rho^T h(u-v) = \rho^T h, \tag{A.12}$$

$$\frac{\partial^2}{\partial u^2}\frac{dJ}{d\lambda_+} = 0. \tag{A.13}$$

The update step on the MIH is more complicated and so I divide the MIH expression into three parts

$$\frac{d^2 J}{d\lambda_+^2} = \mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3,$$

where the terms $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ are given by the following set of equations

$$\mathcal{A}_1 = (f_2 - f_1)^T \Omega (f_2 - f_1)$$

$$\mathcal{A}_2 = \rho^T (D_x f_2 \cdot f_2 + D_x f_1 \cdot f_1 - 2 D_x f_1 \cdot f_2)$$

$$\mathcal{A}_3 = -D_x l \cdot (f_2 - f_1).$$

Let $l_2 = \frac{d^2 J}{d\lambda_+^2}$. Using the Gâteux derivative,

$$\frac{\partial l_2}{\partial u} = \frac{\partial l_2(u+\epsilon\eta)}{\partial \epsilon}\bigg|_{\epsilon=0} = \frac{\partial \mathcal{A}_1(\cdot)}{\partial \epsilon} + \frac{\partial \mathcal{A}_2(\cdot)}{\partial \epsilon} + \frac{\partial \mathcal{A}_3(\cdot)}{\partial \epsilon}\bigg|_{\epsilon=0}.$$

Then,

$$\begin{aligned}
\frac{\partial \mathcal{A}_1}{\partial u} &= \frac{\partial}{\partial \epsilon} \mathcal{A}_1(u+\epsilon\eta)\bigg|_{\epsilon=0} \\
&= \frac{\partial}{\partial \epsilon}([h\Big((u+\epsilon\eta)-v\Big)]^T \Omega\ [h\Big((u+\epsilon\eta)-v\Big)])\bigg|_{\epsilon=0} \\
&= (h\eta)^T \Omega\ (h \cdot (u+\epsilon\eta-v)) + (h(u+\epsilon\eta-v))^T \Omega\ h\eta\bigg|_{\epsilon=0} \\
&= \eta^T h^T \Omega\ hu - \eta^T h^T \Omega\ hv + u^T h^T \Omega\ h\eta - v^T h^T \Omega\ h\eta \\
&= \Big(u^T h^T \Big(\Omega^T + \Omega\Big) h - v^T h^T \Big(\Omega^T + \Omega\Big) h\Big)\eta
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{A}_2}{\partial u} &= \frac{\partial}{\partial \epsilon}\mathcal{A}_2(u+\epsilon\eta)\Big|_{\epsilon=0} \\
&= \frac{\partial}{\partial \epsilon}\rho^T(D_xf_1\cdot f_1 - 2\ D_xf_1\cdot f_2 + D_xf_2\cdot f_2)\Big|_{\epsilon=0} \\
&= \frac{\partial}{\partial \epsilon}\rho^T(D_x(g+hv)\cdot(g+hv) - 2D_x(g+hv)(g+h(u+\epsilon\eta)) \\
&\quad + D_x(g+h(u+\epsilon\eta))(g+h(u+\epsilon\eta)))\Big|_{\epsilon=0} \\
&= \rho^T(-2\ D_x(g+hv)h\eta + D_x(h\eta)\cdot(g+hu) + D_x(g+hu)\cdot h\eta) \\
&= \rho^T(-2D_x(g+hv)h\eta + D_x(h\eta)g + D_x(h\eta)hu + D_xgh\eta + D_x(hu)h\eta) \\
&= \rho^T(-D_xgh\eta - 2D_x(hv)h\eta + D_x(h\ \eta)\cdot g + D_x(h\ \eta)\cdot hu + D_x(hu)\cdot h\eta) \\
&= \rho^T\Big(-D_xg\cdot h\eta - 2D_x(\sum_{k=1}^{m}h_ku_{d_k})\cdot h\eta + D_x(\sum_{k=1}^{m}h_k\eta_k)\cdot g + D_x(\sum_{k=1}^{m}h_k\eta_k)hu \\
&\quad + D_x(\sum_{k=1}^{m}h_ku_{2_k})\cdot h\eta\Big) \\
&= -\rho^TD_xgh\eta - 2v^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)h\eta + \eta^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)g + \eta^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)hu \\
&\quad + u^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)h\eta \\
&= -\rho^TD_xgh\eta - 2v^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)\cdot h\eta + g^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T\eta + u^Th^T\cdot(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T\eta \\
&\quad + u^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)\cdot h\eta
\end{aligned}$$

$$\begin{aligned}
&= \Big[-\rho^TD_xg\cdot h - 2v^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)\cdot h + g^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T + u^Th^T\cdot(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T \\
&\quad + u^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)\cdot h\Big]\cdot\eta.
\end{aligned}$$

Last,

$$\begin{aligned}\frac{\partial\mathcal{A}_3}{\partial u}&=\frac{\partial}{\partial\epsilon}\mathcal{A}_3(u+\epsilon\eta)\Big|_{\epsilon=0}\\&=-\frac{\partial l}{\partial x}\frac{\partial}{\partial\epsilon}\Big(g+h(u+\epsilon\eta)-g-hv\Big)\Big|_{\epsilon=0}\\&=-\frac{\partial l}{\partial x}h\eta\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{\partial l_2}{\partial u}=&u^T\Big[h^T\Big(\Omega^T+\Omega\Big)h+h^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T+(\sum_{k=1}^{n}(D_xh_k)\rho_k)h\Big]\\&-v^T\Big[h^T\Big(\Omega^T+\Omega\Big)h+2(\sum_{k=1}^{n}(D_xh_k)\rho_k)h\Big]-\rho^TD_xg\cdot h+g^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T-\frac{\partial l}{\partial x}h.\end{aligned}$$

Solving for the minimizer, $\frac{\partial l_2}{\partial u}^T=0$, I get

(A.14)

$$\begin{aligned}0=&\Big[h^T\Big(\Omega^T+\Omega\Big)h+h^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T+(\sum_{k=1}^{n}(D_xh_k)\rho_k)h\Big]u\\&-\Big[h^T\Big(\Omega^T+\Omega\Big)h+2h^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T\Big]v-D_xg^T\rho h-(\sum_{k=1}^{n}(D_xh_k)\rho_k)g+h^TD_xl^T\end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned}u=&\Big[h^T\Big(\Omega^T+\Omega\Big)h+h^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T+(\sum_{k=1}^{n}(D_xh_k)\rho_k)h\Big]^{-1}\Big[h^T\Big(\Omega^T+\Omega\Big)h\\&+2h^T(\sum_{k=1}^{n}(D_xh_k)\rho_k)^T\Big]v+D_xg^T\rho h-(\sum_{k=1}^{n}(D_xh_k)\rho_k)g+h^TD_xl^T\end{aligned}$$

The terms shown in (A.12) and (A.14), together with a penalty term for the control, are the gradient and Hessian terms used for the Newton update step that appears in (5.6).

## A.4. Global Error for Taylor-Based Koopman Operators

In each time step, there is error induced in the updated function. Let $e_k^{(m)}$ indicate the deviation from the accurate value of the $f_k^{(m)}$ function at time $t_k$, where $m \in \mathbb{Z} \cap [0,n]$. That is,

$$\begin{aligned} e_k &= \tilde{f}_k - f_k \\ e'_k &= \tilde{f}'_k - f'_k \\ &\vdots \\ e_k^{(n)} &= \tilde{f}_k^{(n)} - f_k^{(n)} \end{aligned} \tag{A.15}$$

Next, consider how previous errors accumulate in the prediction of a function:

$$\begin{aligned} \tilde{f}_{k+1} &= \tilde{f}_k + \tilde{f}'_k \cdot \Delta t + \tilde{f}''_k \cdot \frac{\Delta t^2}{2!} + \cdots + \tilde{f}_k^{(n)} \frac{\Delta t^n}{n!} \\ &= (f_k + e_k) + (f'_k + e'_k) \cdot \Delta t + (f''_k + e''_k) \cdot \frac{\Delta t^2}{2!} + \cdots + (f_k^{(n)} + e_k^{(n)}) \frac{\Delta t^n}{n!} \\ &= f_k + f'_k \cdot \Delta t + f''_k \frac{\Delta t^2}{2!} + \cdots + f_k^{(n)} \frac{\Delta t^n}{n!} + \underbrace{e_k + e'_k \cdot \Delta t + e''_k \cdot \frac{\Delta t^2}{2!} + \cdots + e_k^{(n)} \frac{\Delta t^n}{n!}}_{\textit{error terms}}, \end{aligned} \tag{A.16}$$

such that

$$e_{k+1} = e_k + e'_k \cdot \Delta t + e''_k \cdot \frac{\Delta t^2}{2!} + \cdots + e_k^{(n)} \frac{\Delta t^n}{n!} + \frac{\Delta t^{n+1}}{(n+1)!} f_{k,k+1}^{(n+1)}, \tag{A.17}$$

where the last error term is from Lagrange's remainder formula and is added at each step. Remember, $f_{k,k+1}^{(n)}$ is the $n$th derivative of a function evaluated at some $t \in [t_k, t_{k+1}]$.

Similarly,

$$e_k = e_{k-1} + e'_{k-1} \cdot \Delta t + e''_{k-1} \cdot \frac{\Delta t^2}{2!} + \cdots + e^{(n)}_{k-1} \frac{\Delta t^n}{n!} + \frac{\Delta t^{n+1}}{(n+1)!} f^{(n+1)}_{k-2,k-1} \tag{A.18}$$

$$\vdots \tag{A.19}$$

$$e_3 = e_2 + e'_2 \cdot \Delta t + e''_2 \cdot \frac{\Delta t^2}{2!} + \cdots + e^{(n)}_2 \frac{\Delta t^n}{n!} + \frac{\Delta t^{n+1}}{(n+1)!} f^{(n+1)}_{2,3} \tag{A.20}$$

$$e_2 = e_1 + e'_1 \cdot \Delta t + e''_1 \cdot \frac{\Delta t^2}{2!} + \cdots + e^{(n)}_1 \frac{\Delta t^n}{n!} + \frac{\Delta t^{n+1}}{(n+1)!} f^{(n+1)}_{1,2}, \tag{A.21}$$

where $e_1 = f^{(n+1)}_{0,1} \frac{\Delta t^{n+1}}{(n+1)!}$; in the first iteration, I assume the knowledge of the exact values of all derivatives up to order $n$, such that the only numerical error comes from the inaccuracy of the Taylor series expansion. Without loss of generality, the functions are assumed to be known exactly at $k=0$. That is, $e_0=0, e'_0=0, \ldots, e^{(n)}_0=0$. Thus, I can express the error of a function $f$ as

$$e_k = \left(\sum_{i=1}^{k-1} e'_i \cdot \Delta t + e''_i \cdot \frac{\Delta t^2}{2!} + \cdots + e^{(n)}_i \frac{\Delta t^n}{n!} + f^{(n+1)}_{i,i+1} \frac{\Delta t^{n+1}}{(n)!}\right) + f^{(n+1)}_{0,1} \frac{\Delta t^{n+1}}{(n+1)!} \tag{A.22}$$

Similarly, for the errors associated with the higher-order terms, I have

$$\begin{aligned}
e'_k &= \left(\sum_{i=1}^{k-1} e''_i \cdot \Delta t + e''_i \cdot \frac{\Delta t^2}{2!} + \cdots + e^{(n)}_i \frac{\Delta t^{n-1}}{(n-1)!} + f^{(n+1)}_{i,i+1} \frac{\Delta t^n}{(n)!}\right) + f^{(n+1)}_{0,1} \frac{\Delta t^n}{(n)!} \\
&\vdots \\
e^{(n-1)}_k &= \left(\sum_{i=1}^{k-1} e^{(n)}_i \cdot \Delta t + f^{(n+1)}_{i,i+1} \frac{\Delta t^{(n+1)-(n-1)}}{((n+1)-(n-1))!}\right) + f^{(n+1)}_{0,1} \frac{\Delta t^{(n+1)-(n-1)}}{((n+1)-(n-1))!} \\
e^{(n)}_k &= \left(\sum_{i=1}^{k-1} f^{(n+1)}_{i,i+1} \frac{\Delta t^{n+1-n}}{(n+1-n)!}\right) + f^{(n+1)}_{0,1} \frac{\Delta t^{n+1-n}}{(n+1-n)!}
\end{aligned} \tag{A.23}$$

In general, the error of the $p$-th derivative of a function $f$ at time $t_k$ is given by

$$e_k^{(p)} = \left( \sum_{i=1}^{k-1} \left( \left( \sum_{j=1}^{n-p} e_i^{(j)} \frac{\Delta t^j}{j!} \right) + f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1-p}}{(n+1-p)!} \right) \right) + f_{0,1}^{(n+1)} \frac{\Delta t^{n+1-p}}{(n+1-p)!}, \tag{A.24}$$

which can be simplified to

$$e_k^{(p)} = \sum_{i=1}^{k-1} \sum_{j=1}^{n-p} e_i^{(j)} \frac{\Delta t^j}{j!} + \sum_{i=0}^{k-1} f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1-p}}{(n+1-p)!}, \tag{A.25}$$

which is split into the error from the derivative inaccuracies and the error induced by the Taylor series expansion at each step. Note that $n$ is the number of derivatives used to propagate $f$, where $p \in \mathbb{Z} \cap [0,n]$ indicates the order of the derivative of a function for which the error is calculated ($p=0$ refers to the original function $f$).

## A.5. Global Error Bounds for Taylor-Based Koopman Operator

First, consider the error in $f$ when no derivative basis functions are used, that is, $n=0$. Then,

$$e_k = \sum_{i=0}^{k-1} f_{i,i+1}^{(1)} \frac{\Delta t^1}{1!} \tag{A.26}$$

$$|e_k| = \left| \sum_{i=0}^{k-1} f_{i,i+1}^{(1)} \frac{\Delta t^1}{1!} \right| \tag{A.27}$$

$$|e_k| \leq \frac{\Delta t^1}{1!} \sum_{i=0}^{k-1} |f_{i,i+1}^{(1)}| \tag{A.28}$$

To further simplify the analysis, I can assume a maximum value of $f^{(1)}$, $|f^{(1)}_{i,i+1}|\leq|f^{(1)}_{max}|$ for all $i\in\mathbb{Z}\cap[0,k-1]$. Then,

$$|e_k|\leq\frac{\Delta t^1}{1!}\sum_{i=0}^{k-1}|f^{(1)}_{max}| \tag{A.29}$$

$$|e_k|\leq k\cdot\Delta t\cdot|f^{(1)}_{max}|=T\cdot|f^{(1)}_{max}|, \tag{A.30}$$

where $T\triangleq k\cdot\Delta t$ is the time window that I approximate the function over.

Similarly, I compute the error bound for $n=1$ (one derivative of $f$ in the basis functions). Then,

$$e_k=\sum_{i=1}^{k-1}e'_i\Delta t+\sum_{i=0}^{k-1}f^{(2)}_{i,i+1}\frac{\Delta t^2}{2!}, \tag{A.31}$$

where $e'_i$ is given by (A.26). Note that $f^{(1)}$ becomes $f^{(2)}$; what was $e_i$ is now $e'_i$ and the first-order derivative of $e'_i$ is the second-order derivative of $e_i$. Thus,

$$e_k=\sum_{i=1}^{k-1}\Big(\sum_{j=0}^{i-1}f^{(2)}_{j,j+1}\frac{\Delta t^1}{1!}\Big)\Delta t+\sum_{i=0}^{k-1}f^{(2)}_{i,i+1}\frac{\Delta t^2}{2!} \tag{A.32}$$

$$|e_k|\leq\sum_{i=1}^{k-1}\Big(\sum_{j=0}^{i-1}|f^{(2)}_{j,j+1}|\frac{\Delta t^1}{1!}\Big)\Delta t+\frac{\Delta t^2}{2!}\sum_{i=0}^{k-1}|f^{(2)}_{i,i+1}| \tag{A.33}$$

To further simplify things, I again use $|f^{(2)}_{i,i+1}|\leq|f^{(2)}_{max}|$ for all $i\in\mathbb{Z}\cap[0,k-1]$, such that

$$|e_k|\leq\sum_{i=1}^{k-1}\Big(\sum_{j=0}^{i-1}|f^{(2)}_{max}|\frac{\Delta t^1}{1!}\Big)\Delta t+\frac{\Delta t^2}{2!}\sum_{i=0}^{k-1}|f^{(2)}_{max}| \tag{A.34}$$

$$|e_k|\leq\sum_{i=1}^{k-1}i|f^{(2)}_{max}|\Delta t^2+k\cdot\frac{\Delta t^2}{2}|f^{(2)}_{max}| \tag{A.35}$$

$$|e_k|\leq|f^{(2)}_{max}|\Delta t^2\sum_{i=1}^{k-1}i+k\cdot\frac{\Delta t^2}{2}|f^{(2)}_{max}|. \tag{A.36}$$

Using the property $\sum_{i=0}^{n}i=\sum_{i=1}^{n}i=\frac{n(n+1)}{2}$,

$$|e_k|\leq|f^{(2)}_{max}|\Delta t^2\frac{(k-1)k}{2}+k\cdot\frac{\Delta t^2}{2}|f^{(2)}_{max}| \tag{A.37}$$

$$|e_k|\leq|f^{(2)}_{max}|\Delta t^2\frac{k^2-k}{2}+k\cdot\frac{\Delta t^2}{2}|f^{(2)}_{max}| \tag{A.38}$$

$$|e_k|\leq\frac{(k\cdot\Delta t)^2}{2}|f^{(2)}_{max}|=\frac{T^2}{2}|f^{(2)}_{max}|. \tag{A.39}$$

From $n=0$ and $n=1$, I notice a pattern in the error bound expression. Using proof by induction, I next show that the error bound is given by

$$|e_k|\leq\frac{T^{n+1}}{(n+1)!}|f^{(n+1)}_{max}|. \tag{A.40}$$

### A.5.1. Base Case: $n=0$

From (A.30), it is true that, for $n=0$,

$$|e_k|\leq T\cdot|f^{(1)}_{max}|. \tag{A.41}$$

### A.5.2. Induction Step:

Assuming that the relationship holds for $n{-}1$, I show it is also true for $n$. For the case of using basis functions with derivatives up to order $n$, the error in the derivative function of order $p{=}0$ is, using (A.25), given by

$$e_k=\sum_{i=1}^{k-1}\sum_{j=1}^{n}e_i^{(j)}\frac{\Delta t^j}{j!}+\sum_{i=0}^{k-1}f_{i,i+1}^{(n+1)}\frac{\Delta t^{n+1}}{(n+1)!}. \tag{A.42}$$

Taking the absolute value of the error,

$$|e_k|=|\sum_{i=1}^{k-1}\sum_{j=1}^{n}e_i^{(j)}\frac{\Delta t^j}{j!}+\sum_{i=0}^{k-1}f_{i,i+1}^{(n+1)}\frac{\Delta t^{n+1}}{(n+1)!}| \tag{A.43}$$

$$\leq\sum_{i=1}^{k-1}\sum_{j=1}^{n}|e_i^{(j)}|\frac{\Delta t^j}{j!}+\sum_{i=0}^{k-1}|f_{i,i+1}^{(n+1)}|\frac{\Delta t^{n+1}}{(n+1)!}, \tag{A.44}$$

where $\frac{\Delta t^j}{j!}$ and $\frac{\Delta t^{n+1}}{(n+1)!}$ are non-negative. Then, I use the relationship to substitute for the terms $|e_i^{(j)}|$. Note that $|e_i^{(1)}|$ is equivalent to $|e_k|$ using $n{-}1$ derivatives. Similarly, each term $|e_i^{(j)}|$ for $j{\in}\mathbb{Z}{\cap}[1,n]$ is equivalent to $|e_k|$ for $n{-}j$ derivatives. Thus, I use the relationship that holds for up to $n{-}1$ derivatives, such that

$$\begin{aligned}|e_k|&\leq\sum_{i=1}^{k-1}\sum_{j=1}^{n}|\frac{(i\Delta t)^{n+1-j}}{(n+1-j)!}f_{max}^{(n+1)}|\frac{\Delta t^j}{j!}+\sum_{i=0}^{k-1}|f_{i,i+1}^{(n+1)}|\frac{\Delta t^{n+1}}{(n+1)!}\\&=|f_{max}^{(n+1)}|\Delta t^{n+1}\sum_{i=1}^{k-1}\sum_{j=1}^{n}\frac{i^{n+1-j}}{(n+1-j)!j!}+|f_{max}^{(n+1)}|\sum_{i=0}^{k-1}\frac{\Delta t^{n+1}}{(n+1)!},\end{aligned} \tag{A.45}$$

where $|f_{max}^{(n+1)}|{\geq}|f_{i,i+1}^{(n+1)}|$. Next, I use $j'{=}n{+}1{-}j$ to simplify the inner sum term

$$\sum_{j=1}^{n}\frac{i^{n+1-j}}{(n+1-j)!j!}=\sum_{j'=n}^{1}\frac{i^{j'}}{(j')!(n+1-j')!}, \tag{A.46}$$

which can be rewritten for $j$ and the summation order can also be reversed such that

$$|e_k| \leq |f_{max}^{(n+1)}|\Delta t^{n+1} \sum_{i=1}^{k-1}\sum_{j=1}^{n} \frac{i^j}{(n+1-j)!j!} + |f_{max}^{(n+1)}| \sum_{i=0}^{k-1} \frac{\Delta t^{n+1}}{(n+1)!}. \tag{A.47}$$

Further simplifying

$$|f_{max}^{(n+1)}| \sum_{i=0}^{k-1} \frac{\Delta t^{n+1}}{(n+1)!} = k|f_{max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!}, \tag{A.48}$$

gives

$$\begin{aligned} |e_k| &\leq |f_{max}^{(n+1)}|\Delta t^{n+1} \sum_{i=1}^{k-1}\sum_{j=1}^{n} \frac{i^j}{(n+1-j)!j!} + k|f_{max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!} \\ &= \left[ \sum_{i=1}^{k-1}\sum_{j=1}^{n} \frac{i^j}{(n+1-j)!j!} + \frac{k}{(n+1)!} \right] |f_{max}^{(n+1)}|\Delta t^{n+1}. \end{aligned} \tag{A.49}$$

Using the binomial coefficient

$$\frac{a!}{(a-b)!b!} = \binom{a}{b}, \tag{A.50}$$

to rewrite the term $(n+1-j)!j!$,

$$|e_k| \leq \left[ \sum_{i=1}^{k-1}\sum_{j=1}^{n} \frac{i^j}{(n+1)!}\binom{n+1}{j} + \frac{k}{(n+1)!} \right] |f_{max}^{(n+1)}|\Delta t^{n+1}. \tag{A.51}$$

Switching the summation order,

$$|e_k| \leq \left[ \sum_{j=1}^{n} \binom{n+1}{j} \sum_{i=1}^{k-1} i^j + k \right] |f_{max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!}. \tag{A.52}$$

To use Pascal's identity [**235**]:

$$\sum_{p=0}^{k}\binom{k+1}{p}\sum_{j=1}^{n}j^{p}=(n+1)^{k+1}-1, \tag{A.53}$$

I rewrite the error bound as

$$|e_k|\leq\left[\sum_{j=0}^{n}\binom{n+1}{j}\sum_{i=1}^{k-1}i^{j}-\binom{n+1}{0}\sum_{i=1}^{k-1}i^{0}+k\right]|f_{max}^{(n+1)}|\frac{\Delta t^{n+1}}{(n+1)!}, \tag{A.54}$$

such that

$$\begin{aligned}|e_k|&\leq\Big[(k^{n+1}-1)-(k-1)+k\Big]|f_{max}^{(n+1)}|\frac{\Delta t^{n+1}}{(n+1)!}\\&=k^{n+1}|f_{max}^{(n+1)}|\frac{\Delta t^{n+1}}{(n+1)!}\\&=\frac{T^{n+1}}{(n+1)!}|f_{max}^{(n+1)}|,\end{aligned} \tag{A.55}$$

where $T=k\cdot\Delta t$. Therefore,

$$|e_k|\leq\frac{T^{n+1}}{(n+1)!}|f_{max}^{(n+1)}| \text{ for all } n\in\mathbb{Z}^{\geq 0}. \tag{A.56}$$

## A.6. Incremental Update of Koopman Operator

Consider $P$ measurements used to last update the Koopman operator and $\Delta P$ new measurements. I incrementally compute a Koopman operator using all $P_{total}=P+\Delta P$ measurements as follows. The Koopman operator is computed using (7.5), which I split the expression into past and new measurements, such that

$$\begin{aligned}\mathcal{A}_{new} &= \frac{1}{P_{total}} \sum_{k=0}^{P_{total}-1} \Psi(s_{k+1},u_{k+1})\Psi(s_k,u_k)^T \\ &= \frac{1}{P_{total}} (\sum_{k=0}^{P-1} \Psi(s_{k+1},u_{k+1})\Psi(s_k,u_k)^T + \sum_{k=P}^{P_{total}-1} \Psi(s_{k+1},u_{k+1})\Psi(s_k,u_k)^T)\end{aligned} \tag{A.57}$$

and, using

$$\mathcal{A} = \frac{1}{P} \sum_{k=0}^{P-1} \Psi(s_{k+1},u_{k+1})\Psi(s_k,u_k)^T, \tag{A.58}$$

$\mathcal{A}_{new}$ can be written as

$$\mathcal{A}_{new} = \frac{1}{P_{total}} (P\mathcal{A} + \sum_{k=P}^{P_{total}-1} \Psi(s_{k+1},u_{k+1})\Psi(s_k,u_k)^T. \tag{A.59}$$

Similarly,

$$\mathcal{G}_{new} = \frac{1}{P_{total}} (\mathcal{G}P + \sum_{k=P}^{P_{total}-1} \Psi(s_k,u_k)\Psi(s_k,u_k)^T). \tag{A.60}$$

## A.7. Gradient Descent for SOC algorithm

Here I derive the gradient descents for the SOC algorithm, shown in (9.7). Let

$$f = \frac{1}{2} \|Y - S^{-1}OCSX - BU\|_F^2$$

Using the identity

$$\|X\|_F^2 = \mathrm{Tr}(X^T X)$$

and expanding the product $(Y-S^{-1}OCSX-BU)^T(Y-S^{-1}OCSX-BU)$, I rewrite the optimization problem as

$$\begin{aligned}f=&\frac{1}{2}\mathrm{Tr}(Y^TY-Y^TS^{-1}OCSX-Y^TBU\\&-X^TS^TC^TO^TS^{-T}Y+X^TS^TC^TO^TS^{-T}S^{-1}OCSX+X^TS^TC^TO^TS^{-T}BU\\&-U^TB^TY+U^TB^TS^{-1}OCSX+U^TB^TBU).\end{aligned}$$

To calculate the gradients of $f$, I use the identities

$$\frac{\partial}{\partial X}\mathrm{Tr}(AXB)=A^TB^T \tag{A.12}$$

$$\frac{\partial}{\partial X}\mathrm{Tr}(AX^TB)=BA \tag{A.13}$$

$$\frac{\partial}{\partial X}\mathrm{Tr}(B^TX^TCXB)=(C^T+C)XBB^T. \tag{A.14}$$

The gradient of $f$ with respect to $C$ is

$$\begin{aligned}\frac{\partial}{\partial C}f=&\frac{1}{2}(-O^TS^{-T}YX^TS^T-O^TS^{-T}YX^TS^T+(2O^TS^{-T}S^{-1}O)CSXX^TS^T\\&+O^TS^{-T}BUX^TS^T+O^TS^{-T}BUX^TS^T)\\=&-O^TS^{-T}(Y-S^{-1}OCSX-BU)X^TS^T.\end{aligned} \tag{A.15}$$

Similarly, the gradient of $f$ with respect to $O$ is

$$\begin{aligned}\frac{\partial}{\partial O}f=&\frac{1}{2}(-S^{-T}YX^TS^TC^T-S^{-T}YX^TS^TC^T+(2S^{-T}S^{-1})OCSXX^TS^TC^T)\\&+S^{-T}BUX^TS^TC^T+S^{-T}BUX^TS^TC^T)\\=&-S^{-T}(Y-S^{-1}OCSX-BU)X^TS^TC^T.\end{aligned} \tag{A.16}$$

The gradient of $f$ with respect to $B$ is

$$\begin{aligned}\frac{\partial}{\partial B}f=&\frac{1}{2}(-YU^T+S^{-1}OCSXU^T-YU^T+S^{-1}OCSXU^T+2BUU^T)\\=&-(Y-S^{-1}OCSX-BU)U^T.\end{aligned} \tag{A.17}$$

Last, the gradient of $f$ with respect to $S$ is calculated as follows:

$$\frac{1}{2}\|Y-S^{-1}OCSX-BU\|_F^2.=\langle R-Y|R-Y\rangle,$$

where $R=S^{-1}OCSX+BU$. Then, using the property

$$\frac{\partial X^{-1}}{\partial q}=-X^{-1}\frac{\partial X}{\partial q}X^{-1}$$

I calculate

$$\dot{R}=-S^{-1}\dot{S}S^{-1}OCSX+S^{-1}OC\dot{S}X,$$

such that

$$\begin{aligned}\dot{f} &= \frac{1}{2}\langle \dot{R}|R-Y\rangle + \langle R-Y|\dot{R}\rangle \\ &= \langle R-Y|\dot{R}\rangle \\ &= \langle R-Y| -S^{-1}\dot{S}S^{-1}OCSX + S^{-1}OC\dot{S}X\rangle \\ &= \langle -S^{-T}(R-Y)X^TS^TC^TO^TS^{-T} + C^TO^TS^{-T}(R-Y)X^T|\dot{S}\rangle.\end{aligned}$$

Thus, the gradient of $f$ with respect to $S$ is

$$\begin{aligned}\frac{\partial}{\partial S}f &= S^{-T}(Y-S^{-1}OCSX-BU)X^TS^TC^TO^TS^{-T} \\ &\quad -C^TO^TS^{-T}(Y-S^{-1}OCSX-BU)X^T. \end{aligned} \tag{A.18}$$

To simplify the notations, I use $E=Y-S^{-1}OCSX-BU$ and rewrite the gradients (A.15) through (A.18) as

$$\begin{aligned}\nabla_C f &= -O^TS^{-T}EX^TS^T \\ \nabla_O f &= -S^{-T}EX^TS^TC^T \\ \nabla_B f &= -EU^T \\ \nabla_S f &= S^{-T}EX^TS^TC^TO^TS^{-T} - C^TO^TS^{-T}EX^T,\end{aligned}$$

where I use the notation $\nabla_X(\cdot) \equiv \frac{\partial}{\partial X}(\cdot)$.

## A.8. Equivalent Matrix Representation for Sum of Squares Error

Note that the Frobenius norm of $A \in \mathbb{R}^{m\times n}$ is

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}.$$

Let $\tilde{\mathcal{K}}_d \in \mathbb{R}^{W\times W}$ and $\Psi(\cdot) \in \mathbb{R}^W$ and consider the expression

$$\sum_{k=1}^{P}\|\Psi(s(t_k+\Delta t),u(t_k+\Delta t))-\tilde{\mathcal{K}}_d\Psi(s(t_k),u(t_k))\|^2. \tag{A.19}$$

Then, using the Frobenius norm definition for the vector

$$\Psi(s(t_k+\Delta t),u(t_k+\Delta t))-\tilde{\mathcal{K}}_d\Psi(s(t_k),u(t_k)) \in \mathbb{R}^W,$$

(A.19) becomes

$$\text{(A.19)} = \sum_{k=1}^{P}(\sum_{i=1}^{W}|\Psi_i(s(t_k+\Delta t),u(t_k+\Delta t))-\tilde{\mathcal{K}}_{d_i}\Psi_i(s(t_k),u(t_k))|^2,$$

where $\tilde{\mathcal{K}}_{d_i} \in \mathbb{R}^W$ is a row vector that corresponds to the $i$th row of $\tilde{\mathcal{K}}_d$. Then, consider the term inside the absolute value as the $\mathcal{G}_{ik}$ element of a matrix $\mathcal{G} \in \mathbb{R}^{W\times P}$ such that

$$\text{(A.19)} = \sum_{k=1}^{P}(\sum_{i=1}^{W}|\mathcal{G}_{ij}||^2.$$

$\mathcal{G}$ can be expressed as

$$\mathcal{G}^T = \begin{bmatrix} (\Psi(s(t_1+\Delta t),u(t_1+\Delta t))-\tilde{\mathcal{K}}_d\Psi(s(t_1),u(t_1)))^T \\ \vdots \\ (\Psi(s(t_P+\Delta t),u(t_P+\Delta t))-\tilde{\mathcal{K}}_d\Psi(s(t_P),u(t_P)))^T \end{bmatrix}$$

and rewrite it as

$$\mathcal{G}^T = \begin{bmatrix} \Psi(s(t_1+\Delta t),u(t_1+\Delta t))^T \\ \vdots \\ \Psi(s(t_P+\Delta t),u(t_P+\Delta t))^T \end{bmatrix} - \begin{bmatrix} \Psi(s(t_1),u(t_1)))^T \\ \vdots \\ \Psi(s(t_P),u(t_P)))^T \end{bmatrix} \tilde{\mathcal{K}}_d^T.$$

Let $X,Y \in \mathbb{R}^{W\times P}$ be given by

$$X = \begin{bmatrix} \Psi(s(t_1),u(t_1))^T \\ \vdots \\ \Psi(s(t_P),u(t_P))^T \end{bmatrix}^T \quad Y = \begin{bmatrix} \Psi(s(t_1+\Delta t),u(t_1+\Delta t))^T \\ \vdots \\ \Psi(s(t_P+\Delta t),u(t_P+\Delta t))^T \end{bmatrix}^T$$

such that

$$\mathcal{G} = Y - \tilde{\mathcal{K}}_d X.$$

Then, using the Frobenius norm definition, (A.19) can be written as

$$\begin{aligned} (\text{A.19}) &= \|\mathcal{G}\|_F^2 \\ &= \|Y - \tilde{\mathcal{K}}_d X\|_F^2. \end{aligned}$$

## A.9. Memory-Preserving Gradient Descents for SOC Algorithm

Let $\Psi(s(t)) \in \mathbb{R}^W$ be given by

$$\Psi(s(t)) = \begin{bmatrix} \Psi_1(s(t)) & \Psi_2(s(t)) & \dots & \Psi_W(s(t)) \end{bmatrix}^T$$

and $X,Y \in \mathbb{R}^{W\times P}$ be given by

$$X = \begin{bmatrix} \mathbf{\Psi}(s(t_1)) & \mathbf{\Psi}(s(t_2)) & \dots & \mathbf{\Psi}(s(t_P)) \end{bmatrix}$$

and

$$Y = \begin{bmatrix} \mathbf{\Psi}(s(t_1+\Delta t)) & \mathbf{\Psi}(s(t_2+\Delta t)) & \dots & \mathbf{\Psi}(s(t_P+\Delta t)) \end{bmatrix}.$$

Then,

$$XX^T = \begin{bmatrix} \Psi(s(t_1)) & \Psi(s(t_2)) & ... & \Psi(s(t_P)) \end{bmatrix} \begin{bmatrix} \Psi(s(t_1))^T \\ \Psi(s(t_2))^T \\ \vdots \\ \Psi(s(t_P))^T \end{bmatrix}$$

$$= \begin{bmatrix} \Psi_1(s(t_1)) & \Psi_1(s(t_2)) & ... & \Psi_1(s(t_P)) \\ \Psi_2(s(t_1)) & \Psi_2(s(t_2)) & ... & \Psi_2(s(t_P)) \\ \vdots & \vdots & ... & \vdots \\ \Psi_W(s(t_1)) & \Psi_W(s(t_2)) & ... & \Psi_W(s(t_P)) \end{bmatrix} \cdot \begin{bmatrix} \Psi_1(s(t_1)) & \Psi_2(s(t_1)) & ... & \Psi_W(s(t_1)) \\ \Psi_1(s(t_2)) & \Psi_2(s(t_2)) & ... & \Psi_W(s(t_2)) \\ \vdots & \vdots & ... & \vdots \\ \Psi_1(s(t_P)) & \Psi_2(s(t_P)) & ... & \Psi_W(s(t_P)) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{k=1}^{P} \Psi_1(s(t_k)\Psi_1(s(t_k) & ... & \sum_{k=1}^{P} \Psi_1(s(t_k)\Psi_W(s(t_k) \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{P} \Psi_W(s(t_k)\Psi_1(s(t_k) & ... & \sum_{k=1}^{P} \Psi_W(s(t_k)\Psi_W(s(t_k) \end{bmatrix}$$

$$= \sum_{k=1}^{P} \begin{bmatrix} \Psi_1(s(t_k)\Psi_1(s(t_k) & ... & \Psi_1(s(t_k)\Psi_W(s(t_k) \\ \vdots & \ddots & \vdots \\ \Psi_W(s(t_k)\Psi_1(s(t_k) & ... & \Psi_W(s(t_k)\Psi_W(s(t_k) \end{bmatrix}$$

$$= \sum_{k=1}^{P} \Psi(s(t_k)\Psi(s(t_k)^T$$

$$= \mathcal{G}.$$

Similarly, $YX^T = \mathcal{A}$, $XU^T = X_U$, $YU^T = Y_U$, and $UU^T = U_U$.