DECOUPLED DATA-BASED CONTROL (D2C) FOR COMPLEX ROBOTIC SYSTEMS

A Dissertation

by

RAN WANG

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Suman Chakravorty
Co-Chair of Committee,	Dileep Kalathil
Committee Members,	John Valasek
	Raktim Bhattahcharya
Head of Department,	Ivett Leyva

August 2022

Major Subject: Aerospace Engineering

ABSTRACT

The problem of Reinforcement Learning (RL) is equivalent to the search for an optimal feedback control policy from data without system dynamics information. Most RL techniques search over a complex global nonlinear parametrization, such as deep neural nets, with drawbacks in training efficiency and solution variance. In this dissertation, we propose a decoupled data-based framework for RL/ data-based control that is highly efficient, robust and optimal when compared to state-of-the-art RL approaches. The efforts are primarily in three directions: learning to control 1) efficiently and reliably, 2) for high-dimensional nonlinear complex systems with partial state observations, and 3) under process and sensing uncertainties.

First, we propose a decoupling principle that leads to the decoupled data-based control (D2C) framework which designs the open-loop optimal trajectory and the closed-loop feedback law separately to achieve high training efficiency. Its convergence to the global optimum is proved. Simulation results on benchmark examples show its significant advantages in training efficiency, training reliability and robustness to noise over state-of-the-art RL methods.

Second, the D2C is extended to partially observed problems using a suitably defined “information state" which is implemented using autoregressive–moving-average (ARMA) system identification. We show that the resulting solution is the global optimum and satisfies a generalized minimum principle for the partially observed problem. The extended D2C technique allows us to solve the optimal control problem for partially observed, high-dimensional and nonlinear robotic systems.

Finally, we show that when learning to control in the fully observed case with process noise only, the extended D2C method converges to the global optimum. However, it is also shown that the method gives a biased result in the partially observed case with both process and measurement noise, where multiple rollouts need to be averaged to recover optimality.

DEDICATION

To Daorong Chen, the first aerospace scientist in my life, who has been inspiring me since high school.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor and chair of my committee Prof. Suman Chakravorty, for this precious opportunity to pursue a Ph.D. degree and for leading me through this journey. His great support and guidance both in research and life made everything that I have achieved possible. His wealth of knowledge, passion and vision in research, great mathematical skills, and so much more have always been what I look up to in the past few years and will still be in my future life.

I'm extremely grateful to my committee, Prof. Dileep Kalathil, Prof. John Valasek, Prof. Daniele Mortari and Prof. Raktim Bhattacharya for their great support and constructive advice in my research. The discussions with them, the courses and research work from them benefitted not only my academic development, but my professional development as well. Thanks should also go to Prof. Robert E. Skelton for the inspiration in the tensegrity research.

This endeavor would not have been possible without my labmates and friends. I want to thank Dr. Raman Goyal for his great help in my research. His brilliant mind, rich knowledge and great diligence impressed and motivated me a lot. Special thanks to Dr. Dan Yu, Karthikeya Parunandi, Mohamed Naveed Gul Mohamed and Aayushman Sharma for the contributions to this work and generous support in life. I enjoy working with these excellent researchers and feel extremely lucky about it. I am also grateful to Dr. Yuling Shen, Prof. Shuo Ma, Xuan Yang, Jiacheng Liu, Xiaolong Bai and Qifan Li. I will never forget the good memories when we explored this new world and worked together towards our goals. Words cannot express my gratitude to Dr. Muhao Chen, who has been by my side for the past nine years. I do appreciate the care, encouragement and guidance he gave me which carried me through hard times and added more joy to the good ones.

Most importantly, I want to thank my family. I could not have undertaken this journey without the great support and love from my parents Ying Wang and Shuili Wang, my cousins Ziyan Zha, Yu Pei and Zhuomin Liu, and my grandparents. I have been so far away from them these years, but they have always been my strongest backing. Most most importantly, I am deeply indebted to my

wife Jiaoyang Fan. In the past four years, she has been supporting me, taking great care of me and bringing tons of love to me. She is the light in my life and I know I am the luckiest man on earth to have her by my side for the rest of my life. I love you.

Last but not least, I wish to extend my sincere thanks to the great faculties and staff in the Department of Aerospace Engineering for their excellent lectures and kind help.

Things have changed a lot since 2020, I do hope the world we once believed in will shine again in grace and peace can be enjoyed everywhere.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Prof. Suman Chakravorty, Prof. John Valasek and Prof. Raktim Bhattacharya of the Department of Aerospace Engineering and Prof. Dileep Kalathil of the Department of Electrical Engineering.

Prof. Dileep Kalathil contributed to the discussion of Chapter 2. The DDPG results in Section 2.5 were generated by previous lab member Karthikeya Parunandi. The tensegrity modeling and control theory in Appendix A.1 and A.2 were developed by Dr. Raman Goyal and Prof. Robert Skelton. The material data in Chapter 3 was provided by current lab member Aayushman Sharma. Current lab member Mohamed Naveed Gul Mohamed developed the method of characteristics (MOC) results used in Section 3.4.1 and shared thoughts on the replanning results in Section 3.5. Dr. Raman Goyal also contributed to the theoretical results in Chapter 4 and the discussion of some simulation implementations. My advisor Prof. Suman Chakravorty has been my guiding light throughout this journey.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a graduate research assistantship and the AERO Grad Excellence fellowship from the Department of Aerospace Engineering, Texas A&M University. The research work was supported by NSF grants ECCS-1637889, CDSE 1802867, and AFOSR DDIP grant FA9550-17-1-0068.

NOMENCLATURE

DP	Dynamic Programming
RL	Reinforcement Learnin
TAMU	Texas A&M University
DDPG	Deep Deterministic Policy Gradien
DDP	Differential Dynamic Programming
ILQR	Iterative Linear Quadratic Regulator
ILQG	Iterative Linear Quadratic Gaussian
MPC	Model Predictive Control
ARMA	Autoregressive Moving Average
T-PFC	Trajectory-Optimized Perturbation Feedback Control
LLS	Linear Least Square
LQR	Linear Quadratic Regulator
KF	Kalman Filter
LQG	Linear Quadratic Gaussian control
D2C	Decoupled Data-Based Control
POD2C	Partially Observed Decoupled Data-Based Control
POD-iLQR	Partially Observed Data-Based ILQR
SQP	Sequential Quadratic Programming
SVD	Singular Value Decomposition
MBC	Model-Based Control
I.I.D.	Independent Identically Distributed
NLP	Non-Linear Programming

LLS-CD	Linear Least Squares by Central Difference
TD3	Twin-Delayed DDPG
SAC	Soft Actor-Critic
PDE	Partial Differential Equation
FD	Finite Difference
AR	Autoregressive
MA	Moving-Average
NARMA	Nonlinear ARMA
POMDPs	Partially Observed Markov Decision Processes
MDPs	Markov Decision Processes
LTV	Linear Time-Varying
LTI	Linear Time-Invariant
MOC	Method of Characterisrics

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

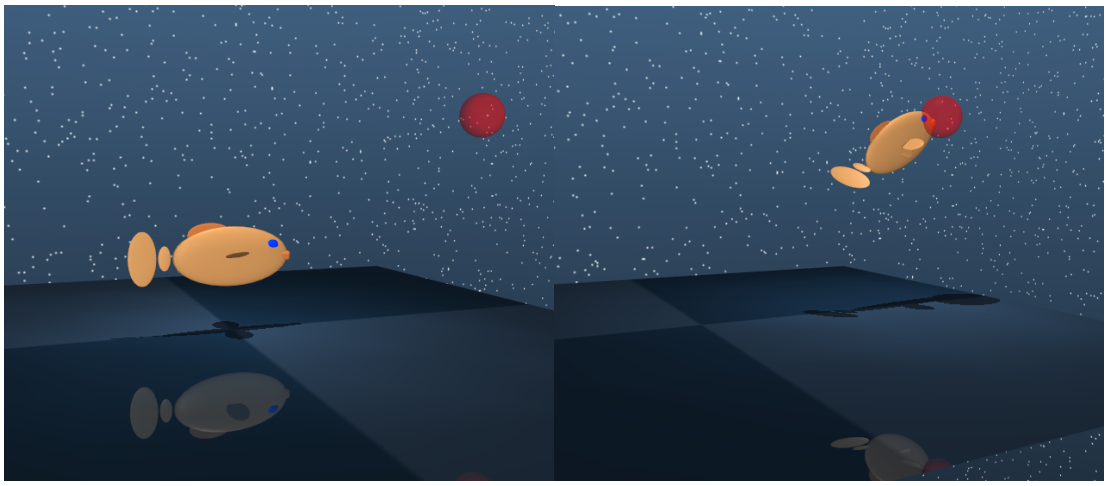
1. INTRODUCTION

This dissertation is targeted at establishing a systematic optimal control methodology for complex robotic systems in a stochastic environment. Complex robotic systems are those with high dimensionality and high nonlinearity. These features make it difficult to express the system dynamics analytically. Two examples of such systems are the multi-link swimmer robots and the tensegrity structures. In the swimmer robots, the fluid-structure interactions are intractable to model analytically and greatly increase the nonlinearity of the system. For the tensegrity structures, the number of dimensions can be more than a hundred in applications such as a robotic arm and airfoils [1]. On the other hand, these complex systems possess great advantages that traditional systems do not have in many real-world applications. For example, the agility of bionic swimmer robots, the tensegrity design can yield efficient minimal mass structure and the adaptive stiffness allows easy response adjustment. These properties can be useful in areas such as unmanned underwater vehicles, space structures, and earthquake-safe structure design [2, 3].

Figure. 1.1 shows the optimal control task for a robotic fish example. The fish is modeled as a multi-body system interacting with a static fluid. It has 27 states and 6 control inputs which are the torques on the fin and tail joints. The target is to solve for the optimal control policy for the fish to swim from its initial position to the red ball shown in the plot. With fluid interaction, the dynamics are hard to derive analytically, and thus, a data-based control method is necessary for this problem in the sense that the control algorithm only has access to a blackbox simulation model of the system.

1.1 Data-Based Optimal Control of Nonlinear Systems

To control the systems discussed above, classical control methods which mainly address linear problems are not suitable due to the system complexity and nonlinearity as well as model intractability. The study of the optimal control problem, which can be considered as a subset of optimization, has a rich literature [4, 5]. In the optimal control problem, a cost function needs to be selected according to the task. The system dynamics are the constraints that the optimization is subjected



(a) Initial State (b) Final State

Figure 1.1: Task configuration for the fish example.

to. Other constraints include the state and control bounds. Furthermore, the problem falls into the category of nonlinear optimization because of the nonlinear systems considered here.

In general, approaches to this problem can be divided into two broad classes, local and global. Global methods search for a solution that is optimal over the whole state and control space utilizing dynamic programming (DP) [6]. It searches backward from the terminal condition for the optimal solution that minimizes the cost-to-go at all time steps. When the number of states and controls is small, dynamic programming is effective. However, in complex systems, the state and control space is continuous. Thus discretization is necessary which results in the “curse of dimensionality" and makes the problem computationally intractable. The data-based version of dynamic programming is popularly known as reinforcement learning (RL) [7], which seeks to find a control policy by repeated interactions with the environment while observing the system’s responses. Standard RL includes value function-based methods like Q-learning and policy-based methods like policy gradient algorithms. The data collected in the interactions can be used to estimate the system dynamics for model-based methods [8], or estimate the control policy directly in a model-free fashion [7, 9, 10]. As great progress has been achieved in function approximation using deep neural networks during the last few years, reinforcement learning methods based on deep neural

networks have shown significantly improved performance in controlling dynamical systems. Among these methods, deep deterministic policy gradient (DDPG) [9] has many successful applications in nonlinear examples such as swimmer robots. In the DDPG algorithm, deep neural networks are used to approximate both the value function and the control policy. A solution is generated by improving the neural network fitting in each iteration for the DP equation. Despite the success, the training time required and the training variance of RL methods are still prohibitive, especially for the complex systems considered here.

In contrast, local “trajectory-based" methods can be more efficient. For example, the differential dynamic programming (DDP) approach [11] quadratizes the cost and system dynamics while the iterative linear quadratic regulator (iLQR) approach [12, 13, 14] quadratizes the cost and linearizes the system dynamics. Then starting from an initial guess, the optimal trajectory can be found iteratively. These methods have shown promising performance. However, it is unclear how efficient and reliable they are compared to global approaches in training as well as their performance in terms of robustness to noise. Furthermore, fast and reliable local planners have the potential to recover global optimality by coupling with methods such as model predictive control (MPC) [15] and trajectory-optimized perturbation feedback control (T-PFC) [16] that replan when necessary. By solving the optimization problem at each step, MPC enables the development of a globally optimum feedback law. The replanning is a bottleneck in that the optimization horizon may have to be limited so that the optimization takes a reasonable amount of time. For T-PFC, an optimal local (linear) feedback controller is wrapped around the nominal trajectory and replanning only happens when the state deviates sufficiently far from the nominal, which lessens the computational burden.

The model-based methods discussed above can not be applied directly due to the lack of analytical system equations, which leads us to the use of data-based methods. To extend model-based methods to data-based, system identification techniques can be used. For example, with state perturbation, the discrete-time Linear Time-Varying (LTV) system can be estimated using linear least square (LLS). Given input-output data collected from interactions with the environment, Q-Markov methods [17] solve for the LTV system matrices from the correlation of inputs and

outputs. Another option is the statistical models that are commonly used in time series analysis such as auto-regressive moving average (ARMA). Also, machine learning techniques and neural networks can be used for system identification [18]. These techniques make it possible to solve for the optimal control solution when data from either simulation or hardware experiments are available.

1.2 Partially Observed Control Problems

In many real-world applications, the sensors can only be placed on limited spots and the measurement is a function of the actual state information. Consider the hardware version of the fish example, a set of cameras can capture the position of the head and some encoders can return the angle of the actuators on the fins and tails. In this case, full-state measurement is no longer available, which leads us to partially observed problems. In state-of-the-art reinforcement learning methods such as DDPG, partially-observed cases are not considered. In the literature, such problems are typically extensively studied as belief space planning problems, where the belief state is defined as the filtered probability distribution over the states and provides a basis for acting under sensing uncertainty [19, 20]. One parametrization of the belief state is to assume a Gaussian belief that can be represented by the mean and covariance of the distribution [21]. However, the generic belief state is an infinite-dimensional object, and even the Gaussian parametrization requires $O(n_x + n_x^2)$ number of states, which makes it computationally intractable for higher-dimensional systems. Thus, our next goal is to propose a data-based method that can search for the optimal control policy with only partial observations.

1.3 Learning to Control under Uncertainty

In situations where even a simulation model is not available, or is too computationally intensive such as complex fluid dynamics software, hardware experiments have to be conducted to collect the data that are required by the data-based methods. Unlike in simulation software, the states and measurements are always subject to process and measurement noise in real-world experiments. For example, the slip in a wheeled robot, the complex fluid dynamics that simulation models do not

consider, and noisy sensors. Thus, techniques that minimize the effect of both process and sensor noise need to be implemented in the training step, for instance, the linear quadratic regulator (LQR) [22] and the Kalman filter (KF) [23]. RL methods like DDPG discussed above are expected to return the optimal policy in the stochastic environment, although most applications only have an exploration noise added during training. Thus, its performance under a persistent process noise, and its training stability, remain unclear.

1.4 Major Contributions

Given the above research lacunae, this dissertation presents a systematic approach to solve discrete-time, finite-horizon stochastic optimal control problems in a data-based fashion for complex, nonlinear high-dimensional robotic systems. The major contributions are listed below:

1. A near-optimal decoupling principle is developed, which demonstrates the closeness of the stochastic and deterministic optimal policies applied to the stochastic system. Further, it is revealed that the open-loop nominal part and the first order feedback part of the deterministic optimal policy can be solved separately. This result is the theoretical foundation of the proposed decoupled data-based control (D2C) algorithm.

2. The D2C algorithm is established based on the decoupling principle. It takes the data from interactions between the system and the environment to solve for the linear deterministic optimal control policy. The implementation of D2C on the control of tensegrity structures is demonstrated. Advantages are shown over the model-based shape control method.

3. The global optimality of the D2C algorithm is proved, which guarantees that under mild conditions, the optimization result of D2C always converges to the global optimum. Simulation experiments are conducted to demonstrate the performance of D2C on 1) training efficiency, 2) closed-loop performance, 3) reliability of training, in which advantages over state-of-the-art RL methods are also shown.

4. An extension for partially observed systems (POD2C) is developed which converts the partially observed problem to a “fully observed" problem in the information state form and

designs a local feedback control law. The conditions to exactly match the ARMA model with the LTV model are provided. We show that the solution found by POD2C is the global optimum of the partially observed optimal control problem and satisfies a generalized minimum principle. This algorithm requires only the input-output data where the output can be a small subset of the states. The performance of this algorithm is tested on complex, nonlinear high-dimensional robotic systems under process and sensing uncertainties.

5. The problem of learning under noise is studied by applying POD2C to fully observed and partially observed systems under noise. It is proved that the open-loop optimal trajectory design method partially observed data-based iLQR (POD-iLQR) converges to the global minimum in the fully observed case with process noise only. In the partially observed case with process and measurement noise, it is shown that POD-iLQR can not converge to the global optimum due to the bias in the ARMA model fitting. To recover optimality, multiple rollouts need to be averaged to eliminate the bias.

1.5 Dissertation Outline

The Outline of this dissertation is shown in Fig. 1.2.

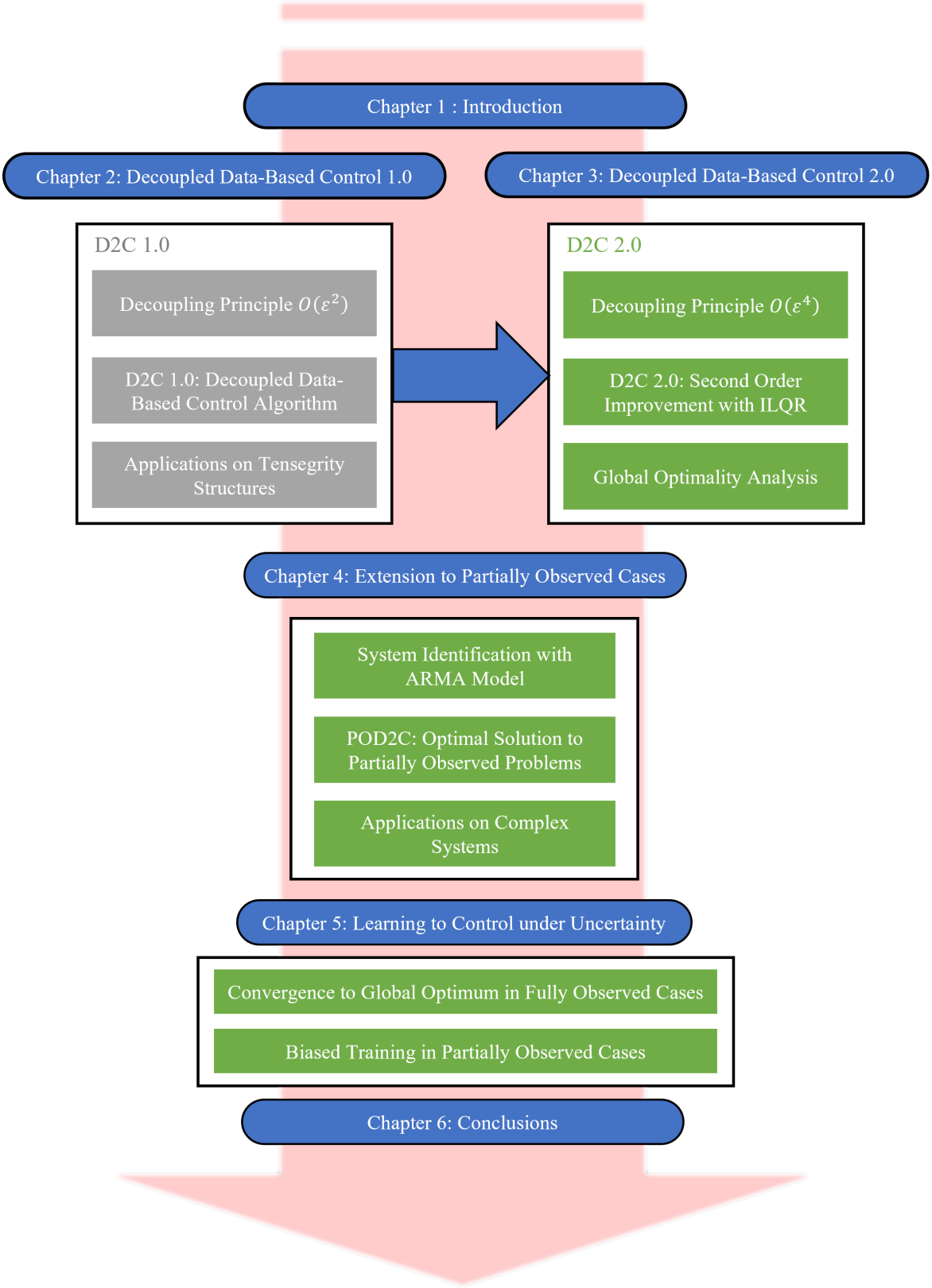


Figure 1.2: Dissertation outline.

2. DECOUPLED DATA-BASED CONTROL 1.0: A FIRST ORDER APPROACH*

2.1 Introduction

The control of an unknown dynamical system adaptively has a rich history in control literature [4]. This classical literature provides a rigorous analysis of the asymptotic performance and stability of the linear closed-loop system. The optimal control of a stochastic nonlinear system with continuous state space and action space is a significantly more challenging problem due to the "curse of dimensionality", the exponential computational complexity growth associated with dynamic programming. *Learning to control* problems where the model of the system is unknown also suffer from these computational complexity issues, in addition to the usual identifiability problems in adaptive control.

The last several years have seen significant progress in deep neural networks-based reinforcement learning approaches for controlling unknown dynamical systems, with applications in many areas like playing games [24], locomotion [25] and robotic hand manipulation [26]. Many new algorithms that show promising performance are proposed [27] [10] and various improvements and innovations have been continuously developed. However, despite excellent performance on some tasks, RL is still considered very data and time-intensive. The training time for such algorithms is typically really large. Moreover, high variance and reproducibility issues on the performance are reported [28].

In general, the solution approaches to the problem of controlling unknown dynamical systems can be divided into two broad classes, local and global.

The most computationally efficient among these techniques are "local" trajectory-based methods such as DDP, [11, 29], which quadratize the dynamics and the cost-to-go function around a nominal trajectory, and the iLQR, [12, 13], which only linearizes the dynamics, and thus, is more efficient.

*Part of this chapter is reprinted with permission from "Decoupled data-based approach for learning to control nonlinear dynamical systems" by R. Wang, K. S. Parunandi, D. Yu, D. Kalathil and S. Chakravorty, 2021. IEEE Transactions on Automatic Control, Copyright 2021 by IEEE and "Model and data based approaches to the control of tensegrity robots" by R. Wang, R. Goyal, S. Chakravorty and R. E. Skelton, 2020. IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 3846-3853, Copyright 2020 by IEEE.

The local approaches to control unknown systems have been explored before [14, 30], however, they have always been characterized as “trajectory optimization" techniques and have been thought of as distinct from RL. Also, the advantages in training efficiency and robustness to noise of the local approaches when compared to the global approaches have not been recognized.

Global methods, more popularly known as approximate dynamic programming [31, 6] or RL methods [7], seek to improve the control policy by repeated interactions with the environment while observing the system's responses. The repeated interactions, or learning trials, allow these algorithms to compute the solution of the dynamic programming problem (optimal value/Q-value function or optimal policy) either by constructing a model of the dynamics (model-based) [32, 8, 33], or directly estimating the control policy (model-free) [7, 10]. Standard RL algorithms are broadly divided into value-based methods, like Q-learning, and policy-based methods, like policy gradient algorithms. Recently, function approximation using deep neural networks has significantly improved the performance of reinforcement learning algorithms, leading to a growing class of literature on “deep reinforcement learning" [27, 10, 34, 9]. Despite the success, the amount of samples and training time required still seem prohibitive. On the other hand, works such as [35] demonstrated that simple policies such as the ones with linear parameterization showed a promising performance comparable to benchmark results obtained by policies represented using deep neural networks.

In this chapter, we propose a novel decoupled data-based control algorithm (D2C 1.0) for learning to control an unknown nonlinear dynamical system. Our approach introduces a rigorous decoupling of the open-loop (planning) problem from the closed-loop (feedback control) problem. This decoupling allows us to come up with a highly efficient approach to solve the problem in a completely data-based fashion. Our approach proceeds in two steps: (i) first, we optimize the nominal open-loop trajectory of the system using the first order gradient descent method with data from a blackbox simulation model, (ii) then we identify the LTV system governing perturbations from the nominal trajectory using random input-output perturbation data, and design an LQR controller for this linearized system. We show that the performance of the D2C 1.0 algorithm is approximately optimal, in the sense that the decoupled design is near-optimal to the second order in

a suitably defined noise parameter. Moreover, simulation performance on various robotic examples suggests a significant reduction in training time compared to other state-of-the-art RL algorithms. To further demonstrate the performance of D2C 1.0, we test it using tensegrity structures in simulation and compare it with a model-based shape control method.

The rest of the chapter is organized as follows. In Section 2.2, the basic problem formulation is outlined. In Section 2.3, a decoupling result which solves the MDP in a "decoupled open loop-closed loop " fashion is introduced. In Section 2.4, we propose the D2C 1.0 algorithm based on the decoupling principle, with discussions of implementation problems. In Section 2.5, we test the proposed approach using four typical benchmarking examples with comparisons to a state-of-the-art RL technique. In Section 2.6, we demonstrate the efficacy of D2C 1.0 on the tensegrity structures in comparison with a model-based shape control (MBC) method.

2.2 Problem Formulation

First, let's define the stochastic optimal control problem:

$$\begin{aligned} &\min_{u_t} \tilde{J}(x_0) = \mathbb{E}[\sum_{t=0}^{T-1} c(x_t, u_t) + c_T(x_T)] \\ &s.t. \quad x_{t+1} = f(x_t, u_t) + \epsilon w_t, \end{aligned} \tag{2.1}$$

where $x_t \in \mathbb{R}^{n_x}$ and $u_t = \pi_t(x_t) \in \mathbb{R}^{n_u}$ are the state and control vector at time t, $c(\cdot)$ and $c_T(\cdot)$ are the incremental and terminal cost functions, $\epsilon < 1$ is a small parameter and w_t is the Gaussian white noise. The optimization is subject to the system dynamics which are affine in control for mechanical systems. The *stochastic optimal control* problem is to find the the control policy $\pi^o = \{\pi_0^o, \pi_2^o, \cdots, \pi_{T-1}^o\}$ such that the expected cumulative cost is minimized, i.e., $\pi^o = \arg\min_\pi \ \tilde{J}^\pi(x_0)$. In the following, we assume that the initial state x_0 is fixed, and denote $\tilde{J}^\pi(x_0)$ simply as $\tilde{J}^\pi$.

2.3 A Near-Optimal Decoupling Principle

To construct a data-based solution to this problem, we first outline a near-optimal decoupling principle in stochastic optimal control that paves the way for the D2C 1.0 algorithm described in Section 2.4.

2.3.1 Linearization w.r.t. Nominal Trajectory

Consider a noiseless version of the system dynamics given in Eq. (2.1), $w_t = 0$ for all t. We denote the 'nominal' state trajectory as $\bar{x}_t$ and the 'nominal' control as $\bar{u}_t$, with the initial condition, $\bar{x}_0 = x_0$, known exactly. The resulting dynamics without noise is given by $\bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t)$. Let $\pi = (\pi_t)_{t=0}^{T-1}$ be a given control policy, i.e., $u_t = \pi_t(x_t)$, and thus, $\bar{u}_t = \pi_t(\bar{x}_t)$.

Assuming that $f(\cdot)$ and $\pi_t(\cdot)$ are sufficiently smooth, we can linearize the dynamics about the nominal trajectory. Denoting $\delta x_t = x_t - \bar{x}_t, \delta u_t = u_t - \bar{u}_t$, we can express,

$$\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t + S_t(\delta x_t, \delta u_t) + \epsilon w_t, \tag{2.2}$$

$$\delta u_t = K_t \delta x_t + \tilde{S}_t(\delta x_t), \tag{2.3}$$

where $A_t = \frac{\partial f}{\partial x}|_{\bar{x}_t, \bar{u}_t}$, $B_t = \frac{\partial f}{\partial u}|_{\bar{x}_t, \bar{u}_t}$, $K_t = \frac{\partial \pi_t}{\partial x}|_{\bar{x}_t}$, and $S_t(\cdot, \cdot), \tilde{S}_t(\cdot)$ are second and higher-order terms in the respective expansions. Similarly, we can linearize the instantaneous cost $c(x_t, u_t)$ about the nominal values $(\bar{x}_t, \bar{u}_t)$ as,

$$c(x_t, u_t) = c(\bar{x}_t, \bar{u}_t) + C_t^x \delta x_t + C_t^u \delta u_t + H_t(\delta x_t, \delta u_t), \tag{2.4}$$

$$c_T(x_T) = c_T(\bar{x}_T) + C_T^x \delta x_T + H_T(\delta x_T), \tag{2.5}$$

where $C_t^x = \frac{\partial c}{\partial x}|_{\bar{x}_t, \bar{u}_t}$, $C_t^u = \frac{\partial c}{\partial u}|_{\bar{x}_t, \bar{u}_t}$, $C_T^x = \frac{\partial c_T}{\partial x}|_{\bar{x}_t}$, and $H_t(\cdot, \cdot)$ and $H_T(\cdot)$ are second and higher-order terms in the respective expansions.

Using Eq. (2.2) and (2.3), we can write the closed-loop dynamics of the trajectory $(\delta x_t)_{t=1}^T$ as,

$$\delta x_{t+1} = \underbrace{(A_t + B_t K_t)}_{\bar{A}_t} \delta x_t + \underbrace{\{B_t \tilde{S}_t(\delta x_t) + S_t(\delta x_t, K_t \delta x_t + \tilde{S}_t(\delta x_t)))\}}_{\bar{S}_t(\delta x_t)} + \epsilon w_t, \tag{2.6}$$

where $\bar{A}_t$ represents the linear part of the closed-loop systems and the term $\bar{S}_t(.)$ represents the second and higher-order terms in the closed-loop system. Similarly, the closed-loop incremental cost given in Eq. (2.4) can be expressed as

$$c(x_t, u_t) = \underbrace{c(\bar{x}_t, \bar{u}_t)}_{\bar{c}_t} + \underbrace{[C_t^x + C_t^u K_t]}_{\bar{C}_t} \delta x_t + \underbrace{H_t(\delta x_t, K_t \delta x_t + \tilde{S}_t(\delta x_t))}_{\bar{H}_t(\delta x_t)}. \tag{2.7}$$

Therefore, the cumulative cost of any given closed-loop trajectory $(x_t, u_t)_{t=0}^T$ can be expressed as,

$$\begin{aligned} J^\pi &= \sum_{t=0}^{T-1} c(x_t, u_t = \pi_t(x_t)) + c_T(x_T) \\ &= \sum_{t=0}^{T} \bar{c}_t + \sum_{t=0}^{T} \bar{C}_t \delta x_t + \sum_{t=0}^{T} \bar{H}_t(\delta x_t), \end{aligned} \tag{2.8}$$

where $\bar{c}_T = c_T(\bar{x}_T), \bar{C}_T = C_T^x, \bar{H}_T(.) = H_T(.)$.

We first show the following results.

Lemma 1. *The state perturbation equation*

$$\delta x_{t+1} = \bar{A}_t \delta x_t + \bar{S}_t(\delta x_t) + \epsilon w_t$$

given in Eq. (2.6) *can be equivalently characterized as*

$$\delta x_t = \delta x_t^l + \bar{\bar{S}}_t, \ \delta x_{t+1}^l = \bar{A}_t \delta x_t^l + \epsilon w_t \tag{2.9}$$

where $\bar{\bar{S}}_t$ is an $O(\epsilon^2)$ function that depends on the entire noise history $\{w_0, w_1, \cdots w_t\}$ and δx_t^l

evolves according to the linear closed-loop system as above.

Proof. We proceed by induction. The first general instance of the recursion occurs at $t = 3$. It can be shown that:

$$\delta x_3 = \underbrace{(\bar{A}_2\bar{A}_1(\epsilon w_0) + \bar{A}_2(\epsilon w_1) + \epsilon w_2)}_{\delta x_3^l} + \underbrace{\{\bar{A}_2\bar{S}_1(\epsilon w_0) + \bar{S}_2(\bar{A}_1(\epsilon w_0) + \epsilon w_1 + \bar{S}_1(\epsilon w_0))\}}_{\bar{\bar{S}}_3}. \quad (2.10)$$

Noting that $\bar{S}_1(.)$ and $\bar{S}_2(.)$ are second and higher order terms, it follows that $\bar{\bar{S}}_3$ is $O(\epsilon^2)$. Suppose now that $\delta x_t = \delta x_t^l + \bar{\bar{S}}_t$ where $\bar{\bar{S}}_t$ is $O(\epsilon^2)$. Then: $\delta x_{t+1} = \bar{A}_{t+1}(\delta x_t^l + \bar{\bar{S}}_t) + \epsilon w_t + \bar{S}_{t+1}(\delta x_t) = \underbrace{(\bar{A}_{t+1}\delta x_t^l + \epsilon w_t)}_{\delta x_{t+1}^l} + \underbrace{\{\bar{A}_{t+1}\bar{\bar{S}}_t + \bar{S}_{t+1}(\delta x_t)\}}_{\bar{\bar{S}}_{t+1}}$. Noting that $\bar{S}_{t+1}$ is $O(\epsilon^2)$ and $\bar{\bar{S}}_{t+1}$ is $O(\epsilon^2)$ by assumption, the result follows. □

Lemma 2. *Let δJ_1^π, δJ_2^π be as defined in Eq.* (2.11)*. Then, $\mathbb{E}[\delta J_1^\pi \delta J_2^\pi]$ is an $O(\epsilon^4)$ function.*

Proof. In the following, we suppress the explicit dependence on π for δJ_1^π and δJ_2^π for convenience. Recall that $\delta J_1 = \sum_{t=0}^T C_t^x \delta x_t^l$, and $\delta J_2 = \sum_{t=0}^T \bar{H}_t(\delta x_t) + C_t^x \bar{\bar{S}}_t$. As before, let us consider $\bar{\bar{S}}_3$. We have that $\bar{\bar{S}}_3 = \bar{A}_2\bar{S}_1(\epsilon w_0) + \bar{S}_2(\bar{A}_1(\epsilon w_0) + \epsilon w_1 + \bar{S}_1(\epsilon w_0))$. Note that $\epsilon w_0 = \delta x_1^l$ and $\bar{A}_1(\epsilon w_0) + \epsilon w_1 = \delta x_2^l$. Then, it follows that: $\bar{\bar{S}}_3 = \bar{A}_2 \begin{bmatrix} \delta x_1^{l\intercal} \bar{S}_{1,1}^{(2)} \delta x_1^l \\ \vdots \\ \delta x_1^{l\intercal} \bar{S}_{1,n}^{(2)} \delta x_1^l, \end{bmatrix} + \begin{bmatrix} \delta x_2^{l\intercal} \bar{S}_{2,1}^{(2)} \delta x_2^l \\ \vdots \\ \delta x_2^{l\intercal} \bar{S}_{2,n}^{(2)} \delta x_2^l, \end{bmatrix} + O(\epsilon^3)$, where the Hessian matrices $\{\bar{S}_{t,j}^{(2)}, j = 1, 2 \cdots n\}$ correspond to the second order term in the Taylor expansion of the n dimensional vector valued function $\bar{S}_t(.)$. A similar observation holds for $\bar{H}_3(\delta x_3)$ in that: $\bar{H}_3(\delta x_3) = \delta x_3^{l\intercal} \bar{H}_3^{(2)} \delta x_3^l + O(\epsilon^3)$, where $\bar{H}_t^{(2)}$ represents the Hessian matrix corresponding to the second order term in the Taylor expansion of the scalar-valued function $\bar{H}_t(.)$. Therefore, from the above equations, it follows that we may write: $\bar{H}_t(\delta x_t) + C_t^x \bar{\bar{S}}_t = \sum_{\tau=0}^t \delta x_\tau^{l\intercal} Q_{t,\tau} \delta x_\tau^l + O(\epsilon^3)$, for suitably defined matrix coefficients $Q_{t,\tau}$. Therefore, it follows that $\delta J_2 = \sum_{t=0}^T \bar{H}_t(\delta x_t) + C_t^x \bar{\bar{S}}_t = \sum_{\tau=0}^T \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l + O(\epsilon^3)$, for suitably defined matrices $\bar{Q}_{T,\tau}$. Hence, $\delta J_1 \delta J_2 = \sum_{t,\tau=0}^T C_t^x (\delta x_t^l) \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l + O(\epsilon^4)$. Taking expectations on both sides: $\mathbb{E}[\delta J_1 \delta J_2] = \sum_{t,\tau=0}^T \mathbb{E}[C_t^x \delta x_t^l \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l] + O(\epsilon^4)$. Break $\delta x_t^l = (\delta x_t^l - \delta x_\tau^l) + \delta x_\tau^l$, assuming $\tau < t$.

Then, it follows that: $\mathbb{E}[C_t^x \delta x_t^l \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l] = \mathbb{E}[C_t^x \delta x_\tau^l \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l]$, due to the independence of $\delta x_t^l - \delta x_\tau^l$ from δx_τ^l, and the fact that $\mathbb{E}[\delta x_t^l - \delta x_\tau^l] = 0$. Note that we may write $\delta x_\tau^l = \epsilon[\beta_{\tau-1}\omega_{\tau-1} + \cdots + \beta_0\omega_0]$, for suitably defined co-efficients $\beta_0, \beta_1 \cdots$. Therefore, it follows that: $\delta x_\tau^{l\prime} \bar{Q}_{T,\tau} \delta x_\tau^l = \epsilon^2 \sum_{k,l=0}^{\tau} \omega_k' \tilde{Q}_{kl}^{T,\tau} \omega_l$, for suitably defined matrices $\tilde{Q}_{kl}^{T,\tau}$. Therefore, it follows that $C_t^x \delta x_\tau^l \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l = \epsilon^3 \sum_{t_1,t_2,t_3=0}^{\tau-1} \sum_{i,j,k=1}^{p} \omega_{t_1}^i \omega_{t_2}^j \omega_{t_3}^k \alpha_{t_1,t_2,t_3}^{i,j,k}$, for suitably defined constants $\alpha_{t_1,t_2,t_3}^{i,j,k}$, where ω_t^i represents the i^{th} input noise term at time t and p is the total number of inputs to the system. Hence, $\mathbb{E}[C_t^x \delta x_\tau^l \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l] = \epsilon^3 \sum_{t_1,t_2,t_3=0}^{\tau-1} \sum_{i,j,k=1}^{p} \mathbb{E}[\omega_{t_1}^i \omega_{t_2}^j \omega_{t_3}^k] \alpha_{t_1,t_2,t_3}^{i,j,k}$. Note now that $\mathbb{E}[\omega_{t_1}^i \omega_{t_2}^j \omega_{t_3}^k] = 0$ unless $t_1 = t_2 = t_3$, regardless of i, j, k, since the noise is assumed to be white in time. If the input channels are uncorrelated, and the input noise standard Gaussian, it follows that $\mathbb{E}[\omega_s^i \omega_s^j \omega_s^k] = 0$ regardless of i, j, k since odd moments of a zero mean Gaussian variable are zero.

Next, let us consider the case that the input channels are correlated. Then $\omega_s = \sqrt{W}\nu$, where W is the covariance of ω_s and ν is a Gaussian input vector that has identity covariance. Then, it follows that: $\mathbb{E}[\omega_s^i \omega_s^j \omega_s^k] = \sum_{i_1,i_2,i_3=1}^{p} \mathbb{E}[\nu_{i_1}\nu_{i_2}\nu_{i_3}] d_{i_1,i_2,i_3}$, for suitably defined coefficients d_{i_1,i_2,i_3}. However, due to our previous argument, $\mathbb{E}[\nu_{i_1}\nu_{i_2}\nu_{i_3}] = 0$ due to the noise input ν being spatially uncorrelated and Gaussian. Therefore, from the above above argument it follows that: $\mathbb{E}[C_t^x \delta x_\tau^l \delta x_\tau^{l\intercal} \bar{Q}_{T,\tau} \delta x_\tau^l] = 0$. Therefore, using the above fact, it follows that $\mathbb{E}[\delta J_1 \delta J_2] = O(\epsilon^4)$, thereby proving the result when $t > \tau$.

An analogous argument as above can be repeated for the case when $\tau > t$. □

Now, we show the following important result:

Proposition 1. *The mean and variance of the closed-loop cost J^π obey the following relationships, where $\bar{J}^\pi = \sum_{t=0}^{T} \bar{c}_t$, and $\delta J_1^\pi = \sum_{t=0}^{T} \bar{C}_t \delta x_t^l$ (see Eq. (2.8)-(2.9)):*

$$\tilde{J}^\pi = \mathbb{E}[J^\pi] = \bar{J}^\pi + O(\epsilon^2),$$

$$Var(J^\pi) = \underbrace{Var(\delta J_1^\pi)}_{O(\epsilon^2)} + O(\epsilon^4).$$

Proof. Using Eq. (2.9) in Eq. (2.8), we can obtain the cumulative cost of any sample closed-loop

trajectory as,

$$J^\pi = \underbrace{\sum_{t=0}^{T} \bar{c}_t}_{\bar{J}^\pi} + \underbrace{\sum_{t=0}^{T} \bar{C}_t \delta x_t^l}_{\delta J_1^\pi} + \underbrace{\sum_{t=0}^{T} \bar{H}_t(\delta x_t) + \bar{C}_t \bar{\bar{S}}_t}_{\delta J_2^\pi}. \tag{2.11}$$

From Eq. (2.11), we get,

$$\begin{aligned} \tilde{J}^\pi = \mathbb{E}[J^\pi] &= \mathbb{E}[\bar{J}^\pi + \delta J_1^\pi + \delta J_2^\pi], \\ &= \bar{J}^\pi + \mathbb{E}[\delta J_2^\pi] = \bar{J}^\pi + O(\epsilon^2), \end{aligned} \tag{2.12}$$

The first equality in the last line of the equations before follows from the fact that $\mathbb{E}[\delta x_t^l] = 0$, since its the linear part of the state perturbation driven by Gaussian white noise and by definition $\delta x_1^l = 0$. The second equality follows from the fact that δJ_2^π is an $O(\epsilon^2)$ function since $\bar{H}_t(\delta x_t)$ and $\bar{\bar{S}}_t$ are both $O(\epsilon^2)$ functions. Let $\delta \tilde{J}_2^\pi \equiv \mathbb{E}[\delta J_2^\pi]$. Noting that $\delta \tilde{J}_1^\pi \equiv \mathbb{E}[\delta J_1^\pi] = 0$, we obtain:

$$\begin{aligned} \text{Var}(J^\pi) &= \mathbb{E}[J^\pi - \tilde{J}^\pi]^2 \\ &= \mathbb{E}[\bar{J}^\pi + \delta J_1^\pi + \delta J_2^\pi - \bar{J}^\pi - \delta \tilde{J}_2^\pi]^2 \\ &= \mathbb{E}[\delta J_1^\pi + \delta J_2^\pi - \delta \tilde{J}_2^\pi]^2 \\ &= \text{Var}(\delta J_1^\pi) + \text{Var}(\delta J_2^\pi) + 2\mathbb{E}[\delta J_1^\pi (\delta J_2^\pi - \delta \tilde{J}_2^\pi)], \\ &= \text{Var}(\delta J_1^\pi) + \text{Var}(\delta J_2^\pi) + 2\mathbb{E}[\delta J_1^\pi \delta J_2^\pi], \end{aligned} \tag{2.13}$$

where the last equality follows from the fact that $\mathbb{E}[\delta J_1^\pi] = 0$ and the fact that $\delta \tilde{J}_2^\pi$ is non-random. Since δJ_2^π is $O(\epsilon^2)$, $\text{Var}(\delta J_2^\pi)$ is an $O(\epsilon^4)$ function. As is shown in Lemma 2, $\mathbb{E}[\delta J_1^\pi \delta J_2^\pi]$ is $O(\epsilon^4)$ as well. Finally $\text{Var}(\delta J_1^\pi)$ is an $O(\epsilon^2)$ function because δx_t^l is an $O(\epsilon)$ function. Combining these, we get the desired result. □

The following observations can now be made from Proposition 1.

Remark 1 (Expected cost-to-go)**.** *Recall that* $u_t = \pi_t(x_t) = \bar{u}_t + K_t \delta x_t + \tilde{S}_t(\delta x_t)$*. However, note*

that due to Proposition 1, the expected cost-to-go, $\tilde{J}^\pi$, is determined to within $O(\epsilon^2)$ by the nominal control action sequence $\bar{u}_t$.

Remark 2 (Variance of cost-to-go)**.** *Given nominal control action $\bar{u}_t$, variance of the cost-to-go, which is $O(\epsilon^2)$, is determined to within $O(\epsilon^4)$ by the linear feedback term $K_t\delta x_t$.*

2.3.2 Decoupled Approach for Feedback Control

Proposition 1 and the remarks above allow us to propose the following decoupled approach to stochastic nonlinear feedback control in the sense that the open-loop design is decoupled from the closed-loop design.

Open-Loop Design. First, we design an optimal (open-loop) control sequence $\bar{u}_t^*$ for the noiseless system. More precisely,

$$\begin{aligned}(\bar{u}_t^*)_{t=0}^{T-1} &= \arg\min_{(\bar{u}_t)_{t=0}^{T-1}} \sum_{t=0}^{T-1} c(\bar{x}_t, \bar{u}_t) + c_T(\bar{x}_T), \\ \bar{x}_{t+1} &= f(\bar{x}_t, \bar{u}_t), \bar{x}_0 = x_0.\end{aligned} \tag{2.14}$$

Details of this open-loop design are discussed in Section 2.4.

closed-loop Design. We find the optimal feedback gain K_t^* such that the variance of the linear closed-loop system around the optimal nominal path, $(\bar{x}_t, \bar{u}_t^*)$, is minimized.

$$\begin{aligned}(K_t^*)_{t=0}^{T-1} &= \arg\min_{(K_t)_{t=0}^{T-1}} \text{Var}(\delta J_1^\pi), \\ \delta J_1^\pi &= \sum_{t=0}^{T} \bar{C}_t \delta x_t^l, \\ \delta x_{t+1}^l &= (A_t + B_t K_t)\delta x_t^l + \epsilon w_t.\end{aligned} \tag{2.15}$$

We characterize the approximate closed-loop policy below.

Proposition 2. *Construct a closed-loop policy*

$$\pi_t^*(x_t) = \bar{u}_t^* + K_t^* \delta x_t, \tag{2.16}$$

where $\bar{u}_t^*$ is the solution of the open-loop problem Eq. (2.14)*, and K_t^* is the solution of the closed-loop problem Eq.* (2.15)*. Let π^o be the optimal closed-loop policy. Then, $|\tilde{J}^{\pi *} - \tilde{J}^{\pi^o}| = O(\epsilon^2)$. Furthermore, among all policies with nominal control action $\bar{u}_t^*$, the variance of the cost-to-go under policy π_t^*, is within $O(\epsilon^4)$ of the variance of the policy with the minimum variance.*

Proof. Let $\bar{J}^{\pi^o}$ denote the nominal cost of the optimal policy π^o, where recall from before that the nominal cost is the closed-loop cost when all the noise inputs are identically zero. Then, we have $\tilde{J}^{\pi^*} - \tilde{J}^{\pi^o} = \tilde{J}^{\pi^*} - \bar{J}^{\pi^*} + \bar{J}^{\pi^*} - \tilde{J}^{\pi^o} \leq \tilde{J}^{\pi^*} - \bar{J}^{\pi^*} + \bar{J}^{\pi^o} - \tilde{J}^{\pi^o}$. The inequality in the last line above is due the fact that $\bar{J}^{\pi^*} \leq \bar{J}^{\pi^o}$, since the nominal control corresponding to π^*, $\bar{u}_t^*$, is the minimizer for the nominal optimal control (open-loop) problem. Now, using Proposition 1 for the policies π^* and π^o, we have that $|\tilde{J}^{\pi^*} - \bar{J}^{\pi^*}| = O(\epsilon^2)$, and $|\tilde{J}^{\pi^o} - \bar{J}^{\pi^o}| = O(\epsilon^2)$. Also, by definition, we have $\tilde{J}^{\pi^o} \leq \tilde{J}^{\pi^*}$, i.e., the expected cost of π^o is lower than that of π^* since π^o minimizes the expected cost over all feedback policies. Note that this is different from u_t^* minimizing the nominal (open-loop) cost of the system. Then, since $\tilde{J}^{\pi^*} - \tilde{J}^{\pi^o} \geq 0$, using the inequality above, we obtain: $|\tilde{J}^{\pi^*} - \tilde{J}^{\pi^o}| \leq |\tilde{J}^{\pi^*} - \bar{J}^{\pi^*} + \bar{J}^{\pi^o} - \tilde{J}^{\pi^o}| \leq |\tilde{J}^{\pi^*} - \bar{J}^{\pi^*}| + |\bar{J}^{\pi^o} - \tilde{J}^{\pi^o}| = O(\epsilon^2)$. A similar argument holds for the variance as well. □

The closed-loop cost function in Eq. (2.15) can be written as (after noting $\delta u_t = K_t \delta x_t$):

$$\begin{aligned} \text{Var}(\delta J_1^\pi) &= \mathbb{E}[\sum_{t,\tau=0}^{T} [\delta x_t^l \ \delta u_t] \mathcal{Q}_{t,\tau} \begin{bmatrix} \delta x_t^l \\ \delta u_t \end{bmatrix}], \\ \mathcal{Q}_{t,\tau} &= \begin{pmatrix} C_t^{x\intercal} C_\tau^x & C_t^{x\intercal} C_\tau^u \\ C_t^{u\intercal} C_\tau^x & C_t^{u\intercal} C_\tau^u \end{pmatrix}, \end{aligned} \tag{2.17}$$

and $C_T^u = 0$. This problem is non-standard: a standard LQR problem only has a single sum instead

of the double sum over time above. Albeit convex, there is no standard solution to the problem above. Therefore, we solve a standard LQR problem as a surrogate and the effect is one of reducing the variance of the cost-to-go.

Approximate Closed-Loop Problem. We solve the following LQR problem for suitably defined cost function weighting factors Q_t, R_t:

$$\min_{(\delta u_t)_{t=0}^{T}} \mathbb{E}[\sum_{t=0}^{T-1} \delta x_t' Q_t \delta x_t + \delta u_t' R_t \delta u_t + \delta x_T' Q_T \delta x_t],$$
$$s.t. \quad \delta x_{t+1} = A_t \delta x_t + B_t \delta u_t + \epsilon w_t. \tag{2.18}$$

The solution to the above problem furnishes us a feedback gain $\hat{K}_t^*$ which we can use in the place of the true variance minimizing gain K_t^*.

Remark 3. *Proposition 1 states that the expected cost-to-go of the problem is dominated by the nominal cost-to-go. Therefore, even an open-loop policy consisting of simply the nominal control action is within $O(\epsilon^2)$ of the optimal expected cost-to-go. However, the plan with the optimal feedback gain K_t^* is strictly better than the open-loop plan in that it has a lower variance in terms of the cost to go. Furthermore, by solving the approximate closed-loop problem using the surrogate LQR problem, we expect a lower variance of the cost-to-go function due to feedback, which is borne out empirically (see Fig. 2.3).*

2.4 Decoupled Data-Based Control Algorithm 1.0

In this section, we propose a novel decoupled data-based control algorithm (D2C 1.0). The two-step framework to solve the stochastic feedback control problem may be summarized as follows: 1) solve the open-loop optimization problem using the gradient descent method with a blackbox simulation model of the dynamics, 2) identify the LTV from input-output experiment data, and design an LQR controller for the identified LTV system.

2.4.1 Open-Loop Trajectory Optimization

A first order gradient descent-based algorithm is proposed here for solving the open-loop optimization problem given in Eq. (2.14), where the underlying dynamic model is used as a blackbox, and the necessary gradient estimates are found from a sequence of input perturbation experiment data using standard least squares.

Denote the initial guess of the control sequence as $U^{(0)} = \{\bar{u}_t^{(0)}\}_{t=1}^T$, and the corresponding states $\mathcal{X}^{(0)} = \{\bar{x}_t^{(0)}\}_{t=1}^T$. The control policy is updated iteratively via

$$U^{(n+1)} = U^{(n)} - \gamma_n \nabla_U \bar{J}|_{\mathcal{X}^{(n)}, U^{(n)}}, \tag{2.19}$$

where $U^{(n)} = \{\bar{u}_t^{(n)}\}_{t=1}^T$ denotes the control sequence in the n^{th} iteration, $\mathcal{X}^{(n)} = \{\bar{x}_t^{(n)}\}_{t=1}^T$ denotes the corresponding states, and γ_n is the time-varying line search parameter. As $\bar{J}|_{\mathcal{X}^{(n)}, U^{(n)}}$ is the expected cumulative cost under control sequence $U^{(n)}$ and corresponding states $\mathcal{X}^{(n)}$, the gradient vector is defined as:

$$\nabla_U \bar{J}|_{\mathcal{X}^{(n)}, U^{(n)}} = \left(\frac{\partial \bar{J}}{\partial u_1} \quad \frac{\partial \bar{J}}{\partial u_2} \quad \cdots \quad \frac{\partial \bar{J}}{\partial u_T} \right) |_{\mathcal{X}^{(n)}, U^{(n)}}, \tag{2.20}$$

which is the gradient of the average cumulative cost w.r.t the control sequence after n iterations. The following paragraph elaborates on how to estimate the above gradient.

Let us define a rollout to be an episode in the simulation that starts from the initial settings to the end of the horizon with a control sequence. For each iteration, multiple rollouts are conducted sequentially with both the expected cumulative cost and the gradient vector updated iteratively after each rollout. During one iteration of the control sequence, the expected cumulative cost is calculated as

$$\bar{J}|^{j+1}_{\mathcal{X}^{(n)}, U^{(n)}} = (1 - \frac{1}{j}) \bar{J}|^{j}_{\mathcal{X}^{(n)}, U^{(n)}} + \frac{1}{j} (J|_{\mathcal{X}^{j,(n)}, U^{j,(n)}}), \tag{2.21}$$

where j denotes the j^{th} rollout within the current iteration process of control sequence. $\bar{J}|^{j}_{\mathcal{X}^{(n)}, U^{(n)}}$

is the expected cumulative cost after j rollouts while $J|_{\mathcal{X}^{j,(n)},U^{j,(n)}}$ denotes the cost of the j^{th} rollout under control sequence $U^{j,(n)}$ and corresponding states $\mathcal{X}^{j,(n)}$. Note that $U^{j,(n)} = \{\bar{u}_t^{(n)} + \delta u_t^{j,(n)}\}_{t=1}^T$ where $\{\delta u_t^{j,(n)}\}_{t=1}^T$ is the zero-mean, i.i.d Gaussian noise added as perturbation to the control sequence $U^{(n)}$. Then the gradient vector is calculated in a similar sequential manner as

$$\nabla_U \bar{J}|^{j+1}_{\mathcal{X}^{(n)},U^{(n)}} = (1 - \frac{1}{j})\nabla_U \bar{J}|^{j}_{\mathcal{X}^{(n)},U^{(n)}} + \frac{1}{j\sigma_{\delta u}}(J|_{\mathcal{X}^{j,(n)},U^{j,(n)}} - \bar{J}|^{j+1}_{\mathcal{X}^{(n)},U^{(n)}})(U^{j,(n)} - U^{(n)}), \tag{2.22}$$

where $\sigma_{\delta u}$ is the variance of the control perturbation and $\nabla_U \bar{J}|^{j+1}_{\mathcal{X}^{(n)},U^{(n)}}$ denotes the gradient vector after j rollouts. After m rollouts, the control sequence is updated by Eq. (2.19) in which $\nabla_U \bar{J}|_{\mathcal{X}^{(n)},U^{(n)}}$ is estimated by $\nabla_U \bar{J}|^{m}_{\mathcal{X}^{(n)},U^{(n)}}$, and the procedure repeated till convergence.

2.4.2 Linear Time-Varying System Identification

The closed-loop control design specified in Eq. (2.15) requires the knowledge of the parameters $A_t, B_t, 1 \leq t \leq T$, of the perturbed linear system. We propose a linear time-varying (LTV) system identification procedure to estimate these parameters.

First start from perturbed linear system given by Eq. (2.18). Using only first order information, we estimate the system parameters A_t, B_t in the LTV form: $\delta x_{t+1} = \hat{A}_t \delta x_t + \hat{B}_t \delta u_t$.

Now write out their components for each iteration in vector form as,

$$Y = [\delta x_{t+1}^0 \delta x_{t+1}^1 \cdots \delta x_{t+1}^{N-1}], \qquad X = \begin{bmatrix} \delta x_t^0 & \cdots & \delta x_t^{N-1} \\ \delta u_t^0 & \cdots & \delta u_t^{N-1} \end{bmatrix},$$
$$Y = [\hat{A}_t \mid \hat{B}_t]X, \tag{2.23}$$

where N is the total iteration number. δx_{t+1}^n denotes the output state deviation, δx_t^n denotes the input state perturbations and δu_t^n denotes the input control perturbations at time t of the n^{th} iteration. All the perturbations are zero-mean, i.i.d, Gaussian random vectors whose covariance matrix is σI where I is the identity matrix and σ is a scalar. Note that here one iteration only has one rollout.

Using the least squares method $\hat{A}_t$ and $\hat{B}_t$ can be calculated as follows:

$$[\hat{A}_t \mid \hat{B}_t] = YX'(XX')^{-1}, \tag{2.24}$$

The calculation procedure can also be done sequentially using recursive least squares. It is highly amenable to parallelization and is memory efficient.

2.4.3 Closed-Loop Control Design

Given the estimated perturbed linear system, we design a finite horizon, discrete time LQR [22] along the trajectory for each timestep to minimize the cost function $J = \delta x_T^T Q \delta x_T + \sum_{t=0}^{T-1} (\delta x_t^T Q \delta x_t + u_t^T R u_t + 2\delta x_t^T N u_t)$, subjects to $\delta x_{t+1} = \hat{A}_t \delta x_t + \hat{B}_t \delta u_t$, where $u_t = -K_t \delta x_t$. The feedback gains are calculated as $K_t = (R + B^T P_{t+1} B)^{-1}(B^T P_{t+1} A + N^T)$, where P_t is solved in a back propagation fashion from the Riccati equation: $P_{t-1} = A^T P_t A - (A^T P_t B + N)(R + B^T P_t B)^{-1}(B^T P_t A + N^T) + Q, P_T = Q, N = 0$. The closed-loop control policy is $u_t(x_t) = \bar{u}_t^* - K_t \delta x_t$, where δx_t is the state deviation vector from the nominal state at timestep t.

Algorithm 1: D2C 1.0 Algorithm

---

1) Solve the deterministic open-loop optimization problem for optimal open-loop nominal control sequence and trajectory $(\{\bar{u}_t^*\}_{t=1}^T, \{\bar{x}_t^*\}_{t=1}^T)$ using gradient descent method (Section 2.4.1).
2) Identify the LTV system $(\hat{A}_t, \hat{B}_t)$ via least square estimation (Section 2.4.2).
3) Solve the Riccati equations using estimated LTV system equation for feedback gain $\{K_t^*\}_{t=1}^T$ (Section 2.4.3).
4) Set $t = 1$, given initial state $x_1 = \bar{x}_1^*$ and state deviation $\delta x_1 = 0$.
while $t \leq T$ **do**

$$\begin{aligned} u_t &= \bar{u}_t^* + K_t^* \delta x_t, \\ x_{t+1} &= f(x_t, u_t) + \epsilon w_t, \\ \delta x_{t+1} &= x_{t+1} - \bar{x}_{t+1}^*, \end{aligned} \tag{2.25}$$

$t = t + 1$.
end while

2.4.4 Convergence of the D2C Algorithm 1.0

The D2C 1.0 algorithm is summarized in Algorithm 1. In the following, we provide a convergence analysis of the open-loop design and LTV identification parts of the D2C 1.0 algorithm.

Proposition 3. Gradient Descent. *Let the gradient $\nabla \bar{J}$ be Lipshitz continuous, i.e., $||\nabla \bar{J}(U_1) - \nabla \bar{J}(U_2)||| \leq L||U_1 - U_2||$, for some $L < \infty$, and the step size parameters in Eq.* (2.19) *satisfy $\sum_t \gamma_t = \infty$, and $\sum_t \gamma_t^2 < \infty$. Given $E||U^{(n)}||^2 < \infty$, the iterates in Eq.* (2.19)*, $U^{(n)}$ almost surely converge to a set S, where $\nabla \bar{J} = 0$ on the set S.*

Proof. The sample paths of the stochastic gradient descent algorithm Eq. (2.19) almost surely can be approximated asymptotically by the ODE $\dot{U} = -\nabla \bar{J}(U)$, due to the above assumptions and the fact that Eq. (2.22) is an unbiased estimator of the true gradient, and the convergence of the algorithm is almost surely determined by the limit points of the ODE (this follows from the so-called 'ODE method' approach to Stochastic Approximation algorithms [36]). To characterize the limit points, choose the Lyapunov function $\bar{J}$ for the above ODE, then $\dot{\bar{J}} = -\nabla \bar{J} \cdot \nabla \bar{J} \leq 0$. Hence, $\bar{J}$ converges to a set S where $\nabla \bar{J} = 0$, proving the result. □

Complexity of Stochastic Gradient Descent. The complexity of the gradient descent algorithm, per gradient descent step, is $O(pT)$, where p is the number of inputs, and T is the control horizon. However, due to the nonlinear cost function J, the convergence guarantees are only asymptotic, and the complexity of the whole algorithm is $O(K_\infty pT)$ where K_∞ is the steps to convergence which can vary with the initial guess and the learning parameter schedule.

Proposition 4. Convergence of LTV identification. *The least squares estimates in Eq.* (2.24)*, $[\hat{A}_t, \hat{B}_t] \to [A_t, B_t]$, as $N \to \infty$ in the mean square sense.*

Proof. Without loss of generality, let the $cov(\delta x_t^{(i)}) = I_n$, and $cov(\delta u_t^{(i)}) = I_p$, where I_q denotes a $q \times q$ identity matrix. The least squares solution in (2.24) can be written as $[\hat{A}_t, \hat{B}_t] = Y^{(N)} X^{(N)\intercal} (X^{(N)} X^{(N)\intercal})^{-1}$, where $Y^{(N)} = [\delta x_{t+1}^{(1)} ... \delta x_{t+1}^{(N)}]$, $X^{(N)} = [\delta \bar{x}_t^{(1)}, \cdots \delta \bar{x}_t^{(N)}]$, and $\delta \bar{x}_t^{(i)} = [\delta x_t^{(i)}, \delta u_t^{(i)}]^\intercal$.

Using the Law of Large numbers, it is relatively straightforward to see that $\frac{1}{N}X^{(N)}X^{(N)\intercal} \to I_{n+p}$ as $N \to \infty$ almost surely. Let the noise after N steps be $V^{(N)}$ (which is zero mean with covariance I_N), i.e., $Y^{(N)} = [A_t, B_t]X^{(N)} + V^{(N)}$. Then, the error in the LS estimate is $V^{(N)}X^{(N)\intercal}(X^{(N)}X^{(N)\intercal})^{-1}$, which is zero mean and has covariance $(X^{(N)}X^{(N)\intercal})^{-1} \to \frac{1}{N}I_{n+p} \to 0$ as $N \to \infty$. Therefore, the LS estimate converges in mean square sense to the true parameter values. □

Complexity of LTV identification. The key to the complexity of the above identification is how quickly does the sample covariance $\frac{1}{N}X^{(N)}X^{(N)\intercal} \to I_{n+p}$. It can be shown that $N = O(n + p)$ samples are good enough to get close to the limit with a very high probability if $n + p$ is large enough (Theorem 4.7.1 in [37]). We do not go into more details here due to space constraints but this is borne out by our empirical evidence. Thus, the complexity of the LTV identification is $O(n + p)T$ since we have T such identification steps.

Complexity of the D2C 1.0 algorithm. The complexity of the open-loop design is $O(K_\infty pT)$ while that of the LTV identification, and hence the closed-loop design, is $O(n + p)T$. However, in general, the steps to convergence, $K_\infty >> n, p$, and thus, the training time of the D2C 1.0 algorithm is overwhelmingly dominated by the open-loop part, an observation that is borne out by our empirical results that follow (see Table 2.1).

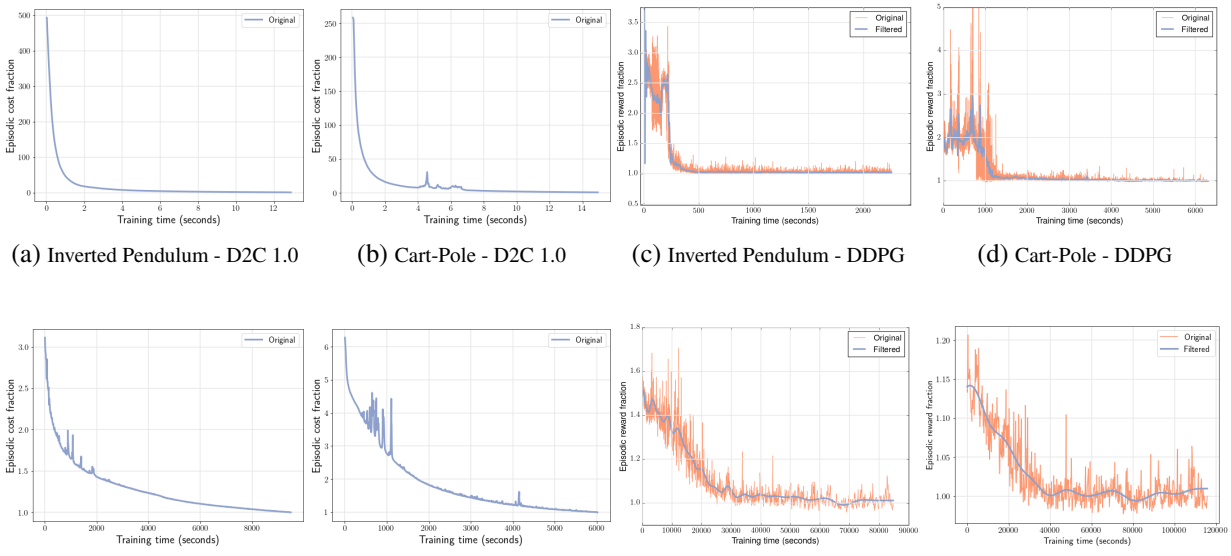
2.5 Empirical Results on Benchmarking Examples

In this section, we compare the D2C approach with the well-known deep reinforcement learning algorithm - Deep Deterministic Policy Gradient (DDPG) [9]. For comparison, we evaluate both methods in the following three aspects: 1) Efficiency in training - the amount of time and storage required to achieve a desired task, 2) Robustness to noise - the deviation from the predefined task due to random noise in the process in the testing stage, and 3) Ease of training - the challenges involved in training with either of the data-based approaches.

We tested our method with four benchmark tasks, all implemented in MuJoCo simulator [38]: Inverted pendulum, Cart-pole, 3-link swimmer, and 6-link swimmer (please see [39] for details). The

state space ranges from 2 to 26 dimensions while the control space ranges from 1 to 6 dimensions in these examples. An off-the-shelf implementation of DDPG provided by *Keras-RL* [40] library has been customized for our simulations. For a fair comparison, “episodic reward/cost fraction" is considered with both methods. It is defined as the fraction of reward obtained in an episode during training w.r.t the nominal episodic reward (converged reward).

Training Efficiency: One way of measuring efficiency is to collate the times taken for the episodic cost (or reward) to converge during training. Plots in Fig. 2.1 show the training process with both methods on the systems considered. Each plot shows the training curve of one experiment. The curve marked as original is the actual training curve reflecting the original reward data. The one marked as filtered is the curve after smoothing out the spikes to show a better view of the reward trend as the training goes. Table 2.1 delineates the times taken for training respectively. As the system identification and feedback gain calculation in the case of D2C take only a small portion of time, the total time comparison in (Table 2.1) shows that D2C learns the optimal policy substantially faster than DDPG, and hence, has a better training efficiency.

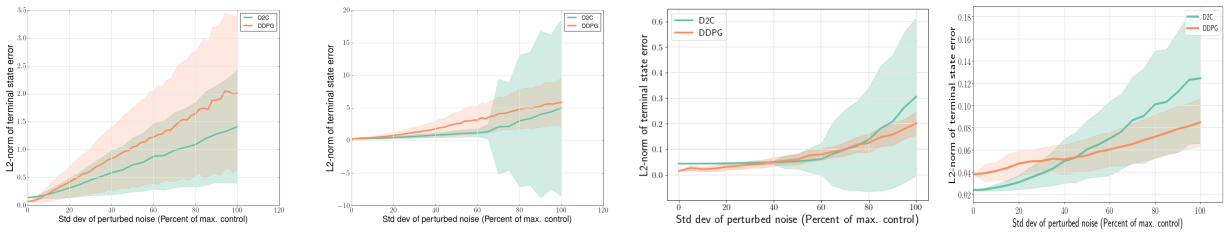


(a) Inverted Pendulum - D2C 1.0 (b) Cart-Pole - D2C 1.0 (c) Inverted Pendulum - DDPG (d) Cart-Pole - DDPG

(e) 6-link Swimmer - D2C 1.0 (f) Fish - D2C 1.0 (g) 6-link Swimmer - DDPG (h) Fish - DDPG

Figure 2.1: Episodic reward fraction vs time taken during training.

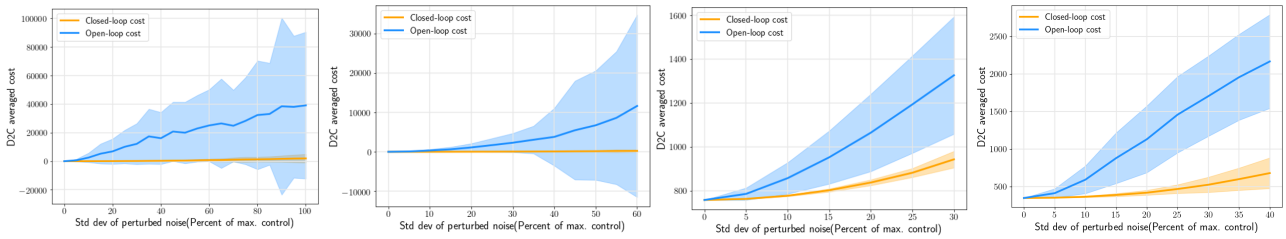
Robustness to noise: However, from plots in Fig. 2.2, it is evident that the performance of D2C 1.0 is on par with or better than DDPG up to a certain level of noise. It may also be noted that the error variance in the D2C method increases abruptly when the noise level is higher than a threshold and drives the system too far away from the nominal trajectory that the LQR controller cannot fix it. This could be considered a drawback for D2C. However, it must be noted that the range of noise levels (up until 100 % of the maximum control signal) that we are considering here is far beyond what is typically encountered in practical scenarios. Moreover, it must also be noted that at the point where the DDPG performance overtakes that of D2C, the performance of both methods is poor from the viewpoint of attaining the given task. In Fig. 2.3, we compare the episodic cost during testing between the open-loop policy applied along and the closed-loop policy of D2C 1.0. As expected, the closed-loop performance is much better than the open-loop performance albeit the closed-loop design is only a very small fraction of the training cost (see Table 2.1).



(a) Inverted Pendulum (b) Cart-Pole (c) 6-link Swimmer (d) Fish

Figure 2.2: Terminal MSE comparison of D2C 1.0 and DDPG during testing.

Ease of training: To elucidate the ease of training from an empirical perspective, the exploration noise that is required for training in DDPG mandates the system to operate with a shorter timestep than a threshold, beyond which the simulation fails due to an unbearable magnitude of control actions in the system. For this, we train both the swimmers in one such case (with $\Delta t = 0.01$ sec) till it fails and execute the intermediate policy. Figure 2.4 shows the plot in the testing stage with both methods. It is evident from the terminal state mean-squared error at zero noise level that the



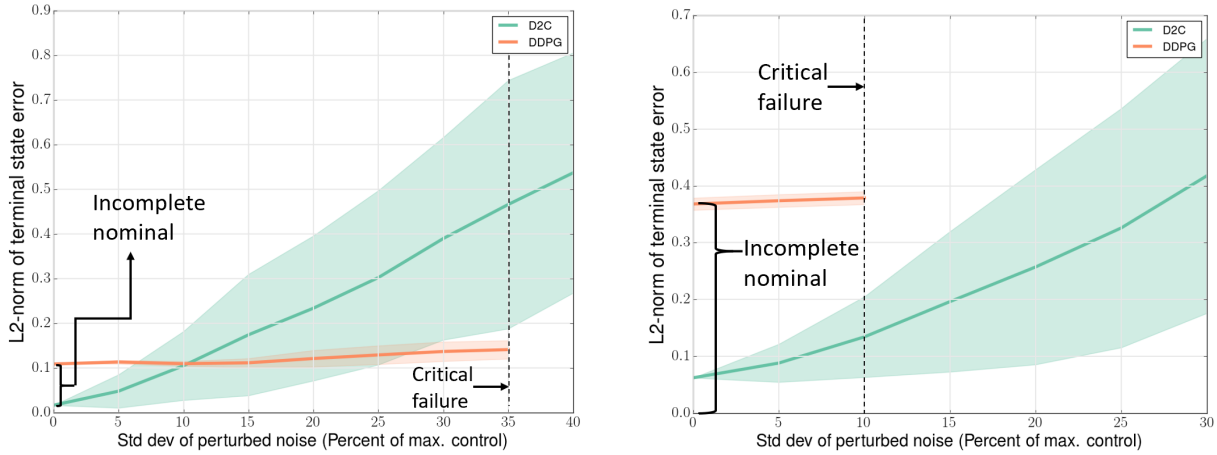
(a) Inverted Pendulum (b) Cart-Pole (c) 6-link Swimmer (d) Fish

Figure 2.3: Averaged episodic reward fraction vs noise level during testing for D2C 1.0.

nominal trajectory of DDPG is incomplete and its policy failed to reach the goal. The effect is more pronounced in the higher-dimensional 6-link swimmer system (Fig. 2.4b), where the DDPG's policy can be deemed to be downright broken. Note, from Table 2.1, that the systems have been trained with DDPG for a time that is more than thrice with the 3-link swimmer and 4 times with the 6-link swimmer. Moreover, the starred entries in Table I indicate that DDPG failed to converge. On the other hand, under the same conditions, the seamless training of D2C results in a working policy with even greater data efficiency.

Table 2.1: Simulation parameters and training outcomes.

System	Steps per episode	Time-step (in sec.)	Training time (in sec.)		
			D2C 1.0		DDPG
			Open-loop	Closed-loop	
Inverted Pendulum	30	0.1	12.9	< 0.1	2261.15
Cart pole	30	0.1	15.0	1.33	6306.7
3-link Swimmer	1600 800	0.005 0.01	7861.0 4001.0	13.1 4.6	38833.64 13280.7
6-link Swimmer	1500 900	0.006 0.01	9489.3 3585.4	26.5 16.4	88160 15797.2
Fish	1200	0.005	6011.2	75.6	124367.6



(a) 3-link swimmer

(b) 6-link swimmer

Figure 2.4: D2C 1.0 vs DDPG at $\Delta t = 0.01s$.

2.6 Empirical Results on Tensegrity Structures

As discussed in Chapter 1, the tensegrity structures have great potential in soft robotics. A systematic solution for controlling tensegrity robots can be of great value for soft robot development. Thus, in this section, we apply D2C 1.0 to the tensegrity structures and compare the performance with the MBC method. We first give the background of tensegrity structure design and control. Then we compared these methods in simulation and delineate their relative advantages and disadvantages. The details of the modeling and model-based control of high DOF tensegrity robotic systems are presented in the Appendix Section A.1 and A.2.

2.6.1 Background on Tensegrity Structures

The design of high DOF soft robotic systems has attracted increasing interest in recent years [41, 42]. In this regard, tensegrity structures offer a tantalizing prospect for the principled design of such soft robotic systems. As described in the introduction, the tensegrity structures also possess great advantages that traditional systems do not have in many real-world applications, such as minimal mass and adaptive stiffness. On the other hand, the complexity and the number of

dimensions of tensegrity structures greatly increase the difficulty to control such systems. Thus the tensegrity structures are good application examples to test the D2C 1.0 approach.

Tensegrity structures are designed by placing bars and strings in a methodical arrangement to yield certain desired properties [1]. The dynamics of a tensegrity system can be very accurately modeled as all the members in the system are 1-dimensional elements that only take uni-directional loading [43]. The absence of bending moments on any individual element not only allows for the accurate modeling of the system but also provides the minimum mass solution to various kinds of loading conditions in engineering mechanics [1, 44]. For the same shape, the stiffness of the structure can also be changed by changing the pre-tension in the strings. The minimal mass architecture along with the variable stiffness characteristic makes it suitable for soft robotic applications like planetary landers [45], flexible robots [46, 47], and deployable space structures [48]. Some of the researchers used model-based approaches [46, 47] and some used learning/evolutionary algorithm-based approaches [45, 49, 50] to control the tensegrity structures but no discussion has been given in the past to compare the two methods.

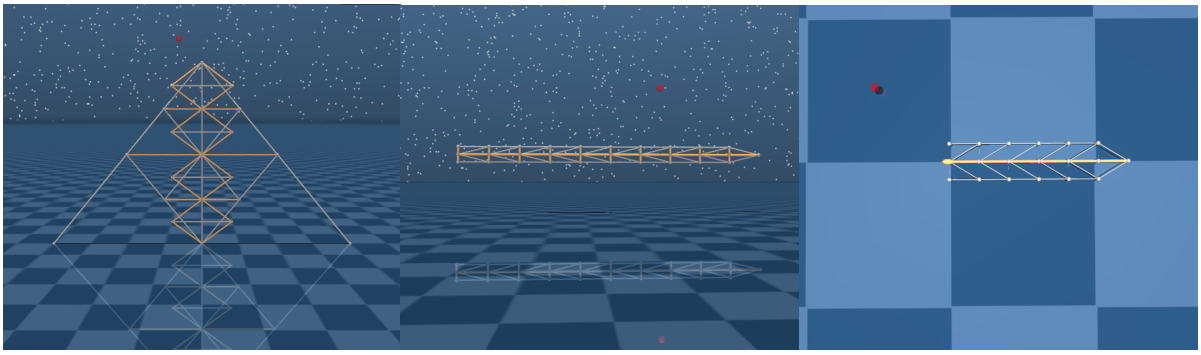
The control of tensegrity systems amounts to the design of a nonlinear stochastic controller for a very high DOF complex nonlinear system. The classical literature provides rigorous analysis of the asymptotic performance and stability of the closed-loop system, mostly for linear systems or finite state and control space systems. The optimal control of a possibly unknown nonlinear dynamical system with continuous state and action space is a significantly more challenging problem and suffers from computational complexity issues, in addition to the usual identifiability problems of adaptive control. For the model-based methods, the computational time is often negligible if the analytical model is known.

2.6.2 Empirical Results

In this subsection, we present the simulation results of both the model-based shape control and the data-based D2C 1.0 approach applied to three tensegrity robotic systems. The states are the angles and angular velocities of the connecting hinge joints between bars. The control inputs are the tension applied to the strings.

2.6.2.1 *Structures and Tasks*

We simulated all three robotic models in the MuJoCo simulator [38]: Reacher, Arm and 6-link swimmer. Each of the systems and their tasks are defined as follows:



(a) Reacher (b) Arm (c) 6-link Swimmer

Figure 2.5: Models simulated in MuJoCo in their initial states.

Reacher: The minimal mass solution to compressive loading is provided by T-bar structure [1]. Tensegrity D-bar structure has also been shown to require less mass than a simple continuum bar to take the same load with the added advantage of deployability which makes it suitable for large motion space robotic applications. A T2D1 tensegrity structure is made by combining T-bar and D-bar structures as shown in Fig. 2.5(a). The two-dimensional structure has 22 bars and 22 strings. The bars in the system are connected by hinge joints reducing the degrees of freedom to 14 and the system is controlled by controlling the tension in the 22 control channels (strings). The control task is to move the top tip (end effector) to the target position (red dot).

Arm: The arm model is composed of 10 T-bar elements, made by rigidly attaching a vertical bar with a horizontal bar. The horizontal bars are aligned in the direction of the arm length. The first vertical bar is fixed as the base of the model. The model has 18 dimensions of states and 38 control channels. The control task is to move the tip to the target position (red dot).

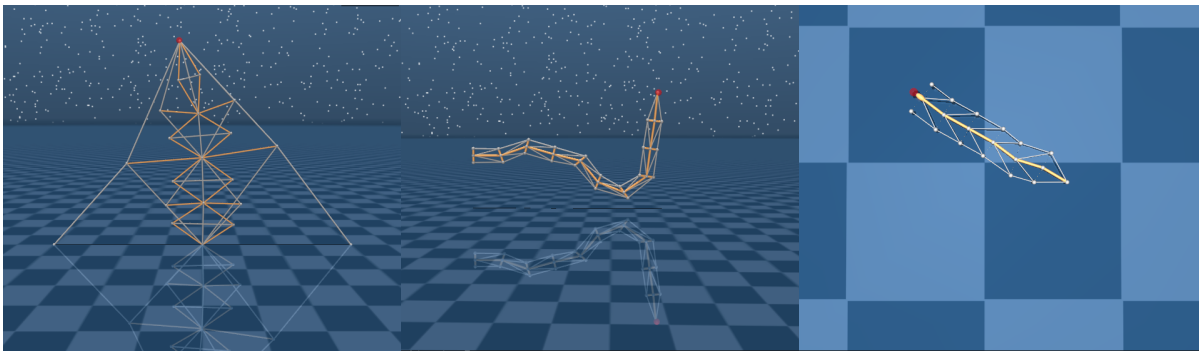
6-link Swimmer: The 6-link swimmer is composed of 6 T-bar elements. The T-bar element

here has the same structure as the arm model. Compared with the arm model, the first bar for the swimmer is not fixed so it could swim in the environment. The fluid density for the environment is 3000 kg/m^3, three times the density of water. The model swims by swaying its body so the horizontal bars would interact with the fluid to generate a forward force. Note that the diameter of the vertical bars is very small (difficult to see in the figure) so that they won't generate too much resistance while swimming. The model has 16 state variables and 22 control channels. The control task is to swim to the target position (red dot).

The initial positions of the models are shown in Fig. 2.5. The orange objects are the bars, and the grey ones are the strings.

2.6.2.2 Training and Testing

D2C implementation is done in three stages corresponding to those mentioned in the previous section and 'MuJoCo Pro 200 C++ version' is used as the environment for simulating the blackbox model. The MBC approach does not require any training time since it is a closed-form solution.

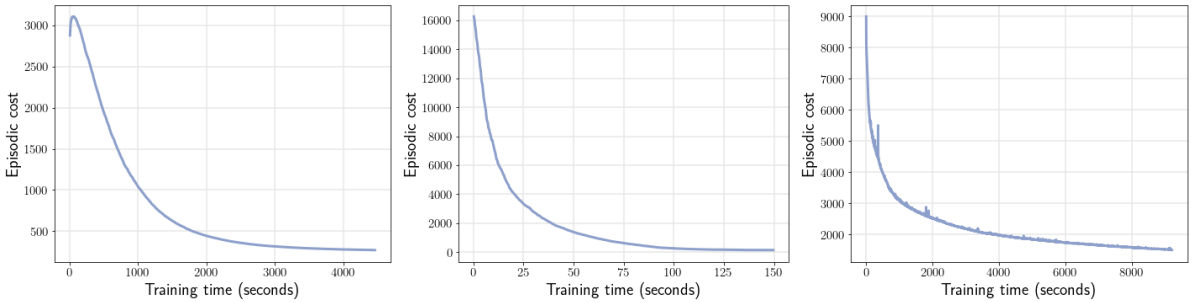


(a) Reacher (b) Arm (c) 6-link Swimmer

Figure 2.6: Models simulated in MuJoCo in their terminal states.

Training (D2C only): The open-loop training plots in Fig. 2.7 show the cost curves during training. After the cost curves converge, we get the optimal control sequence that could drive the systems to accomplish their tasks. The respective model positions at the end of the horizon are

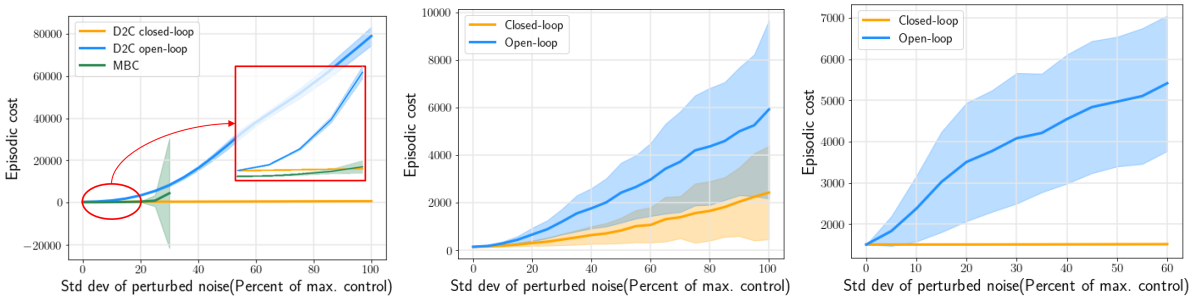
shown in Fig. 2.6. The training parameters and outcomes are summarized in (Table 2.2).



(a) Reacher (b) Arm (c) 6-link Swimmer

Figure 2.7: Episodic cost vs time taken during D2C 1.0 open-loop training.

Testing Criterion: For the closed-loop design, we proceed with the system identification and feedback gain design step of the D2C 1.0 algorithm mentioned in the previous section to get the closed-loop control policy. For testing, we compare the performance between the open-loop D2C 1.0 control policy and the closed-loop D2C 1.0 control policy under different noise levels. We also compare the D2C 1.0 closed-loop performance to the MBC design for the Reacher example. The MBC design cannot be applied to the arm or the swimmer because of the moment of the elements and solid-fluid interaction. The open-loop control policy is to apply the optimal control sequence solved in the open-loop training step without any feedback. So the perturbation drives the model off the nominal trajectory and increases the episodic cost as the noise level increases. Zero-mean Gaussian independent identically distributed (i.i.d.) noise is added to every control channel at each step. The standard deviation of the noise is proportional to the maximum control signal in the designed optimal control sequence for D2C 1.0. As for MBC, we use the maximum control value in the noiseless nominal control sequence. It must be noted that the range of noise levels (at least up until about 60% of the maximum nominal control signal) that we are considering here is far beyond what is typically encountered in practical scenarios. As for the criterion for performance, we use episodic cost at each noise level. 500 rollouts are simulated at every noise level tested.



(a) Reacher (b) Arm (c) 6-link Swimmer

Figure 2.8: Performance comparison between D2C 1.0 open-loop and closed-loop control policy.

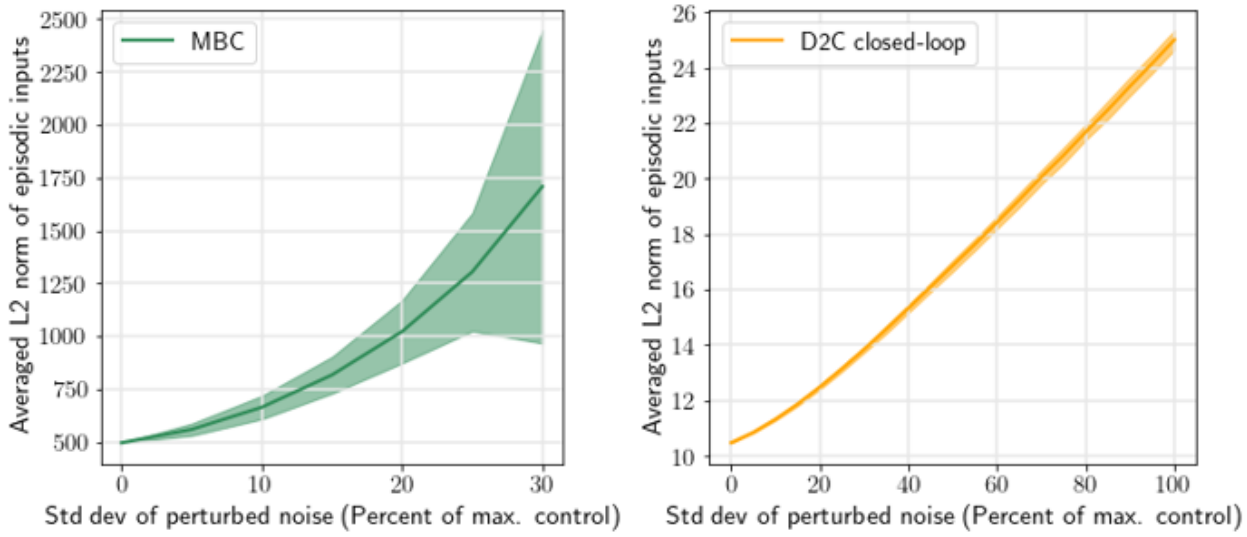


Figure 2.9: Control energy comparison between MBC and D2C 1.0.

Robustness to Noise: From Fig. 2.8(a), we can see that both the mean episodic cost and the cost variance of the closed-loop D2C 1.0 policy are much smaller than that of the open-loop policy for the reacher example throughout the noise level range shown in the plots which prove the success of using D2C 1.0 on tensegrity models. Although the MBC design can only take up to 30% noise before the simulation fails, it has similar performance to the D2C 1.0 closed-loop policy when applicable. For the D2C approach, the swimmer example fails after 60% noise, while the reacher and the arm example worked up to 100% noise level. When there is no noise, the analytical solution

has the lowest cost of all three. From Fig. 2.9, it is clear that the control energy used by the MBC solution is way larger than that of D2C 1.0, where the episodic energy is calculated as the L2-norm of the control sequence. This is the result of the optimization-based formulation of D2C 1.0. Also, MBC takes up to 30% noise and has a higher variance than D2C 1.0, which aligns with what is shown in Fig. 2.8(a). With less control effort, the D2C 1.0 policy gets a feedback policy close to the accuracy of the analytical solution and endures a wider range of noise. The closed-loop cost for all three models increases as the noise level increases. If we keep increasing the noise level, there exists a threshold that the noise will drive the system too far away from the nominal trajectory that the LQR controller cannot fix it. However, till that point, the closed-loop policy always performs better than the open-loop policy (Fig. 2.8).

Discussion: The advantage of the MBC approach is that, whenever applicable, it has negligible training time when compared to a data-based approach like D2C 1.0. However, the D2C 1.0 approach can achieve the same performance as the model-based approach with far less control effort. One of the reasons for this might be that the reference system needed by the MBC design has not been designed optimally, and thus, a more careful design might result in control efforts that are on par with D2C 1.0. In a sense, the philosophies followed by D2C 1.0 and MBC are quite similar in that a reference nominal system to be tracked is designed in both, which is then tracked using the closed-loop control. However, D2C 1.0 obtains this via optimizing a suitable finite horizon cost of the nonlinear system whereas the MBC approach provides a somewhat arbitrary prescribed behavior, which, in turn, needs much more control effort in order to be tracked. As mentioned above, the arm and swimmer models are difficult to model due to increased complexity and solid-fluid interaction, but the data-based D2C approach can still be applied to these models, indicating a wider application potential for such data-based control design approaches.

2.7 Conclusions

To solve the stochastic optimal control problem described in Eq. (2.1), we proposed a near-optimal control algorithm D2C 1.0 under fully observed conditions and showed that our method is able to scale up to a higher dimensional state-space with unknown system dynamics. Due to

Table 2.2: D2C 1.0 training parameters and outcomes.

System	Steps per episode	Timestep (in sec.)	Rollout number	Iteration number	Training time (in sec.)	
					Open-loop	Closed-loop
Reacher	400	0.01	300	1300	4462.1	4.25
Arm	400	0.01	20	600	149.5	4.14
Swimmer	1500	0.006	50	7000	9204.0	5.83

[1] The open-loop training is run on a laptop with I7-7700HQ CPU and 8G RAM. No multi-threading.

the sequential calculation used in the open-loop optimization and the system identification, D2C 1.0 is highly memory efficient and also convenient for parallelization. We tested its performance and compared it with a state-of-the-art deep RL technique - DDPG. From the results, D2C 1.0 has significant advantages over DDPG in terms of training efficiency and ease of training. This primarily stems from the far smaller parameter space, essentially open-loop sequences, that D2C 1.0 searches over, as opposed to a complex parameterization like deep neural networks for DDPG. The feedback of D2C 1.0 is designed such that the closed-loop trajectory is kept close to the nominal trajectory. The robustness of this design is shown to be better/ comparable with DDPG in most cases but has scope for further improvement by employing a more sophisticated feedback design and ensuring that the data efficiency is not compromised. Also, further reduction in the training time can be achieved by parallelization and a more efficient solution to the open-loop problem. We also tested D2C 1.0 on three tensegrity structures and compared it with the MBC method. The MBC method takes negligible time to generate the control law since the system dynamics are known. D2C 1.0 is more energy efficient as the control energy is considered in the optimization. Also, the D2C 1.0 control policy is more robust to noise. The following chapters will focus on improving these aspects and demonstrating the performance of the D2C technique on more complex, nonlinear high-dimensional examples.

3. DECOUPLED DATA-BASED CONTROL 2.0: A SECOND ORDER APPROACH*

3.1 Introduction

Given the promising result of D2C 1.0, there remains space to improve especially in the open-loop optimal trajectory design step. The gradient descent method used in D2C 1.0 can converge to a local minimum with proper line search, but the convergence is slow because it only utilizes the first order information of the cost function. As the open-loop optimization step takes the majority of the total training time, it is crucial to improve the training efficiency and convergence. In this chapter, with a data-base extension of iLQR and further analysis of the decoupling principle, we propose the D2C 2.0 algorithm which features higher training efficiency and global optimality compared to D2C 1.0. We prove that the decoupled design is near-optimal to the fourth order and D2C 2.0 is guaranteed to converge to the unique global optimum.

Recently, there has been a growing class of literature on “deep reinforcement learning" owing to the progress made in the function approximation using deep neural networks [9, 51, 52]. These algorithms improved the performance of RL in terms of efficiency and ease of training. However, a systematic approach is still lacking. *The issues with RL can be attributed to the typically complex parametrization of the global feedback policy, and the related fundamental question of what this feedback parametrization ought to be?*

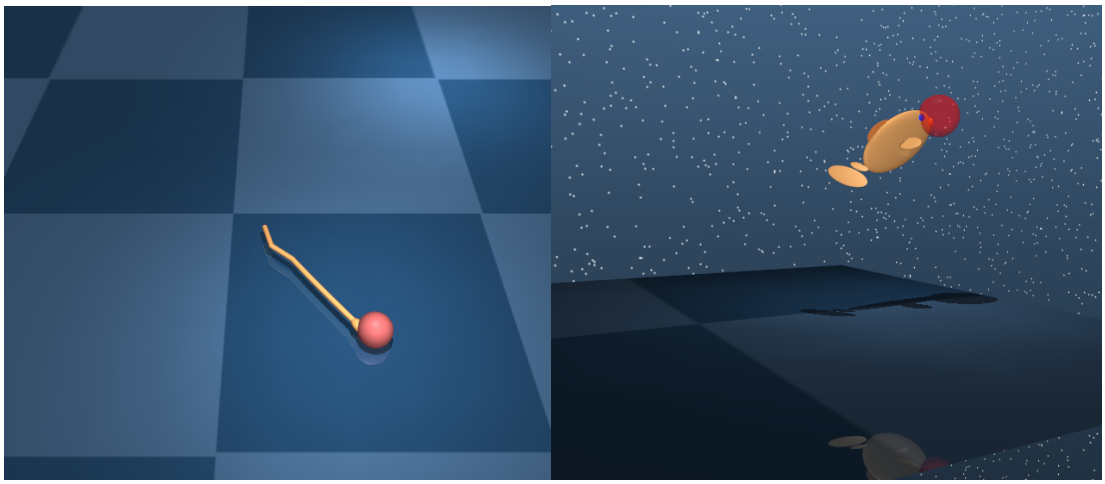
We advocate that for RL to be a) efficient in training, b) reliable in its result, and c) robust to noise, one needs to use a local feedback parametrization as opposed to a global parametrization. Further, this local feedback parametrization consists of an open-loop control sequence combined with a linear feedback law around the nominal open-loop sequence. Searching over this parametrization is highly efficient when compared to the global RL search, and can be shown to reliably converge to the global optimum while having performance that is superior to the global RL solution. In particular, this search is sufficiently fast and reliable that one can recover the optimal global feedback law by

*Part of this chapter is reprinted with permission from "On the search for feedback in reinforcement learning" by R. Wang, K. S. Parunandi, A. Sharma, R. Goyal and S. Chakravorty, 2021. 60th IEEE Conference on Decision and Control (CDC), pp. 1560-1567, Copyright 2021 by IEEE.

replanning whenever necessary. The sole caveat is that these claims are true for a deterministic, albeit unknown, system. However, we show that: 1) theoretically, the deterministic optimal feedback law is near-optimal to fourth order in a small noise parameter to the optimal stochastic law, and 2) empirically, RL methods have difficulty in learning on stochastic systems, so much so that most RL algorithms typically find a feedback law for the deterministic system in which the only noise is an asymptotically vanishing exploration noise.

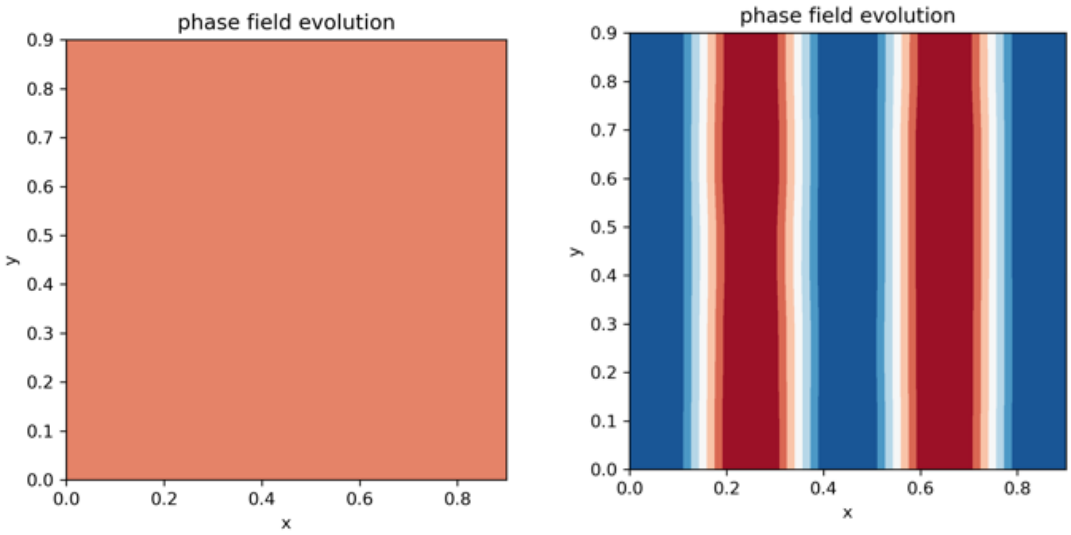
In general, the methods to solve optimal control problems for unknown dynamical systems can be divided into two broad classes, local and global as discussed in Section 2.1. The most computationally efficient among these techniques are “local" trajectory-based methods such as DDP [11, 29] and iLQR [12, 13]. In fact, it was never established that these local approaches to RL are highly efficient, globally optimum, and extremely reliable (insignificant variance), when compared to the global approaches. Further, the local approaches are superior in performance in terms of robustness to noise, i.e., they have better “global" performance compared to global approaches. The reliability of iLQR comes from its guaranteed convergence to the global optimum: we show that iLQR is a Sequential Quadratic Programming (SQP) approach to the optimal control scenario, using which we show that under relatively mild assumptions, iLQR converges to the global minimum from any initial guess. Thus, we can expect that iLQR always yields the same optimal result from different runs. Further, we establish that such local approaches can recover the optimal global feedback law when coupled with replanning, whenever necessary as in MPC [53, 15], which becomes feasible because of the highly efficient and reliable local search that is guaranteed to converge to a globally optimum open-loop solution, and the associated optimal linear feedback law.

Our primary contribution in this chapter is to show that performing RL via a local feedback parametrization, is highly efficient and reliable, in terms of low variance, when compared to the global approaches. They are also superior in terms of robustness to noise, i.e., their performance is better “globally" than the global methods (see Fig. 3.2). Also, the optimal global feedback law can be recovered by replanning, which is made feasible via the fast and reliable local planner.



(a) 6-link Swimmer

(b) Fish



(c) Initial state

(d) Final state

Figure 3.1: Models controlled in this section using D2C 2.0 including multi-body systems with fluid-structure interactions, and a material microstructure model governed by the Allen-Cahn phase field partial differential equation.

To achieve this goal, we propose the D2C 2.0 algorithm which uses a data-based extension of iLQR to replace the first order gradient descent method. ILQR is a Newton-like second order optimization method similar to DDP. They both have faster convergence than first order methods. Further, iLQR only uses linearized system dynamics, and thus is more efficient. For the data-based

extension, we use an efficient randomized least square procedure to estimate the linearized time-varying system parameters from simulated rollouts of the system. As for the convergence and optimality, we first show the equivalence between SQP and the iLQR algorithm used in D2C. Then we notice that under mild conditions, iLQR is guaranteed to globally converge to a stationary point starting from any initial point. Combined with results in paper [54], where due to the Method of Characteristics (MOC) development, if the dynamics are control affine and the cost is quadratic in control, the global minimum is assured as iLQR satisfies the necessary conditions of optimality. Thus, the open-loop optimization with iLQR is guaranteed to globally converge to the global minimum. Note that the linear feedback law of D2C is found locally around a nominal open-loop sequence, which can be combined with T-PFC to conduct fast and reliable replanning when the linear feedback law fails to keep the state close to the nominal trajectory. Simulation results demonstrate the improved robustness compared to D2C without replanning and RL methods.

This chapter is organized as follows. Section 3.2 outlines the problem formulation. Section 3.3 derives the $O(\epsilon^4)$ near-optimal decoupling principle. In Section 3.4, we establish the iLQR-based decoupled data-based control algorithm (D2C 2.0) and prove its convergence to the global optimum. Its performance is demonstrated in terms of training efficiency, training reliability and robustness to noise in simulation experiments on typical benchmarking examples with comparisons to D2C 2.0 with replanning and state-of-the-art RL methods in Section 3.5.

3.2 Problem Formulation

Consider the following discrete time nonlinear stochastic dynamical system: $x_{t+1} = F(x_t, u_t, w_t)$, where $x_t \in \mathbb{R}^{n_x}$, $u_t \in \mathbb{R}^{n_u}$ are the state measurement and control vector at time t, respectively. The process noise w_t is assumed as zero-mean, uncorrelated Gaussian white noise, with covariance W. The *stochastic optimal control* problem is to find the the control policy $\pi^o = \{\pi_0^o, \pi_2^o, \cdots, \pi_{T-1}^o\}$ such that the expected cumulative cost is minimized, i.e., $\pi^o = \arg\min_\pi \ \tilde{J}^\pi(x)$, where, $\tilde{J}^\pi(x) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} c_t(x_t, u_t) + c_T(x_T)|x_0 = x\right]$, $u_t = \pi_t(x_t)$, $c_t(\cdot,\cdot)$ is the instantaneous cost function, and $c_T(\cdot)$ is the terminal cost function. In the following, we assume that the initial state x_0 is fixed, and denote $\tilde{J}^\pi(x_0)$ simply as $\tilde{J}^\pi$.

3.3 An $O(\epsilon^4)$ Near-Optimal Decoupling Principle

For the problem described in 2.1, directly solving for the stochastic optimal policy can be difficult and computationally intractable. Instead, it is noticed that the optimal open-loop policy obtained in the deterministic system wrapped with the optimal linear feedback law is a good approximation to the true stochastic optimal policy, i.e., the deterministic optimal policy and the stochastic optimal policy agree locally up to $O(\epsilon^4)$. Further, the open-loop nominal design is found to be independent of the linear feedback design, which suggests the following decoupled procedure to find the local optimal feedback policy: 1) solving for the open-loop optimal policy where $\epsilon = 0$ and 2) solving for the optimal linear feedback policy along the open-loop nominal trajectory that minimizes the cost variance. In the following, we outline an $O(\epsilon^4)$ near-optimal decoupling principle in stochastic

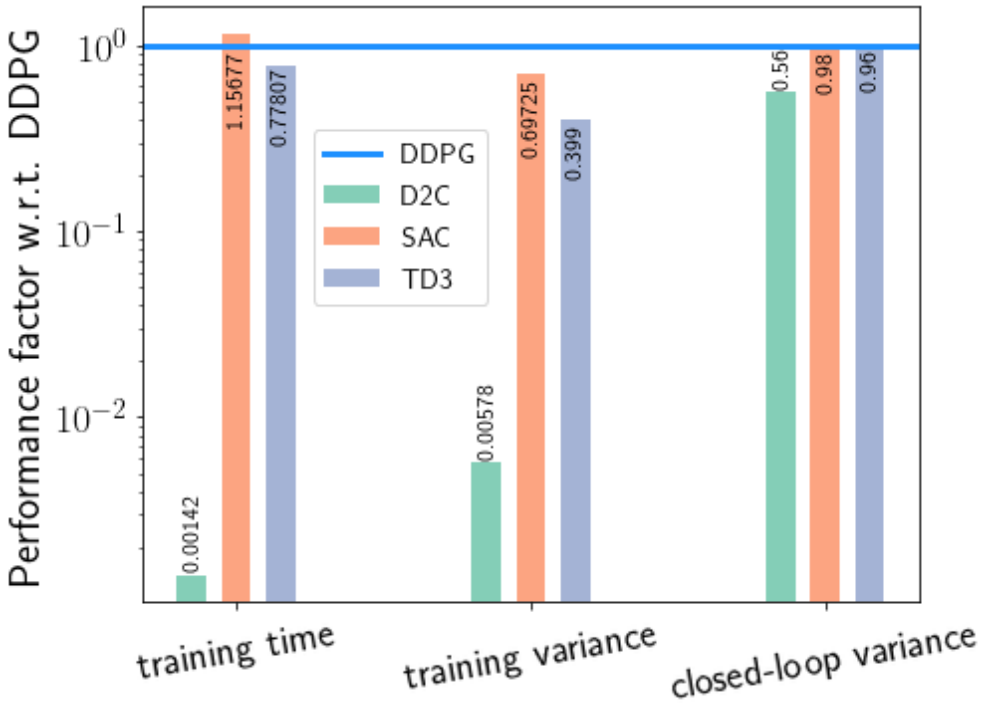


Figure 3.2: Local RL approach (D2C 2.0) vs global RL approaches.
The statistics shown above, found by averaging over all the models simulated, show that although TD3 and SAC have improvements over DDPG, the local approach is still highly efficient (training time), reliable (training variance), while also having superior closed-loop performance (closed-loop variance), when compared to the global RL approaches.

optimal control that paves the way for the D2C 2.0 algorithm described in Section 3.4.

Let the dynamics be given by:

$$x_t = x_{t-1} + \bar{f}(x_{t-1})\Delta t + \bar{g}(x_{t-1})u_t\Delta t + \epsilon\omega_t\sqrt{\Delta t}, \tag{3.1}$$

where ω_t is a white noise sequence, and the sampling time Δt is small enough that the $O(\Delta t^\alpha)$ terms are negligible for $\alpha > 1$. $\bar{f}(\cdot)$ and $\bar{g}(\cdot)$ are nonlinear functions of the state. The noise term above stems from Brownian motion, and hence the $\sqrt{\Delta t}$ factor. Further, the incremental cost function $c(x, u)$ is given as: $c_t(x, u) = \bar{l}_t(x)\Delta t + \frac{1}{2}u'\bar{R}u\Delta t$. Then, we have the following results. Given sufficient regularity, any feedback policy can then be represented as: $\pi_t(x_t) = \bar{u}_t + K_t^1\delta x_t + \delta x_t' K_t^2\delta x_t + \cdots$, where $\bar{u}_t$ is the nominal action with associated nominal state $\bar{x}_t$, i.e., action under zero noise, and $K_t^1, K_t^2, \cdots$ represent the linear and higher order feedback gains acting on the state deviation from the nominal: $\delta x_t = x_t - \bar{x}_t$, due to the noise.

Proposition 5. *The cost function of the optimal stochastic policy, J_t, and the cost function of the 'deterministic policy applied to the stochastic system', φ_t, satisfy: $J_t(x) = J_t^0(x) + \epsilon^2 J_t^1(x) + \epsilon^4 J_t^2(x) + \cdots$, and $\varphi_t(x) = \varphi_t^0(x) + \epsilon^2\varphi_t^1(x) + \epsilon^4\varphi_t^2(x) + \cdots$. Furthermore, $J_t^0(x) = \varphi_t^0(x)$, and $J_t^1 = \varphi_t^1(x)$, for all t, x.*

Proof. We show the result for the scalar case for simplicity (and completeness). The DP equation for the given system is given by:

$$J_t(x) = \min_{u_t}\{c_t(x, u_t) + E[J_{t+1}(x')]\}, \tag{3.2}$$

where $x' = x + \bar{f}(x)\Delta t + \bar{g}(x)u_t\Delta t + \epsilon\omega_t\sqrt{\Delta t}$ and $J_t(x)$ denotes the cost-to-go of the system given that it is at state x at time t. The above equation is marched back in time with terminal condition $J_T(x) = c_T(x)$, and $c_T(\cdot)$ is the terminal cost function. Let $u_t(\cdot)$ denote the corresponding optimal policy. Then, it follows that the optimal control u_t satisfies (since the argument to be minimized is

quadratic in u_t)

$$u_t = -R^{-1}\bar{g}' J^x_{t+1}, \tag{3.3}$$

where $J^x_{t+1} = \frac{\partial J_{t+1}}{\partial x}$.

We know that any cost function, and hence, the optimal cost-to-go function can be expanded in terms of ϵ as:

$$J_t(x) = J^0_t + \epsilon^2 J^1_t + \epsilon^4 J^2_t + \cdots . \tag{3.4}$$

Thus, substituting the minimizing control in Eq. (3.3) into the dynamic programming Eq. (3.2) implies:

$$\begin{aligned} J_t(x) = \bar{l}_t(x)\Delta t + \frac{1}{2} r(\frac{-\bar{g}}{r})^2 (J^x_{t+1})^2 \Delta t + J^x_{t+1}\bar{f}(x)\Delta t \\ + \bar{g}(\frac{-\bar{g}}{r})(J^x_{t+1})^2\Delta t + \frac{\epsilon^2}{2} J^{xx}_{t+1}\Delta t + J_{t+1}(x), \end{aligned} \tag{3.5}$$

where J^x_t, and J^{xx}_t denote the first and second derivatives of the cost-to go function. Substituting Eq. (3.4) into Eq. (3.5) we obtain that:

$$\begin{aligned} (J^0_t + \epsilon^2 J^1_t + \epsilon^4 J^2_t + \cdots) = \bar{l}_t(x)\Delta t + \frac{1}{2}\frac{\bar{g}^2}{r}(J^{0,x}_{t+1} + \epsilon^2 J^{1,x}_{t+1} + \cdots)^2 \Delta t \\ + (J^{0,x}_{t+1} + \epsilon^2 J^{1,x}_{t+1} + \cdots)\bar{f}(x)\Delta t \\ - \frac{\bar{g}^2}{r}(J^{0,x}_{t+1} + \epsilon^2 J^{1,x}_{t+1} + \cdots)^2\Delta t \\ + \frac{\epsilon^2}{2}(J^{0,x}_{t+1} + \epsilon^2 J^{1,x}_{t+1} + \cdots)\Delta t + J_{t+1}(x). \end{aligned} \tag{3.6}$$

Now, we equate the ϵ^0, ϵ^2 terms on both sides to obtain perturbation equations for the cost functions $J^0_t, J^1_t, J^2_t, \cdots$.

First, let us consider the ϵ^0 term. Utilizing Eq. (3.6) above, we obtain:

$$J^0_t = \bar{l}_t\Delta t + \frac{1}{2}\frac{\bar{g}^2}{r}(J^{0,x}_{t+1})^2\Delta t + \underbrace{(\bar{f} + \bar{g}\frac{-\bar{g}}{r}J^{0,x}_t)}_{\bar{f}^0} J^{0,x}_t \Delta t + J^0_{t+1}, \tag{3.7}$$

with the terminal condition $J_T^0 = c_T$, and where we have dropped the explicit reference to the argument of the functions x for convenience.

Similarly, one obtains by equating the $O(\epsilon^2)$ terms in Eq. (3.6) that:

$$J_t^1 = \frac{1}{2}\frac{\bar{g}^2}{r}(2J_{t+1}^{0,x}J_{t+1}^{1,x})\Delta t + J_{t+1}^{1,x}\bar{f}\Delta t - \frac{\bar{g}^2}{r}(2J_{t+1}^{0,x}J_{t+1}^{1,x})\Delta t + \frac{1}{2}J_{t+1}^{0,xx}\Delta t + J_{t+1}^1, \tag{3.8}$$

which after regrouping the terms yields:

$$J_t^1 = \underbrace{(\bar{f} + \bar{g}\frac{-\bar{g}}{r}J_{t+1}^{0,x})}_{=\bar{f}^0} J_{t+1}^{1,x}\Delta t + \frac{1}{2}J_{t+1}^{0,xx}\Delta t + J_{t+1}^1, \tag{3.9}$$

with terminal boundary condition $J_T^1 = 0$. Note the perturbation structure of Eqs. (3.7) and (3.9), J_t^0 can be solved without knowledge of J_t^1, J_t^2 etc, while J_t^1 requires knowledge only of J_t^0, and so on. In other words, the equations can be solved sequentially rather than simultaneously.

Now, let us consider the deterministic policy $u_t^d(\cdot)$ that is a result of solving the deterministic DP equation:

$$\phi_t(x) = \min_{u_t^d}[c_t(x, u_t^d) + \phi_{t+1}(x')], \tag{3.10}$$

where $x' = x + \bar{f}\Delta t + \bar{g}u_t^d\Delta t$, i.e., the deterministic system obtained by setting $\epsilon = 0$ in Eq. (3.1), and ϕ_t represents the optimal cost-to-go of the deterministic system. Analogous to the stochastic case, $u_t^d = \frac{-\bar{g}}{r}\phi_{t+1}^x$. Next, let φ_t denote the cost-to-go of the deterministic policy $u_t^d(\cdot)$ *when applied to the stochastic system, i.e., Eq.* (3.1) *with* $\epsilon > 0$. Then, the cost-to-go of the deterministic policy, when applied to the stochastic system, satisfies:

$$\varphi_t = c_t(x, u_t^d(x)) + E[\varphi_{t+1}(x')], \tag{3.11}$$

where $x' = x + \bar{f}\Delta t + \bar{g}u_t^d\Delta t + \epsilon\sqrt{\Delta t}\omega_t$. Substituting $u_t^d(\cdot) = \frac{-\bar{g}}{r}\phi_t^x$ into the equation above

implies that:

$$\begin{aligned}\varphi_t &= \varphi_t^0 + \epsilon^2\varphi_t^1 + \epsilon^4\varphi_t^2 + \cdots \\ &= \bar{l}_t\Delta t + \frac{1}{2}\frac{\bar{g}^2}{r}(\phi_{t+1}^x)^2\Delta t + (\varphi_{t+1}^{0,x} + \epsilon^2\varphi_{t+1}^{1,x} + \cdots)\bar{f}\Delta t \\ &+ \bar{g}\frac{-\bar{g}}{r}\phi_{t+1}^x(\varphi_{t+1}^{0,x} + \epsilon^2\varphi_{t+1}^{1,x} + \cdots)\Delta t \\ &+ \frac{\epsilon^2}{2}(\varphi_{t+1}^{0,xx} + \epsilon^2\varphi_{t+1}^{1,xx} + \cdots)\Delta t + (\varphi_{t+1}^0 + \epsilon^2\varphi_{t+1}^1 + \cdots). \end{aligned} \tag{3.12}$$

As before, if we gather the terms for ϵ^0, ϵ^2 etc. on both sides of the above equation, we shall get the equations governing φ_t^0, φ_t^1 etc. First, looking at the ϵ^0 term in Eq. (3.12), we obtain:

$$\varphi_t^0 = \bar{l}_t\Delta t + \frac{1}{2}\frac{\bar{g}^2}{r}(\phi_{t+1}^x)^2\Delta t + (\bar{f} + \bar{g}\frac{-\bar{g}}{r}\phi_{t+1}^x)\varphi_{t+1}^{0,x}\Delta t + \varphi_{t+1}^0, \tag{3.13}$$

with the terminal boundary condition $\varphi_T^0 = c_T$. However, the deterministic cost-to-go function also satisfies:

$$\phi_t = \bar{l}_t\Delta t + \frac{1}{2}\frac{\bar{g}^2}{r}(\phi_{t+1}^x)^2\Delta t + (\bar{f} + \bar{g}\frac{-\bar{g}}{r}\phi_{t+1}^x)\phi_{t+1}^x\Delta t + \phi_{t+1}, \tag{3.14}$$

with terminal boundary condition $\phi_T = c_T$. Comparing Eqs. (3.13) and (3.14), it follows that $\phi_t = \varphi_t^0$ for all t. Further, comparing them to Eq. (3.7), it follows that $\varphi_t^0 = J_t^0$, for all t. Also, note that the closed-loop system above, $\bar{f} + \bar{g}\frac{-\bar{g}}{r}\phi_{t+1}^x = \bar{f}^0$ (see Eq. (3.7) and (3.9)).

Next let us consider the ϵ^2 terms in Eq. (3.12). We obtain:

$$\varphi_t^1 = \bar{f}\varphi_{t+1}^{1,x}\Delta t + \bar{g}\frac{-\bar{g}}{r}\phi_{t+1}^x\varphi_{t+1}^{1,x}\Delta t + \frac{1}{2}\varphi_{t+1}^{0,xx} + \varphi_{t+1}^1.$$

Noting that $\phi_t = \varphi_t^0$, implies that (after collecting terms):

$$\varphi_t^1 = \bar{f}^0\varphi_{t+1}^{1,x}\Delta t + \frac{1}{2}\varphi_{t+1}^{0,xx}\Delta t + \varphi_{t+1}^1, \tag{3.15}$$

with terminal boundary condition $\varphi_T^1 = 0$. Again, comparing Eq. (3.15) to (3.9), and noting that $\varphi_t^0 = J_t^0$, it follows that $\varphi_t^1 = J_t^1$, for all t. This completes the proof of the result. □

Remark 4. *The result above shows that the first two terms in the perturbation expansion are identical for the optimal deterministic and optimal stochastic policies, when acting on the stochastic system, given they both start at state x at time t. This essentially means that the optimal deterministic policy and the optimal stochastic policy agree locally up to order $O(\epsilon^4)$.*

Remark 5. *It may also be shown that for the optimal deterministic policy, the ϵ^0 term, φ_t^0, in the cost, stems from the nominal action ($\bar{u}_t$) of the control policy, the ϵ^2 term, φ_t^1, stems from the linear feedback action of the closed-loop (K_t^1), while the higher-order terms stem from the higher-order terms in the feedback law.*

An important practical consequence of Proposition 5 is that we can get $O(\epsilon^4)$ near-optimal performance, by wrapping the optimal linear feedback law around the nominal control sequence ($u_t^* = \bar{u}_t + K_t \delta x_t$), where δx_t is the state deviation from the nominal $\bar{x}_t$ state, and replanning the nominal sequence when the deviation is sufficiently large (for convenience, we have dropped the superscript 1 in denoting the linear feedback gain above). This is similar to the event-driven MPC philosophy of [55, 56]. In the deterministic optimal control problem, the open-loop ($\bar{u}_t$) design is independent of the closed-loop design (K_t) which suggests the following 'decoupled' procedure to find the optimal feedback law (locally), i.e., we may first design the optimal open-loop sequence and then find the optimal linear feedback corresponding to said open-loop sequence.

Open-Loop Design: First, we design an optimal (open-loop) control sequence $\bar{u}_t^*$ for the noiseless system by solving

$$(\bar{u}_t^*)_{t=0}^{T-1} = \arg \min_{(\bar{u}_t)_{t=0}^{T-1}} \sum_{t=0}^{T-1} c_t(\bar{x}_t, \bar{u}_t) + c_T(\bar{x}_T), \tag{3.16}$$

with $\bar{x}_{t+1} = \bar{x}_t + \bar{f}(\bar{x}_t)\Delta t + \bar{g}(\bar{x}_t)\bar{u}_t \Delta t$. Denote $\mathcal{F}(x) = x + \bar{f}(x)\Delta t$ and $\mathcal{G}(x) = \bar{g}(x)\Delta t$ with reference to Eq. (3.1). The global optimum for this open-loop problem can be found by satisfying

the necessary conditions of optimality, as was shown in the first part of [54] using the Method of Characteristics.

Closed-Loop Design: The optimal linear feedback gain K_t corresponding to the nominal trajectory above is calculated as shown in Proposition 6. In the following, $A_t = \frac{\partial \mathcal{F}}{\partial x}|_{\bar{x}_t} + \frac{\partial \mathcal{G}\bar{u}_t}{\partial x}|_{\bar{x}_t}$, $B_t = \mathcal{G}(\bar{x}_t)$, $L_t^x = \frac{\partial l_t}{\partial x}|'_{\bar{x}_t}$ and $L_t^{xx} = \nabla^2_{xx} l_t|_{\bar{x}_t}$. Let $\phi_t(x_t)$ denote the optimal cost-to-go of the detrministic problem, i.e., Eq. (3.1) with $\epsilon = 0$.

Proposition 6. *Given an optimal nominal trajectory $(\bar{x}_t, \bar{u}_t)$, the backward evolutions of the first and second derivatives, $G_t = \frac{\partial \phi_t}{\partial x}|'_{\bar{x}_t}$ and $P_t = \nabla^2_{xx}\phi_t|_{\bar{x}_t}$, of the optimal cost-to-go function $\phi_t(x_t)$, initiated with the terminal boundary conditions $G_T = \frac{\partial c_T}{\partial x}|'_{\bar{x}_T}$ and $P_T = \nabla^2_{xx} c_T|_{\bar{x}_T}$ respectively, are as follows:*

$$G_t = L_t^x + G_{t+1}A_t, \tag{3.17}$$

$$P_t = L_t^{xx} + A_t' P_{t+1} A_t - K_t' R_t K_t + G_{t+1} \otimes \tilde{R}_{t,xx} \tag{3.18}$$

for $t = \{0, 1, ..., T-1\}$, where,

$$K_t = -R_t^{-1}(B_t' P_{t+1} A_t + (G_{t+1} \otimes \tilde{R}_{t,xu})'), \tag{3.19}$$

$\tilde{R}_{t,xx} = \nabla^2_{xx}\mathcal{F}(x_t)|_{\bar{x}_t} + \nabla^2_{xx}\mathcal{G}(x_t)|_{\bar{x}_t}\bar{u}_t$, $\tilde{R}_{t,xu} = \nabla^2_{xu}(\mathcal{F}(x_t) + \mathcal{G}(x_t)u_t)|_{\bar{x}_t,\bar{u}_t}$, *where ∇^2_{xx} represents the Hessian of a vector-valued function w.r.t x, similar for ∇^2_{xu}, and $\otimes$ denotes the kronecker product.*

Proof. Consider the Dynamic Programming equation for the deterministic cost-to-go function:

$$\phi_t(x_t) = \min_{u_t} Q_t(x_t, u_t) = \min_{u_t}\{c_t(x_t, u_t) + \phi_{t+1}(x_{t+1})\}$$

By Taylor's expansion about the nominal state at time $t+1$,

$$\phi_{t+1}(x_{t+1}) = \phi_{t+1}(\bar{x}_{t+1}) + G_{t+1}\delta x_{t+1} + \frac{1}{2}\delta x_{t+1}{}' P_{t+1}\delta x_{t+1} + q_{t+1}(\delta x_{t+1}),$$

where $q_{t+1}(\cdot)$ denotes the higher order terms. Substituting the perturbation expansion of the dynamics, $\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t + r_t(\delta x_t, \delta u_t)$ in the above expansion, where $r_t(\cdot)$ denotes the linearization residual,

$$\begin{aligned}\phi_{t+1}(x_{t+1}) = \phi_{t+1}(\bar{x}_{t+1}) + G_{t+1}(A_t \delta x_t + B_t \delta u_t + r_t(\delta x_t, \delta u_t)) \\ + \frac{1}{2}(A_t \delta x_t + B_t \delta u_t + r_t(\delta x_t, \delta u_t))' P_{t+1}(A_t \delta x_t + B_t \delta u_t + r_t(\delta x_t, \delta u_t)) + q_{t+1}(\delta x_{t+1}).\end{aligned} \tag{3.20}$$

Similarly, expand the incremental cost at time t about the nominal state, where $s_t(\cdot)$ denotes the higher order terms,

$$\begin{aligned}c_t(x_t, u_t) = \bar{l}_t + L_t^x \delta x_t + \frac{1}{2}\delta x_t{}' L_t^{xx} \delta x_t + \frac{1}{2}\delta u_t{}' R_t \bar{u}_t \\ + \frac{1}{2}\bar{u}_t' R_t \delta u_t + \frac{1}{2}\delta u_t{}' R_t \delta u_t + \frac{1}{2}\bar{u}_t' R_t \bar{u}_t + s_t(\delta x_t).\end{aligned} \tag{3.21}$$

$$\begin{aligned}Q_t(x_t, u_t) = \overbrace{[\bar{l}_t + \frac{1}{2}\bar{u}_t' R_t \bar{u}_t + \phi_{t+1}(\bar{x}_{t+1})]}^{\bar{\phi}_t(\bar{x}_t, \bar{u}_t)} + L_t^x \delta x_t + \frac{1}{2}\delta x_t{}' L_t^{xx} \delta x_t + \delta u_t{}'(B_t' \frac{P_{t+1}}{2} B_t + \frac{1}{2} R_t)\delta u_t \\ + \delta u_t{}'(B_t' \frac{P_{t+1}}{2} A_t \delta x_t + \frac{1}{2} R_t \bar{u}_t + B_t' \frac{P_{t+1}}{2} r_t) \\ + (\delta x_t{}' A_t' \frac{P_{t+1}}{2} B_t + \frac{1}{2}\bar{u}_t R_t + r_t' \frac{P_{t+1}}{2} B_t + G_{t+1} B_t)\delta u_t \\ + \delta x_t{}' A_t' \frac{P_{t+1}}{2} A_t \delta x_t + \delta x_t{}' A_t' \frac{P_{t+1}}{2} r_t + (r_t' \frac{P_{t+1}}{2} A_t + G_{t+1} A_t)\delta x_t \\ + r_t' \frac{P_{t+1}}{2} r_t + G_{t+1} r_t + q_{t+1} + s_t \equiv \bar{\phi}_t(\bar{x}_t, \bar{u}_t) + H_t(\delta x_t, \delta u_t).\end{aligned} \tag{3.22}$$

Now,

$$\min_{u_t} Q_t(x_t, u_t) = \min_{\bar{u}_t} \bar{\phi}_t(\bar{x}_t, \bar{u}_t) + \min_{\delta u_t} H_t(\delta x_t, \delta u_t). \tag{3.23}$$

First order optimality: Along the optimal nominal control sequence $\bar{u}_t$, it follows from the

minimum principle that

$$\frac{\partial c_t(x_t, u_t)}{\partial u_t} + \frac{\partial g(x_t)}{\partial u_t}' \frac{\partial \phi_{t+1}(x_{t+1})}{\partial x_{t+1}} = 0$$

$$\Rightarrow R_t \bar{u}_t + B_t' G_{t+1}' = 0. \tag{3.24}$$

By setting $\frac{\partial H_t(\delta x_t, \delta u_t)}{\partial \delta u_t} = 0$, we get:

$$\begin{aligned}\delta u_t^* = -(B_t' P_{t+1} B_t + R_t + B_t' P_{t+1}(\tilde{R}_{t,xu} \otimes \delta x_t))^{-1}(R_t \bar{u}_t \\ + B_t' G_{t+1}' + (B_t' P_{t+1} A_t + (G_{t+1} \otimes \tilde{R}_{t,xu})')\delta x_t \\ + B_t' P_{t+1} r_t + (r_t' P_{t+1} + \delta x_t{}' A_t' P_{t+1})(\tilde{R}_{t,xu} \otimes \delta x_t)).\end{aligned}$$

By neglecting the Δt^2 terms,

$$\begin{aligned}\delta u_t^* = \underbrace{-R_t^{-1}(B_t' P_{t+1} A_t + (G_{t+1} \otimes \tilde{R}_{t,xu})')}_{K_t} \delta x_t \\ \underbrace{-R_t^{-1}(B_t' P_{t+1} r_t + (r_t' P_{t+1} + \delta x_t{}' A_t' P_{t+1})(\tilde{R}_{t,xu} \otimes \delta x_t))}_{p_t}\end{aligned}$$

$$\Rightarrow \delta u_t^* = K_t \delta x_t + p_t,$$

where p_t is the second and higher order terms w.r.t. δx_t. Substituting it in the expansion of ϕ_t and regrouping the terms based on the order of δx_t (till 2^{nd} order), we obtain:

$$\begin{aligned}\phi_t(x_t) = \bar{\phi}_t(\bar{x}_t) + (L_t^x + (R_t \bar{u}_t + B_t' G_{t+1}')K_t + G_{t+1} A_t)\delta x_t \\ + \frac{1}{2}\delta x_t{}'(L_t^{xx} + A_t' P_{t+1} A_t - K_t' R_t K_t + G_{t+1} \otimes \tilde{R}_{t,xx})\delta x_t.\end{aligned}$$

Expanding the LHS about the nominal state results in the recursive equations in Proposition 6. □

3.4 Decoupled Data-Based Control Algorithm 2.0

Based on the decoupling principle, we establish the decoupled data-based control (D2C) 2.0 algorithm. We detail the open-loop trajectory design using iLQR and prove its convergence to the global optimum. Also, the data-based extension to iLQR and the closed-loop design of D2C 2.0 are presented below.

3.4.1 Open-Loop Trajectory Design via ILQR

We present an iLQR [13] based method to solve the open-loop optimization problem. The iLQR iteration which is identical to the SQP iteration is shown in Lemma 3 and the algorithm is outlined in Algorithms 2, 3 and 4. As for the optimality and convergence of iLQR, We prove that iLQR is guaranteed to converge to the global minimum of the optimal control problem under mild assumptions as shown in Theorem 1 and 2.

Lemma 3. *Assuming the cost function to minimize is quadratic in control, i.e.:*

$$\begin{aligned} &\min_{\mathbf{u}} J(\mathbf{x},\mathbf{u}) = \sum_{t=0}^{T-1}(l_t(x_t) + \frac{1}{2}u_t' R u_t) + c_T(x_T), \\ &s.t. \quad x_{t+1} = f(x_t, u_t),\ t = 0, 1, \ldots, T-1, \end{aligned} \tag{3.25}$$

where $l_t(\cdot)$ and $c_T(\cdot)$ are the incremental and terminal state cost functions, $\mathbf{x}$ and $\mathbf{u}$ are the state and control trajectories. The iLQR iteration for the above problem is identical to the SQP iteration.

Proof. First, we derive the SQP solution. The Lagrangian function of the nonlinear programming (NLP) problem in Eq. (3.25) can be written as:

$$\mathcal{L}(\mathbf{x},\mathbf{u},\lambda) = J(\mathbf{x},\mathbf{u}) + \sum_{t=0}^{T-1}\lambda_{t+1}'(x_{t+1} - f(x_t,u_t)). \tag{3.26}$$

Expanding the cost to second order and the constraint to first order yields the QP problem:

$$\begin{aligned}\min_{\delta\mathbf{u}} \delta J(\delta\mathbf{x}, \delta\mathbf{u}) = \sum_{t=0}^{T-1}[l'_{t,x}\delta x_t + \frac{1}{2}\delta x'_t l_{t,xx}\delta x_t + \bar{u}'_t R\delta u_t + \frac{1}{2}\delta u'_t R\delta u_t] \\ + c'_{T,x}\delta x_T + \frac{1}{2}\delta x'_T c_{T,xx}\delta x_T, \\ s.t. \quad \delta x_{t+1} = f_{x_t}\delta x_t + f_{u_t}\delta u_t,\ t = 0, 1, \ldots, T-1,\end{aligned} \tag{3.27}$$

where $\bar{x}_t$ and $\bar{u}_t$ are the resulting state and control of the previous SQP iteration. $l_{t,x} = \frac{\partial l_t}{\partial x}|_{\bar{x}_t}$, $l_{t,xx} = \nabla^2_{xx} l_t|_{\bar{x}_t}$, $c_{T,x} = \frac{\partial c_T}{\partial x}|_{\bar{x}_T}$ and $c_{T,xx} = \nabla^2_{xx} c_T|_{\bar{x}_T}$. The equality constraints are the linearized dynamics where $f_{x_t} = \frac{\partial f}{\partial x}|_{\bar{x}_t}$ and $f_{u_t} = \frac{\partial f}{\partial u}|_{\bar{u}_t}$. Then the Lagrangian function can be estimated with:

$$\mathcal{L}(\delta\mathbf{x}, \delta\mathbf{u}, \lambda) = \delta J(\delta\mathbf{x}, \delta\mathbf{u}) + \sum_{t=0}^{T-1} \lambda'_{t+1}(\delta x_{t+1} - f_{x_t}\delta x_t - f_{u_t}\delta u_t). \tag{3.28}$$

By satisfying the first order necessary conditions w.r.t. x_t, u_t and λ_t, we have,

$$\lambda_t = -l_{t,x} - l_{t,xx}\delta x_t + f'_{x_t}\lambda_{t+1} \tag{3.29a}$$

$$\delta u_t = R^{-1} f'_{u_t}\lambda_{t+1} - \bar{u}_t \tag{3.29b}$$

$$\delta x_t = f_{x_{t-1}}\delta x_{t-1} - f_{u_{t-1}}\delta u_{t-1}, \tag{3.29c}$$

with boundary condition, $\lambda_T = -c_{T,x} - c_{T,xx}\delta x_T$. Let us now show that the Lagrange multipliers have the form $\lambda_t = -v_t - V_t\delta x_t$ for all t. This is trivially true at the terminal timestep where $v_T = c_{T,x}$ and $V_T = c_{T,xx}$. Suppose that $\lambda_{t+1} = -v_{t+1} - V_{t+1}\delta x_t$ for timestep $t+1$, we start with Eq. (3.29b) and (3.29c) at time $t+1$ and substitute for λ_{t+1}:

$$\begin{aligned}\delta x_{t+1} &= f_{x_t}\delta x_t + f_{u_t}(R^{-1} f'_{u_t}\lambda_{t+1} - \bar{u}_t) \\ &= f_{x_t}\delta x_t + f_{u_t}(R^{-1} f'_{u_t}(-v_{t+1} - V_{t+1}\delta x_{t+1}) - \bar{u}_t).\end{aligned} \tag{3.30}$$

By collecting terms and solving for δx_{t+1} yield:

$$\delta x_{t+1} = (I + f_{u_t} R^{-1} f'_{u_t} V_{t+1})^{-1} (f_{x_t} \delta x_t - f_{u_t} R^{-1} f'_{u_t} v_{t+1} - f_{u_t} \bar{u}_t). \tag{3.31}$$

Substituting λ_{t+1} in Eq. (3.29a) with the propagation form and δx_{t+1} with Eq. (3.31):

$$\begin{aligned} \lambda_t = -l_{t,x} - l_{t,xx} \delta x_t + f'_{x_t} (-v_{t+1} - V_{t+1} ((I + f_{u_t} R^{-1} f'_{u_t} V_{t+1})^{-1} \\ \cdot (f_{x_t} \delta x_t - f_{u_t} R^{-1} f'_{u_t} v_{t+1} - f_{u_t} \bar{u}_t))). \end{aligned} \tag{3.32}$$

After separating zero and first order terms w.r.t. δx_t, we can show that at timestep t, $\lambda_t = -v_t - V_t \delta x_t$, where v_t and V_t can be solved as:

$$v_t = l_{t,x} + f'_{x_t} v_{t+1} - f'_{x_t} V_{t+1} f_{u_t} (R + f'_{u_t} V_{t+1} f_{u_t})^{-1} \cdot (f'_{u_t} v_{t+1} + R \bar{u}_t) \tag{3.33a}$$

$$\begin{aligned} V_t &= l_{t,xx} + f'_{x_t} (V_{t+1}^{-1} + f_{u_t} R^{-1} f'_{u_t})^{-1} f_{x_t} \\ &= l_{t,xx} + f'_{x_t} V_{t+1} f_{x_t} - f'_{x_t} V_{t+1} f_{u_t} (R + f'_{u_t} V_{t+1} f_{u_t})^{-1} \cdot f'_{u_t} V_{t+1} f_{x_t}. \end{aligned} \tag{3.33b}$$

Therefore the Lagrangian multiplyer has the form $\lambda_t = -v_t - V_t \delta x_t$ for all t. Then, by substituting Eq. (3.32) in Eq. (3.29b), we can solve for δu_t:

$$\begin{aligned} \delta u_t &= R^{-1} f'_{u_t} (-v_{t+1} - V_{t+1} (f_{x_t} \delta x_t + f_{u_t} \delta u_t)) - \bar{u}_t \\ &= -(R + f'_{u_t} V_{t+1} f_{u_t})^{-1} (R \bar{u}_t + f'_{u_t} v_{t+1} + f'_{u_t} V_{t+1} f_{x_t} \delta x_t). \end{aligned} \tag{3.34}$$

which can be written in the linear feedback form $\delta u_t = -k_t - K_t \delta x_t$, where $k_t = (R + f'_{u_t} V_{t+1} f_{u_t})^{-1} (R \bar{u}_t + f'_{u_t} v_{t+1})$ and $K_t = (R + f'_{u_t} V_{t+1} f_{u_t})^{-1} f'_{u_t} V_{t+1} f_{x_t}$.

Comparing the above results with iLQR [14], it is clear that the control update δu_t in SQP and iLQR are the same. In SQP, the update at time t is $(\delta x_t, \delta u_t)$, in which δx_t can be recursively solved from Eq. (3.29c), with $\delta x_0 = 0$ and $\delta u_t = -k_t - K_t \delta x_t$. In the forward pass of iLQR, the state is updated as $\delta x_{t+1} = f(\bar{x}_t + \delta x_t, \bar{u}_t + \delta u_t) - \bar{x}_{t+1}$. Thus the iLQR iterations are always feasible.

Suppose the updates are small such that the linearization of the dynamics is valid,

$$\begin{aligned}\bar{x}_{t+1} + \delta x_{t+1} &= f(\bar{x}_t, \bar{u}_t) + f_{x_t}\delta x_t + f_{u_t}\delta u_t \\ \delta x_{t+1} &= f_{x_t}\delta x_t + f_{u_t}\delta u_t,\end{aligned} \tag{3.35}$$

which is the same as the SQP iteration in Eq. (3.29c). Therefore, it follows that the iLQR iterations are identical to the SQP iterations for the above optimal control problem. □

To show the convergence of the iLQR algorithm, we list the key assumptions that are needed in the following proof. **A1:** The control cost R is chosen to be positive definite for all timesteps, i.e., there exists a constant $\beta_1 > 0$ such that $\mathbf{d}'R\mathbf{d} \geq \beta_1\|\mathbf{d}\|^2$ for any $\mathbf{d}$.
A2: The cost functions $l_t(\cdot)$ and $c_T(\cdot)$ are chosen such that $\{l_{t,xx}\}$ and $\{c_{T,xx}\}$ are uniformly bounded and positive semi-definite for all t, i.e., there exists a constant $\beta_2 > 0$ such that for each k, $\|l_{t,xx}\| \leq \beta_2$, $\|c_{T,xx}\| \leq \beta_2$, $\mathbf{d}'l_{t,xx}\mathbf{d} \geq 0$ and $\mathbf{d}'c_{T,xx}\mathbf{d} \geq 0$ for all t and any $\mathbf{d}$.
A3: The starting point and all succeeding iterates lie in some compact set $\mathcal{C}$.

Lemma 4. *Under assumptions **A1**, **A2** and **A3**, if $(\mathbf{x}, \mathbf{u})$ is not a stationary point of the NLP problem in Eq.* (3.25)*, the iLQR iteration update $\mathbf{d_s} = [\delta\mathbf{x}' \;\; \delta\mathbf{u}']'$ is a descent direction for the cost function $J(\mathbf{x}, \mathbf{u})$.*

Proof. Denote the constraint from the system dynamics and its gradient as,

$$\begin{aligned}h(\bar{x}_{t+1}, \bar{x}_t, \bar{u}_t) &= \bar{x}_{t+1} - f(\bar{x}_t, \bar{u}_t) \\ \nabla h(\bar{x}_{t+1}, \bar{x}_t, \bar{u}_t) &= [I \;\; -f_{x_t} \;\; -f_{u_t}].\end{aligned} \tag{3.36}$$

By satisfying the first order necessary condition,

$$d'_{s,t}\nabla J'_t = -d'_{s,t}\nabla h'(\bar{x}_{t+1}, \bar{x}_t, \bar{u}_t)\lambda_{t+1} - d'_{s,t}B_t d_{s,t}, \tag{3.37}$$

where B_t is the Hessian of the cost function and $d_{s,t} = [\delta x'_{t+1} \;\; \delta x'_t \;\; \delta u'_t]'$. The first term on the RHS is zero from Eq. (3.35). Using **A2**,

$$\begin{aligned} d'_{s,t} B_t d_{s,t} &= [\delta x'_{t+1} \;\; \delta x'_t \;\; \delta u'_t] \begin{bmatrix} l_{t+1,xx} & 0 & 0 \\ 0 & l_{t,xx} & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} \delta x_{t+1} \\ \delta x_t \\ \delta u_t \end{bmatrix} \\ &= \delta x'_{t+1} l_{t+1,xx} \delta x_{t+1} + \delta x'_t l_{t,xx} \delta x_t + \delta u'_t R \delta u_t \\ &> 0, \end{aligned} \tag{3.38}$$

as δu_t can not be zero before reaching a stationary point and R is chosen to be positive definite. Then, it follows that $d'_{s,t} \nabla J'_t < 0$ for all t, thus the iterations of iLQR always give a descent direction for the cost function. □

Theorem 1. *Under assumptions **A1**, **A2** and **A3**, with the line search method in Algorithm 2 and 3, the iLQR algorithm started at any initial point is guaranteed to converge to a stationary point of the optimization problem in Eq.* (3.25).

Proof. According to the line search method given in Algorithm 2 and 3, as well as the fact that all the iterations are feasible, i.e., $h(x_{t+1}, x_t, u_t) = 0$ for all t, the chosen stepsize α satisfies,

$$\frac{J(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) - J(\mathbf{x}^k, \mathbf{u}^k)}{\Delta J(\alpha)} \geq \sigma_1 > 0, \tag{3.39}$$

where $[\mathbf{x}^{k+1'}, \mathbf{u}^{k+1'}]' = [\mathbf{x}^{k'}, \mathbf{u}^{k'}]' + \alpha \mathbf{d_s^k}$ and $\Delta J(\alpha)$ is defined as $\alpha \nabla J(\mathbf{x}^k, \mathbf{u}^k)' \mathbf{d_s^k}$. In the algorithms below, we write $\Delta J(\alpha)$ as $\Delta cost(\alpha)$. Note that due to our backtracking line search method, α is lower bounded away from zero for all iterations [57]. Substituting for $\Delta J(\alpha)$, Eq. (3.39) can be rewritten as,

$$\begin{aligned} J(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) &\leq J(\mathbf{x}^k, \mathbf{u}^k) + \sigma_1 \alpha \nabla J(\mathbf{x}^k, \mathbf{u}^k)' \mathbf{d_s^k} \\ &\leq J(\mathbf{x}^k, \mathbf{u}^k) - \beta_3 |cos\theta| \|\nabla J(\mathbf{x}^k, \mathbf{u}^k)\| \|\mathbf{d_s^k}\|, \end{aligned} \tag{3.40}$$

where β_3 is a positive constant, θ is the angle between $\nabla J(\mathbf{x}^k, \mathbf{u}^k)$ and $\mathbf{d_s^k}$. Since Lemma 4 shows that $\mathbf{d_s^k}$ is always a descent direction, $|cos\theta|$ and $\|\mathbf{d_s^k}\|$ are bounded away from zero. As the cost function at each iteration is finite, we have,

$$\sum_{k=1}^{\infty} \beta_4 \|\nabla J(\mathbf{x}^k, \mathbf{u}^k)\| < \infty. \tag{3.41}$$

It follows that,

$$\lim_{k\to\infty} \nabla J(\mathbf{x}^k, \mathbf{u}^k) = 0. \tag{3.42}$$

Thus $(\mathbf{x}^k, \mathbf{u}^k)$ from the iLQR algorithm converges to the stationary point of the optimization problem in Eq. (3.25), which completes the proof. □

Theorem 1 and the MOC result in paper [54] which is another work from our group lead to the following:

Theorem 2. *Under assumptions **A1**, **A2** and **A3**, with a cost function that is quadratic in control and a system whose dynamics are affine in control, the iLQR algorithm is guaranteed to converge to the global minimum of the optimization problem in Eq.* (3.25) *from any initial point.*

3.4.2 Data-Based Extension to ILQR

ILQR typically requires the availability of analytical system Jacobians, and thus, cannot be directly applied when such analytical gradient information is unavailable (much like Nonlinear Programming software whose efficiency depends on the availability of analytical gradients and Hessians). To make it an (analytical) model-free algorithm, it is sufficient to obtain estimates of the system Jacobians from simulation data, and a sample-efficient randomized way of doing so is described in the following.

Using Taylor expansion of the non-linear dynamics model in Section 2.2 in the deterministic setting about the nominal trajectory $(\bar{x}_t, \bar{u}_t)$ on both the positive and the negative sides, we obtain the following central difference equation: $f(\bar{x}_t + \delta x_t, \bar{u}_t + \delta u_t) - f(\bar{x}_t - \delta x_t, \bar{u}_t - \delta u_t)$

Algorithm 2: Decoupled Data-based Control (D2C) 2.0 Algorithm

$\Rightarrow$ ***Open-loop trajectory optimization***

Initialization: Set initial state x_0, initial guess $u^0_{0:T-1}$, line search parameter $\alpha = 1$, regularization $\mu = 10^{-6}$, iteration counter $k = 0$, convergence coefficient $\epsilon = 0.001$, line search threshold $\sigma_1 = 0.3$.

while $cost_k/cost_{k-1} < 1 - \epsilon$ **do**

  `/* backward pass */`

  $\{k_{0:T-1}, K_{0:T-1}\} = backward\_pass(u^k_{0:T-1}, x^k_{0:T-1})$.

  `/* forward pass */`

  $z = 0$.

  while $z < \sigma_1$ **do**

    Reduce α,

    $u^{k+1}_{0:T-1}, x^{k+1}_{0:T-1}, cost_k, \Delta cost(\alpha) =$
    $forward\_pass(u^k_{0:T-1}, x^k_{0:T-1}, \{k_{0:T-1}, K_{0:T-1}\}, \alpha)$,

    $z = (cost_k - cost_{k-1})/\Delta cost(\alpha)$.

  end while

  $k = k + 1$.

end while

$\bar{u}_{0:T-1} = u^{k+1}_{0:T-1}$.

$\bar{x}_{0:T-1} = x^{k+1}_{0:T-1}$.

$\Rightarrow$ ***Closed-loop feedback design***

1. $A_t, B_t = LLSCD(\bar{u}_{0:T-1}, \bar{x}_{0:T-1})$.
2. Calculate feedback gain $K_{0:T-1}$ from Eq. (3.19).
3. Final closed-loop control policy: $u^*_t = \bar{u}_t + K_t \delta x_t$,

where δx_t is the state deviation from the nominal trajectory.

Algorithm 3: Forward Pass

Input: Previous iteration nominal trajectory $u^k_{0:T-1}, x^k_{0:T-1}$, iLQR gains $\{k_{0:T-1}, K_{0:T-1}\}$, line search parameter α.

Start from $t = 0$, $cost = 0$, $\Delta cost(\alpha) = 0$, $\bar{x}_0 = x_0$.

while $t < T$ **do**

$$
\begin{aligned}
u^{k+1}_t &= u^k_t + \alpha k_t + K_t(x^{k+1}_t - x^k_t), \\
x^{k+1}_{t+1} &= simulate\_forward\_step(x^{k+1}_t, u^{k+1}_t), \\
cost &= cost + incremental\_cost(x^{k+1}_t, u^{k+1}_t),
\end{aligned}
$$

  $t = t + 1$.

end while

$cost = cost + terminal\_cost(x^{k+1}_T)$,

$\Delta cost(\alpha) = -\alpha \sum_{t=0}^{T-1} k'_t Q_{u_t} - \frac{\alpha^2}{2} \sum_{t=0}^{T-1} k'_t Q_{u_t u_t} k_t$.

return $u^{k+1}_{0:T-1}, x^{k+1}_{0:T-1}, cost, \Delta cost(\alpha)$.

Algorithm 4: Backward Pass

Compute $J_{\mathbf{x_T}}$ and $J_{\mathbf{x_T x_T}}$ using boundary conditions.
$t = T - 1$. `/* start from the terminal step */`
while $t >= 0$ **do**
  `/* estimate the Jacobians using Linear Least Squares by Central Difference (LLS-CD) as shown in Section 3.4.2) */`
  $f_{x_t}, f_{u_t} = LLSCD(x_t, u_t)$.
  `/* obtain the partials of the Q function */`

$$
\begin{aligned}
Q_{x_t} &= c_{x_t} + f'_{x_t} J_{x_{t+1}}, \\
Q_{u_t} &= c_{u_t} + f'_{u_t} J_{x_{t+1}}, \\
Q_{x_t x_t} &= c_{x_t x_t} + f'_{x_t} J_{x_{t+1} x_{t+1}} f_{x_t}, \\
Q_{u_t x_t} &= c_{u_t x_t} + f'_{u_t} (J_{x_{t+1} x_{t+1}} + \mu I_{n_x \times n_x}) f_{x_t}, \\
Q_{u_t u_t} &= c_{u_t u_t} + f'_{u_t} (J_{x_{t+1} x_{t+1}} + \mu I_{n_x \times n_x}) f_{u_t}.
\end{aligned}
$$

  if $Q_{u_t u_t}$ is positive-definite **then**

$$
\begin{aligned}
k_t &= -Q_{u_t u_t}^{-1} Q_{u_t}, \\
K_t &= -Q_{u_t u_t}^{-1} Q_{u_t x_t}.
\end{aligned}
$$

    Decrease μ.
  else
    Increase μ.
    Redo backward pass for current timestep.
  end if
  `/* obtain the partials of the value function` J_t `*/`

$$
\begin{aligned}
J_{x_t} &= Q_{x_t} + K'_t Q_{u_t u_t} k_t + K'_t Q_{u_t} + Q'_{u_t x_t} k_t, \\
J_{x_t x_t} &= Q_{x_t x_t} + K'_t Q_{u_t u_t} K_t + K'_t Q_{u_t x_t} + Q'_{u_t x_t} K_t.
\end{aligned}
$$

  $t = t - 1$.
end while
return $\{k_{0:T-1}, K_{0:T-1}\}$.

$= 2\begin{bmatrix} f_{x_t} & f_{u_t} \end{bmatrix}\begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} + O(\|\delta x_t\|^3 + \|\delta u_t\|^3)$. Multiplying by $\begin{bmatrix} \delta x_t{}' & \delta u_t{}' \end{bmatrix}$ on both sides to the above equation and apply standard least square:

$$\begin{bmatrix} f_{x_t} & f_{u_t} \end{bmatrix} = M\delta Y_t'(\delta Y_t \delta Y_t')^{-1}$$

$$M = \begin{bmatrix} f(\bar{x}_t + \delta x_t^{(1)}, \bar{u}_t + \delta u_t^{(1)}) - f(\bar{x}_t - \delta x_t^{(1)}, \bar{u}_t - \delta u_t^{(1)}) \\ f(\bar{x}_t + \delta x_t^{(2)}, \bar{u}_t + \delta u_t^{(2)}) - f(\bar{x}_t - \delta x_t^{(2)}, \bar{u}_t - \delta u_t^{(2)}) \\ \vdots \\ f(\bar{x}_t + \delta x_t^{(n_s)}, \bar{u}_t + \delta u_t^{(n_s)}) - f(\bar{x}_t - \delta x_t^{(n_s)}, \bar{u}_t - \delta u_t^{(n_s)}) \end{bmatrix}$$

where 'n_s' be the number of samples for each of the random variables, δx_t and δu_t. Denote the random samples as $\delta X_t = \begin{bmatrix} \delta x_t^{(1)} & \delta x_t^{(2)} & \dots & \delta x_t^{(n_s)} \end{bmatrix}$, $\delta U_t = \begin{bmatrix} \delta u_t^{(1)} & \delta u_t^{(2)} & \dots & \delta u_t^{(n_s)} \end{bmatrix}$ and $\delta Y_t = \begin{bmatrix} \delta X_t & \delta U_t \end{bmatrix}$. We are free to choose the distribution of δx_t and δu_t. We assume both are i.i.d. Gaussian distributed random variables with zero mean and a standard deviation of σ. This ensures that $\delta Y_t \delta Y_t'$ is invertible.

Let us consider the terms in the matrix $\delta Y_t \delta Y_t' = \begin{bmatrix} \delta X_t \delta X_t{}' & \delta X_t \delta U_t{}' \\ \delta U_t \delta X_t{}' & \delta U_t \delta U_t{}' \end{bmatrix}$, $\delta X_t \delta X_t{}' = \sum_{i=1}^{n_s} \delta x_t{}^{(i)} \delta x_t{}^{(i)\prime}$. Similarly, $\delta U_t \delta U_t{}' = \sum_{i=1}^{n_s} \delta u_t{}^{(i)} \delta u_t{}^{(i)\prime}$, $\delta U_t \delta X_t{}' = \sum_{i=1}^{n_s} \delta u_t{}^{(i)} \delta x_t{}^{(i)\prime}$ and $\delta X_t \delta U_t{}' = \sum_{i=1}^{n_s} \delta x_t{}^{(i)} \delta u_t{}^{(i)\prime}$. From the definition of sample variance, for a large enough n_s, we can write the above matrix as:

$$\begin{aligned} \delta Y_t \delta Y_t' &= \begin{bmatrix} \sum_{i=1}^{n_s} \delta x_t{}^{(i)} \delta x_t{}^{(i)\prime} & \sum_{i=1}^{n_s} \delta x_t{}^{(i)} \delta u_t{}^{(i)\prime} \\ \sum_{i=1}^{n_s} \delta u_t{}^{(i)} \delta x_t{}^{(i)\prime} & \sum_{i=1}^{n_s} \delta u_t{}^{(i)} \delta u_t{}^{(i)\prime} \end{bmatrix} \\ &\approx \begin{bmatrix} \sigma^2(n_s - 1)\mathrm{I}_{n_x} & 0_{n_x \times n_u} \\ 0_{n_u \times n_x} & \sigma^2(n_s - 1)\mathrm{I}_{n_u} \end{bmatrix} \\ &= \sigma^2(n_s - 1)\mathrm{I}_{(n_x+n_u)\times(n_x+n_u)} \end{aligned} \tag{3.43}$$

Typically for $n_s \sim O(n_x + n_u)$, the above approximation holds good. The reason is as follows. Note that the above least square procedure converges when the matrix $\delta Y_t \delta Y_t'$ converges

to the identity matrix. This is equivalent to estimation of the covariance of the random vector $\delta Y_t = [\delta X_t \; \delta U_t]$ where δX_t, and δU_t are Gaussian i.i.d. samples. Thus, it follows that the number of samples is $O(n_x + n_u)$, given $n_x + n_u$ is large enough (see [37]).

This has important ramifications since the overwhelming bulk of the computations in the D2C iLQR implementation consists of the estimation of these system dynamics. Moreover, these calculations are highly parallelizable.

Henceforth, we will refer to this method as “Linear Least Squares by Central Difference (LLS-CD)".

3.4.3 Data-Based Closed-Loop Design

The iLQR design in the open-loop part also furnishes a linear feedback law, however, this is not the linear feedback corresponding to the optimal feedback law. To accomplish this, we need to use the feedback gain equations (3.18). This can be done in a data-based fashion analogous to the LLS-CD procedure above as shown in the Appendix Section A.3, but in practice, the converged iLQR feedback gain offers very comparable performance to the optimal feedback gain. The entire algorithm is summarized together in Algorithm 2. The “forward pass" and “backward pass" algorithms are summarized in Algorithms 3 and 4 respectively.

3.5 Empirical Results

In the following, we report the results of training and performance of D2C 2.0 on typical benchmarking examples and its comparison to DDPG [9], twin-delayed DDPG (TD3)[52] and soft actor-critic (SAC)[51]. DDPG was regarded as an efficient global deep RL method. TD3 and SAC introduced further improvements and outperformed DDPG on many benchmark examples. These three are the current state-of-the-art RL methods and thus are chosen to compare with D2C 2.0. The physical models of the system are deployed in the simulation platform ‘MuJoCo-2.0’ [38] as a surrogate to their analytical models. The models are imported from the OpenAI gym [58] and Deepmind’s control suite [39]. In addition, to further illustrate scalability, we test the D2C 2.0 algorithm on a material microstructure control problem (state dimension of 400) which is

governed by a PDE called the Allen-Cahn equation. All simulations are done on a machine with the following specifications: AMD Ryzen 3700X 8-Core CPU@3.59 GHz, with 16 GB RAM, with no multi-threading. The D2C is implemented in Matlab and the RL methods are implemented in Python. The DDPG code is adapted from the Keras-rl [40] implementation. SAC and TD3 are adapted based on the Stable Baselines3 implementation [59].

We test the algorithms on four fronts that allow us to test the speed and reliability of the learning, as well as the performance of the learned controllers:

1. *Training efficiency*, where we study the time required for training,
2. *Reliability of the training*, studied using the variance of the resulting answers,
3. *Robustness of the learned controllers* to differing levels of noise, and hence, a test of the "global nature" of the synthesized feedback law, and
4. *Learning in stochastic systems*, where we show the effects of a persistent process noise process on learning and performance of the techniques.

3.5.1 Model Description

1) MuJoCo Models

Here we provide details of the MuJoCo models used in our simulations.

Inverted pendulum: A swing-up task of this 2D system from its downright initial position is considered.

Cart-pole: The state of a 4D under-actuated cart-pole comprises the angle of the pole, the cart's horizontal position and their rates. Within a given horizon, the task is to swing up the pole and balance it in the middle of the rail by applying a horizontal force on the cart.

3-link Swimmer: The 3-link swimmer model has 5 degrees of freedom and together with their rates, the system is described by 10 state variables. The task is to solve the planning and control problem from a given initial state to the goal position located at the center of the ball. Controls can only be applied in the form of torques to two joints. Hence, it is under-actuated by 3 DOF.

6-link Swimmer: The task with a 6-link swimmer model is similar to that defined in the 3-link case. However, with 6 links, it has 8 degrees of freedom and hence, 16 state variables, controlled by 5 joint motors.

Fish: The fish model moves in 3D space, the torso is a rigid body with 6 DOF. The system is described by 26 dimensions of states and 6 control channels. Controls are applied in the form of torques to the joints that connect the fins and tails with the torso. The rotation of the torso is described using quaternions.

2) Material Model

The material microstructure is modeled as a 2D grid with periodic boundary, which satisfies the Allen-Cahn equation [60] at all times. The Allen-Cahn equation is a classical governing partial differential equation (PDE) for phase field models. It has a general form of

$$\frac{\partial \phi}{\partial t} = -M(\frac{\partial F}{\partial \phi} - \gamma \nabla^2 \phi) \tag{3.44}$$

where $\phi = \phi(x, t)$ is called the 'order parameter', which is a spatially varying, non-conserved quantity, and $\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}$, denotes the Laplacian of a function, and causes a 'diffusion' of the phase between neighbouring points. In control parlance, ϕ is the state of the system, and is infinite dimensional, i.e., a spatio-temporally varying function. It reflects the component proportion of each phase of the material system.

In this study, we adopt the following general form for the energy density function *F*:

$$F(\phi; T, h) = \phi^4 + T\phi^2 + h\phi \tag{3.45}$$

Herein, we take both *T*, the temperature, and *h*, an external force field such as an electric field, to be available to control the behavior of the material. In other words, the material dynamics process is controlled from a given initial state to the desired final state by providing the values of *T* and *h*. The control variables T and h are, in general, spatially (over the material domain) as well as temporally varying.

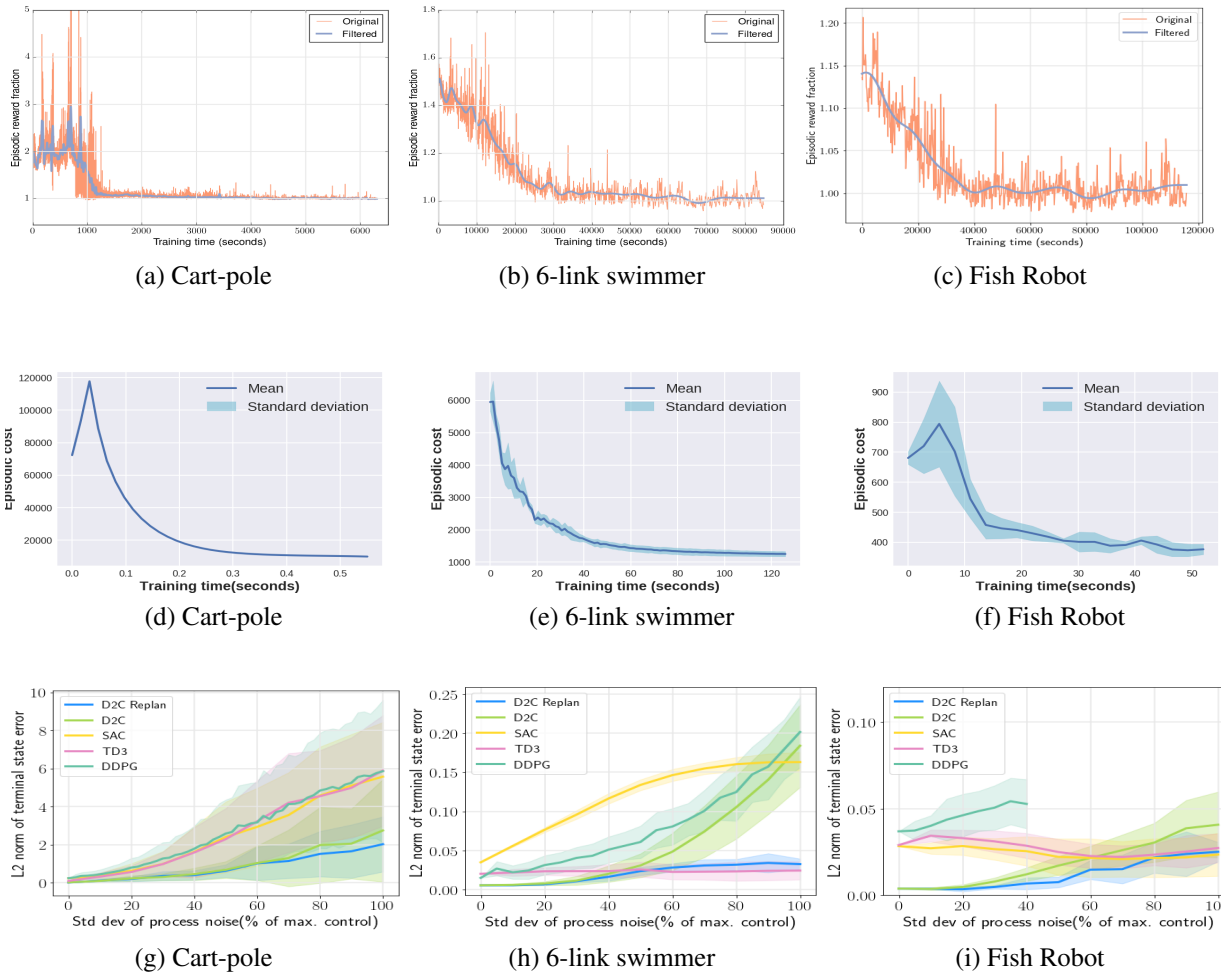
The material model simulated consists of a 2-dimensional grid of dimension 20x20, i.e., 400 states. The order parameter at each of the grid points can be varied in the range $[-1, 1]$. The model is solved numerically using an explicit, second order, central-difference-based finite difference (FD) scheme. The number of control variables is a fourth of the observation space, i.e., 100 each for both control inputs T and h. Physically, it means that we can vary the T and h values over 2×2 patches of the domain. Thus, the model has 400 state variables and 200 control channels. The control task is to influence the material dynamics to converge to a banded phase distribution as shown in Fig. 3.1(d).

The initial and the desired final state of the model are shown in Fig. 3.1(c, d). The model starts at an initial configuration of all states at $\phi = -1$, i.e., the entire material is in one phase. The final state should converge to alternating bands of $\phi = 0$ (red) and $\phi = 1$ (blue), with each band containing 2 columns of grid points. Thus, this is a very high-dimensional example with a 400-dimensional state and 200 control variables.

Remark 6. *We note that the methods have access to the same simulation models and no hidden advantage or extra information is provided to either algorithm. Note also that the models, other than the cart-pole and the pendulum examples, lack an analytical description (analytically intractable) and are computational models. Thus, data-based methods such as DDPG, SAC, TD3 and D2C can be construed as control synthesis techniques for such analytically intractable models.*

3.5.2 Training Efficiency

We measure training efficiency by comparing the times taken for the episodic cost (or reward) to converge during training. Figure. 3.3 shows the training process with DDPG and D2C 2.0 on the systems considered. Table 3.1 delineates the times taken for training for all four methods respectively. The total time comparison in Table 3.1 shows that D2C 2.0 learns the optimal policy orders of magnitude faster than the RL methods. The primary reason for this disparity is the feedback parametrization of the two methods: the RL deep neural nets are complex parametrizations that are difficult to search over, when compared to the highly compact open-loop + linear feedback



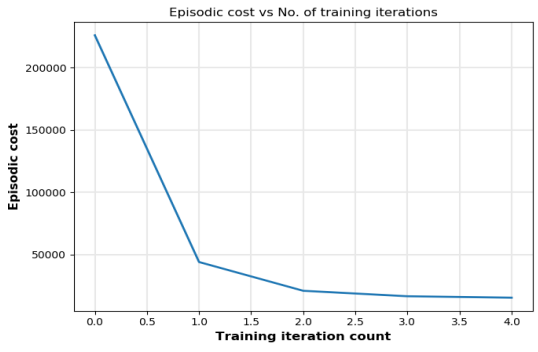
(a) Cart-pole (b) 6-link swimmer (c) Fish Robot

(d) Cart-pole (e) 6-link swimmer (f) Fish Robot

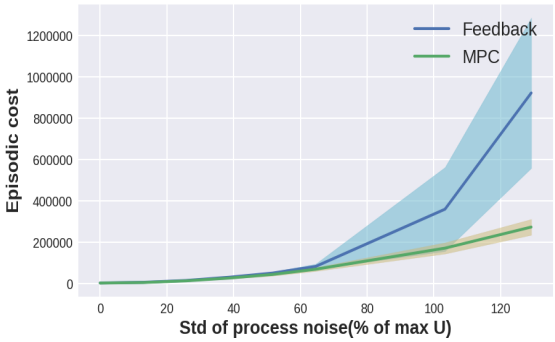
(g) Cart-pole (h) 6-link swimmer (i) Fish Robot

Figure 3.3: Comparison of training and testing results between D2C 2.0 and RL methods. Top row: Convergence of episodic cost in DDPG. Middle row: Convergence of episodic cost in D2C 2.0. Bottom row: L2-norm of terminal state error during testing in D2C 2.0 vs RL methods. The solid line in the plots indicates the mean and the shade indicates the standard deviation of the corresponding metric.

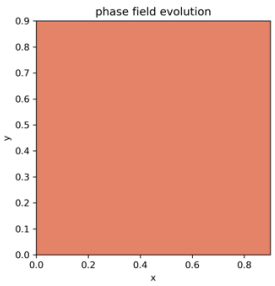
parametrization of D2C 2.0, i.e. the number of parameters optimized during D2C 2.0 training is the number of actuators times the number of timesteps while the RL parameter size equals the size of the neural networks, which is much larger. Due to the much larger network size, the computation done per rollout is much higher for the RL methods. From Fig. 3.4(a), on the material microstructure problem (a 400-dimensional state and 100-dimensional control), we observe that D2C 2.0 converges very quickly, even for a very high dimensional system ($d = 400$), whereas DDPG fails to converge



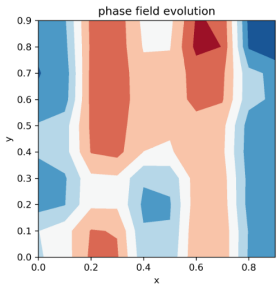
(a) Material Microstructure



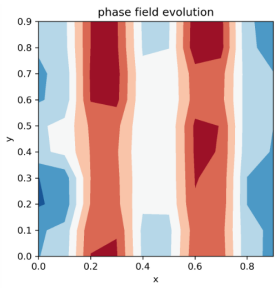
(b) Closed-loop Performance



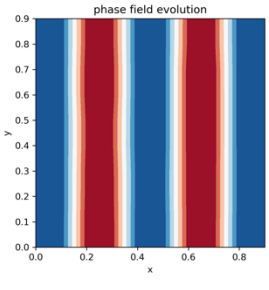
(c) Initial



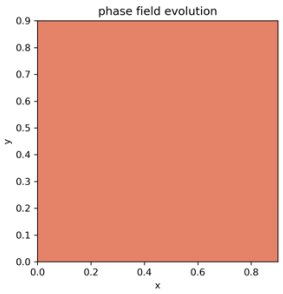
(d) t=0.50s



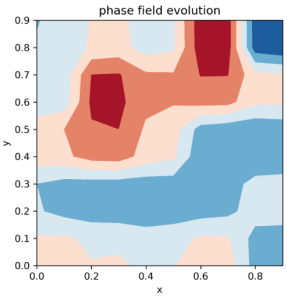
(e) t=1.00s



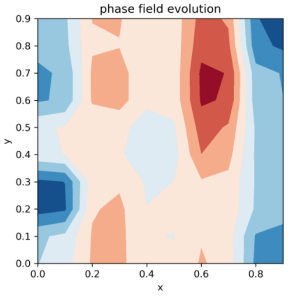
(f) t=1.25s



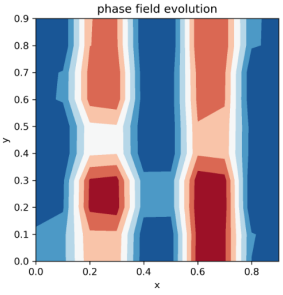
(g) Initial



(h) t=0.50s

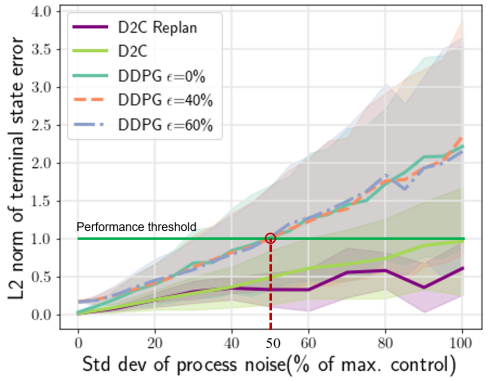


(i) t=1.00s

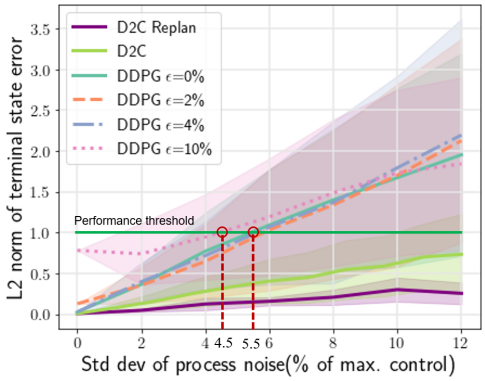


(j) t=1.25s

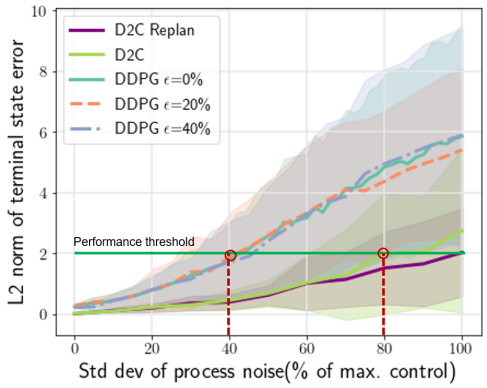
Figure 3.4: D2C 2.0 results in the material microstructure.
Fig. (a): Episodic cost vs. training iteration number in D2C 2.0 for the material microstructure. Fig. (b): Closed-loop performance comparison between D2C 2.0 with LQR feedback and D2C 2.0 with replanning. Figs. (c)-(f): Closed-loop trajectories showing the temporal evolution of the spatial microstructure from the initial configuration on the left to the desired configuration on the extreme right with no input noise. Figs. (g)-(j): Closed-loop trajectories with Gaussian input noise at 50% U_{max} standard deviation.



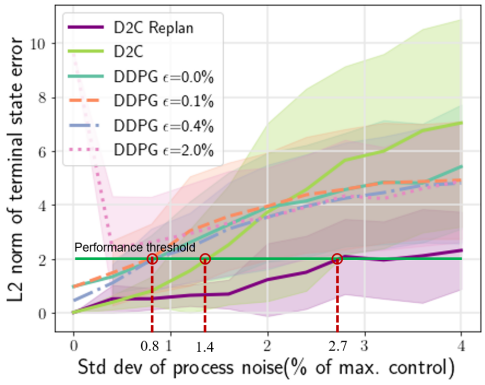
(a) Inverted Pendulum - varied process noise in control



(b) Inverted Pendulum - varied process noise in state and control



(c) Cart-Pole - varied process noise in control



(d) Cart-Pole - varied process noise in state and control

Figure 3.5: Comparison between D2C 2.0, D2C 2.0 with replanning and DDPG on the L2-norm of terminal state error tested under varying process noise.

to the correct goal state. We also note the benefit of ILQR here: due to its quadratic convergence properties, the convergence is very fast, when allied with the randomized LLS-CD procedure for Jacobian estimation.

3.5.3 Closed-loop Performance

It might be expected that since the RL methods search over a nonlinear neural network parametrization, it provides a global feedback law while D2C, by design, only provides a local feedback, and thus, the performance of the RL methods might be better globally. To test this

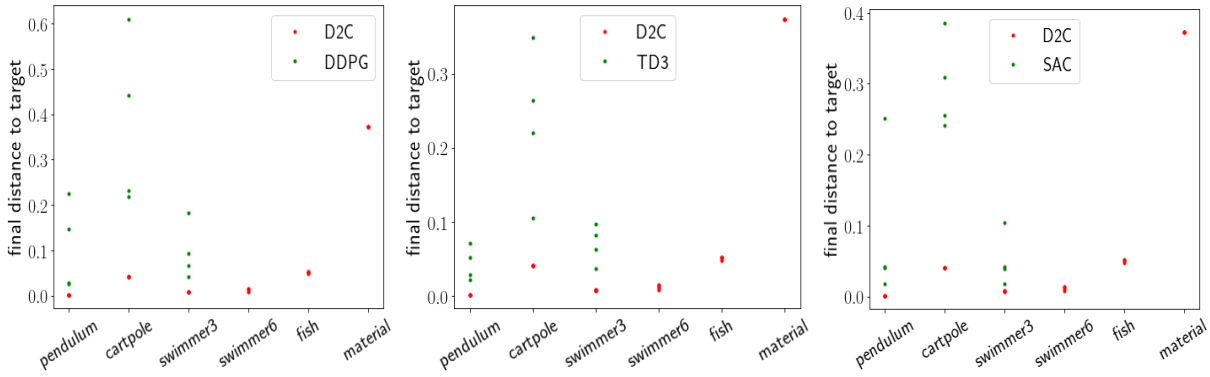


Figure 3.6: Training variance comparison D2C 2.0 vs. RL methods.

hypothesis, we apply noise to the system via the ϵ parameter, and find the average performance of all the methods at each noise level. This has the effect of perturbing the state from its nominal path, and thus, can be used to test the efficacy of the controllers far from a nominal path, i.e., their global behavior. However, it can be seen from Fig. 3.3 bottom row that the performance of D2C 2.0 is better than the RL methods at all noise levels, in the sense that the terminal error of the D2C 2.0 controller is lower than that of the RL controller. This, in turn implies that albeit the RL methods are theoretically global in nature, in practice, it is reliable only locally, and moreover, their performance is inferior to the local D2C approach. We also report the effect of replanning on the D2C scheme, and it can be seen from these plots that the performance of the replanned controller is far better than both D2C 2.0 and the RL methods, thereby regaining globally optimal behavior. We also note that the performance of D2C 2.0 is similar to the high-dimensional material microstructure control problem while DDPG fails to converge in this problem (Fig. 3.4).

Replanning with D2C 2.0: Under large noise levels, the local feedback policy found by D2C 2.0 may not give a good closed-loop performance, and thus we introduce a replanning procedure that re-solves the open-loop design from the current state of the system and wrap another local feedback policy along the new optimal trajectory. During the replanning, we take the current nominal policy as the initial guess. With this warm start, the time and iterations taken in each replanning step are less than solving the open-loop optimization with a 'zero' initial guess in D2C 2.0. Under

$100\%\ U_{max}$ noise, the fish needs 25 seconds and 13 iterations, and the 6-link swimmer needs 90 seconds and 51 iterations, on average, for each replanning. As the cart-pole fails under high noise levels, it is tested with $40\%\ U_{max}$ noise and needs 12 iterations, and 0.5 seconds, on average. Thus, by replanning, the closed-loop performance can be improved with an affordable training time increase. Finally, we note that the estimation of the feedback gain takes a very small fraction of the training time when compared to the open-loop, even though it is a much bigger parameter: this is a by-product of the decoupling result.

3.5.4 Reliability of Training

For any algorithm that has a training step, it is important that the training result be stable and reproducible, and thus reliable. However, reproducibility is a major challenge that the field of RL is yet to overcome, a manifestation of the extremely high variance of RL training results. Thus, we test the training variance of D2C 2.0 by conducting multiple training sessions with the same hyperparameters but different random seeds. The middle row of Fig. 3.3 shows the mean and the standard deviation of the episodic cost data during 16 repeated D2C 2.0 training runs each. For the cart-pole model, the results of all the training experiments are almost the same. Even for more complex models like the 6-link swimmer and the fish, the training is stable and the variance is small. Further investigation into the training results shows that given the set of hyperparameters, D2C 2.0 always results in the same policy (with a very small variance) unlike the results of the RL methods which have high variance even after convergence, which was reported in [61]. We show this in Fig. 3.6, where the final distance to target of the nominal trajectories (i.e., nominal control sequence of D2C 2.0 and the RL methods) generated from 4 different instances of converged training of all the four methods with identical training hyper-parameters. It can be noted that the D2C 2.0 results overlap with each other with a very small variance while the RL methods' results have a much wider spread. Although TD3 and SAC outperformed DDPG, their training variance is still much higher than that of D2C 2.0. The high variance of the training results makes it questionable whether the RL methods converge to an optimal solution or whether the seeming convergence is simply the result of the shrinking exploration noise as training progresses. On the other hand, D2C 2.0

can guarantee the same solution from converged training. Thus, the advantage of a local approach like D2C in training stability and reproducibility makes it far more reliable for solving data-based optimal control problems when compared to global approaches like DDPG, TD3 and SAC.

Table 3.1: Comparison of the training outcomes of D2C 2.0 with DDPG, TD3 and SAC.

System	**Training time (in sec.)**				**Training variance**			
	D2C	**DDPG**	**TD3**	**SAC**	**D2C**	**DDPG**	**TD3**	**SAC**
Inverted Pendulum	0.33	2261.2	937.6	1462.6	6.7×10^{-5}	0.08	0.02	0.09
Cart pole	0.55	6306.7	4304.5	7407.4	0.0004	0.16	0.09	0.06
3-link Swimmer	186.2	38833.6	48038.0	64035.0	0.0007	0.05	0.02	0.03
6-link Swimmer	127.2	88160.0	47508.4	57372.7	0.0023	*	*	*
Fish	54.8	124367.6	46238.7	95769.1	0.0016	*	*	*

* No data because the training time taken is too long.

3.5.5 Learning on Stochastic Systems

A noteworthy facet of the D2C 2.0 design is that it is agnostic to the uncertainty, encapsulated by ϵ, and the near-optimality stems from the local optimality (identical nominal control and linear feedback gain) of the deterministic feedback law when applied to the stochastic system. One may then question the fact that the design is not for the true stochastic system, and thus, one may expect RL techniques to perform better since they are applicable to the stochastic system. However, in practice, most RL algorithms only consider the deterministic system, in the sense that the only noise in the training simulations is the exploration noise in the control, and not from a persistent process noise. We now show the effect of adding a persistent process noise with a small to moderate value of ϵ to the training of DDPG, in the control as well as the state.

We trained the DDPG policy on the pendulum and cart-pole examples. To simulate the stochastic environment, Gaussian i.i.d. random noise is added to all the input channels as process noise. The noise level ϵ is the noise standard deviation divided by the maximum control value of the open-loop optimal control sequence. Figure. 3.5 shows the closed-loop performance of DDPG policies trained

and tested under different levels of process noise. The closed-loop performance is measured by the mean and variance of the L2-norm of terminal state error. In the first column, process noise is only added to input channels during training and testing while it is added to both state and input in the second column. The performance threshold is the L2-norm of terminal state error, chosen such that the system at that terminal state is close enough to the target state and can be stabilized with an LQR controller designed around the target state. For the pendulum, it is chosen such that the angle and angular velocity are smaller than $40°$ and $40°/s$ respectively. For the cart-pole, the angle and angular velocity of the bar need to be smaller than $40°$ and $40°/s$ respectively, the cart position and velocity need to be smaller than $0.8m$ and $1.5m/s$. The thresholds of testing process noise for the policies to keep a decent performance are marked in the plots. When the testing process noise is larger than the threshold, the tested policy can not get close enough to the target state. The problem is greatly exacerbated in the presence of state noise as seen from Fig. 3.5(b)(d) that results in bad performance in the different examples for even small levels of noise. Hence, although theoretically, RL algorithms such as DDPG can train on the stochastic system, in practice, the process noise level ϵ must be limited to a small value for training convergence and/or good policies. Further, the DDPG policy trained at a specific noise level does not perform better than a DDPG policy trained at a different noise level, when tested at that specific noise level. Also, in all the cases shown in these plots, the D2C 2.0 policy outperforms all the DDPG policies within the performance threshold. Further, the D2C 2.0 policy with replanning outperforms all the DDPG policies over all tested noise levels. Thus, this begs the question as to whether we should train on the stochastic system rather than appeal to the decoupling result that the deterministic policy is locally identical to the optimal stochastic policy, and thus train on the deterministic system. A theoretical exploration of this topic, in particular, the variance inherent in RL, is the subject of another paper from our group [62].

3.6 Conclusions

In this section, we have presented an improved data-based control method, D2C 2.0, for synthesizing the feedback control law for analytically intractable nonlinear optimal control problems. The D2C 2.0 policy is not global, i.e., it does not claim to be valid over the entire state space,

however, seemingly global deep RL methods do not offer better performance as can be seen from our experiments. Further, owing to the fast and reliable open-loop solver, D2C 2.0 could offer a real-time solution even for high-dimensional problems when allied with high-performance computing. In such cases, one could replan whenever necessary, and this replanning procedure will make the D2C 2.0 approach global in scope, as we have shown in this section, albeit not in real-time.

There might be a sentiment that the comparison with the RL methods is unfair due to the wide chasm in the training times, however, the primary point is to show theoretically, as well as empirically, that the local parametrization and search procedure advocated via D2C 2.0, is a highly efficient and reliable (almost zero variance) alternative that is still superior in terms of closed-loop performance when compared to typical global RL algorithms like DDPG, TD3 and SAC. Thus, for data-based optimal control problems that need efficient training, reliable near-optimal solutions, and robust closed-loop performance, such local RL techniques, coupled with replanning, should be the preferred method over typical global RL methods.

Finally, we would like to note that methods such as DDP/ iLQR are termed “trajectory optimization" techniques and are thought to be distinct from RL algorithms. We have shown in this section that these methods are indeed (local) RL, or data-based control methods, when suitably modified as presented in this section, and thus, should not be thought of as distinct from RL.

4. EXTENSION TO PARTIALLY OBSERVED CASES*

4.1 Introduction

The optimal control of a nonlinear dynamical system is computationally intractable for complex high order systems due to the “curse of dimensionality" associated with solving dynamic programming [4]. The problem becomes more challenging when the model of the system is unknown and even more formidable when only some of the states are available for measurement, i.e., under partial state observation. In fact, most problems tend to be partially observed.

In this chapter, we propose a data-based approach for learning to optimally control complex partially observed nonlinear dynamical systems. The primary idea is to convert the partially observed optimal control problem to a “fully-observed" problem described using the information state which is comprised of the past several measurements and controls. The proposed approach then generalizes the iLQR algorithm [13] to partially observed problems by iteratively generating LTV state-space models, represented in the information state, to obtain the optimized nominal information space trajectory. These LTV models are constructed with Autoregressive–Moving-Average (ARMA) models [63] along a nominal trajectory using input-output perturbation data (rollouts of the system). We term this generalized iLQR as partially observed data-based iLQR (POD-iLQR) and use it as the new open-loop optimal trajectory design method. The ARMA modeling also allows for the development of a specific linear quadratic Gaussian (LQG) controller [64] as the new closed-loop feedback design.

In comparison with other research in the classic iLQR literature, previous work such as [14] employed finite differencing in computing the Jacobians using complete state information as opposed to this work which utilizes information state-based LTV systems that only uses output information, and thus, this work suitably generalizes the iLQR method to partially observed systems in a systematic fashion. Researchers have recently developed an RL-based iLQR which develops a

*Part of this chapter is reprinted with permission from "Data-based control of partially-observed robotic systems" by R. Wang, R. Goyal, S. Chakravorty and R. E. Skelton, 2021. IEEE International Conference on Robotics and Automation (ICRA), pp. 8104-8110, Copyright 2021 by IEEE.

neural network model from the measurement data and searches for the optimal policy by iteratively refining the neural network [65]. Another research couples input-driven sequential variational autoencoder with iLQR-based learning to extend deep learning approaches [66]. A hybrid control and learning approach, called Reinforced iLQR, was also developed recently for learning robotic locomotion [67]. These approaches usually require a very large set of well-constructed data to train neural networks and thus can be hard to solve if the data is not well sampled or if it does not capture the complete nonlinear dynamics.

A belief space variant of iterative LQG (iLQG) has also been proposed which finds a locally optimal solution under motion and sensing uncertainty [68]. The belief state comprises sufficient statistics from the history and the initial belief state that no additional information on past observations and past actions is required [69]. Therefore, the belief state allows converting the modeling of the system from partially observed Markov decision processes (POMDPs) to Markov decision processes (MDPs) [70]. Our work is related to this belief space literature, in the sense, that we generate the optimal trajectory using a generalized information state version of the iLQR method, where “information state" is defined using q past most likely (zero noise) observations (of dimension n_z), with $q \times n_z \leq n_x$, which allows us to solve a large class of complex and high dimensional partially observed control problems in a highly efficient fashion since the complexity is $O(n_x)$ rather than $O(n_x + n_x^2)$ for typical Gaussian belief space planning problems. However, unlike the belief space literature, our theoretical development does not consider process and sensing uncertainties, albeit we do consider both types of uncertainties in our experiments by augmenting our generated information state feedback controller with a Kalman filter used to estimate the information state under noisy observations.

The ARMA models are written as the combination of the autoregressive (AR) model and moving-average (MA) model and are mostly used in the statistical analysis of time series [63]. The autoregressive part contains the sum of past values of the variable itself ($z_t = \sum_{i=1}^{p} \alpha_{t-i} z_{t-i}$) and the moving-average part contains the accumulation of past few noise error terms. In the context of this section, the moving-average part constitutes the input excitation added as i.i.d noise

($z_t = \sum_{i=1}^{q} \beta_{t-i} u_{t-i}$). The ARMA models have been extensively used for modeling and forecasting in the field of statistics, economics, and finance [71]. Researchers have also used the nonlinear version of these input-output parametric models, known as, nonlinear ARMA (NARMA) models for control of nonlinear systems for both deterministic and stochastic cases [72, 73]. Time-varying linear models were also proposed for input-output modeling of nonlinear functions by replacing it with a family of piece-wise linear functions [74]. Chemical industries have also been using this modeling idea for nonlinear model predictive control [75]. In this work, we use the ARMA modeling technique to fit the LTV model. The ARMA model parameters can be calculated from the input-output data, thus state perturbations are no longer needed.

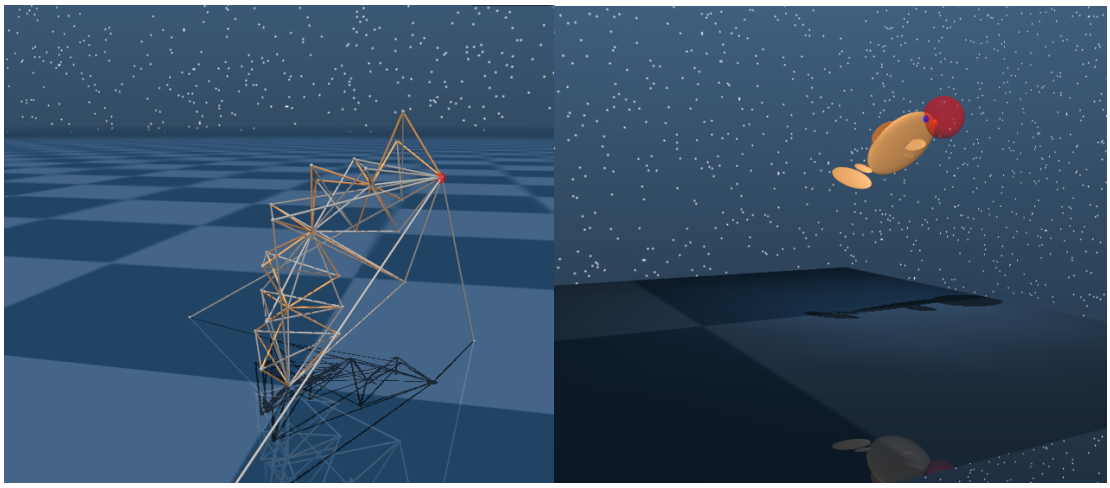


Figure 4.1: Complex robotic model in their final states.
Note the high dimensional, complex, and partially observed nature of the problems (L) Tensegrity Robotic Arm (150 states, 24 outputs), (R) Fish (27 states, 11 outputs).

The approach proposed here, POD2C, is a generalization of D2C 2.0 to partially observed systems using the information state-based ARMA model. In this chapter, we provide the theoretical justification of the information state approach to partially observed problems and detail the framework to solve the problem using POD-iLQR and information state-based LQG along with the

complete algorithm. The extended algorithm is tested on highly nonlinear and high-dimensional dynamical systems, including challenging cases of hard-to-model soft contact constraints and dynamic fluid interactions, with imperfect measurements of only a few of the states. Most importantly, it is observed that the number of measurements required for solving the optimization problem is greatly reduced especially for high-dimensional systems. This LTV-ARMA system identification approach based on the information state is critical since existing LTV identification techniques tend to be very brittle and do not scale to the complex nonlinear problems considered in this paper [76].

This chapter is organized as follows. Section 4.2 provides the optimal control problem formulation for partially observed nonlinear systems. Section 4.3 presents the results to transform a nonlinear partially observed problem into a fully observed problem in the information state form and provides the conditions for the deterministic global optimal solution. Section 4.5 develops the framework to solve the open-loop design problem using POD-iLQR where the LTV-ARMA model is developed in a data-based fashion. Section 4.5 gives the details of the POD2C closed-loop feedback control design along with the complete algorithm. Section 4.6 shows the empirical results on partially observed complex robotic systems tested in the presence of process and sensor noise.

4.2 Problem Formulation

Let us start by writing the non-linear dynamics in discrete time state space form as follows:

$$x_{t+1} = f(x_t, u_t), \tag{4.1}$$

where x_t is the state, u_t is the control input of the system. Let us assume the observation model to be of form: $z_t = h(x_t)$. Let us now define a finite horizon objective function as:

$$J(x_0) = \sum_{t=0}^{T-1} c(z_t, u_t) + c_T(z_T), \tag{4.2}$$

where $c(z_t, u_t)$ denotes a running incremental cost and $c_T(z_T)$ denotes a terminal cost function. The goal of this work is to find an observation-based feedback policy, i.e., a policy that only has access

to the observations z_t, such that the cost above is minimized. The above formulation shall be termed the partially observed optimal control problem in the rest of this section.

4.3 Transformation to an Information State Problem

Let $f^n(x_{t-q}; u_{t-q}, u_{t-q+1}, \cdots, u_{t-q+n-1})$ denote the map from the state at time $t-q$, x_{t-q}, to the state x_{t-q+n} at time $t-q+n$. Given the initial state and inputs from time $t-q$ to time $t-1$, i.e, $\{x_{t-q}; u_{t-q}, u_{t-q+1}, \cdots, u_{t-1}\}$, we can write the following expressions for the observations $\{z_{t-q}, z_{t-q+1}, \cdots, z_t\}$ as:

$$\begin{aligned} z_{t-q} &= h(x_{t-q}), \\ z_{t-q+1} &= h(f^1(x_{t-q}; u_{t-q})), \\ &\vdots \\ z_t &= h(f^q(x_{t-q}; u_{t-q}, u_{t-q+1}, \cdots, u_{t-1})). \end{aligned} \tag{4.3}$$

In the partially observed problem, we start by finding the underlying state x_{t-q}, which is the solution to the above set of nonlinear equations.

Assumption 1. *Observability: We assume that there exists a finite $\bar{q}$, such that for all $q \geq \bar{q}$, Eq.* (4.4) *has a unique solution for x_{t-q}, regardless of (Z_t^q, U_t^q).*

Due to the implicit function theorem and the observability assumption, we can write x_{t-q} as some unique function $\bar{f}(.,.)$ of past measurements and inputs as:

$$x_{t-q} = \bar{f}(Z_t^q, U_t^q). \tag{4.4}$$

Next, let us write the state at current time x_t as: $x_t = f^q(x_{t-q}; u_{t-q}, u_{t-q+1}, \cdots, u_{t-1})$, which can be written again by substituting for x_{t-q} from Eq. (4.4) in some unique functional form, based on observability assumption and implicit function theorem, as:

$$x_t = \Psi(Z_t^q, U_t^q). \tag{4.5}$$

Let us now finally define the "*information state*" $\mathcal{Z}_t^q$ at time t as:

$$\mathcal{Z}_t^q = \left[z_t^T, z_{t-1}^T, \cdots, z_{t-q}^T, u_{t-1}^T, \cdots, u_{t-q}^T \right]^T, \tag{4.6}$$

which allows us to write:

$$x_t = \Psi(\mathcal{Z}_t^q). \tag{4.7}$$

Further, due to the implicit function theorem, if the dynamics and the observation functions f and h are $\mathcal{C}^k$, so is the map Ψ.

The above development can be summarized as follows:

Lemma 5. *Given Assumption 1, there exists a unique function $\Psi(\cdot)$, such that the state at time t, $x_t = \Psi(\mathcal{Z}_t^q)$. In particular, if $f(\cdot)$ and $h(\cdot)$ are $\mathcal{C}^k$, so is the function Ψ.*

The above result shows that the state at time t is some nonlinear map of the observation and the control inputs at the previous "q" time steps.

4.3.1 The Global Optimal Solution for the Partially Observed Problem

Next, we extend our result on the globally optimal solution for the fully observed case to the partially observed case via the information state construct developed above. Let us consider the special case of a system that is affine in control dynamics: $\dot{x} = f(x) + g(x)u$. We write the system dynamics with a forward Euler approximation for a small discretization time:

$$x_{t+1} = x_t + f(x_t)\Delta t + g(x_t)u_t\Delta t, \tag{4.8}$$

and the observation model as:

$$z_t = h(x_t) = h(x_{t-1} + dx_t), \tag{4.9}$$

where $dx_t = f(x_{t-1})\Delta t + g(x_{t-1})u_{t-1}\Delta t$. Given that the time discretization is sufficiently small, the observation model can be written using a first order Taylor expansion as:

$$z_t = h(x_{t-1} + dx_t) = h(x_{t-1}) + \underbrace{\frac{\partial h}{\partial x}\Big|_{x_{t-1}}}_{H(x_{t-1})} dx_t, \tag{4.10}$$

$$z_t = \underbrace{h(x_{t-1}) + H(x_{t-1})f(x_{t-1})\Delta t}_{F(x_{t-1})} + \underbrace{H(x_{t-1})g(x_{t-1})\Delta t}_{G(x_{t-1})} u_{t-1}. \tag{4.11}$$

Thus, we can write the observation z_t as some function of x_{t-1} and u_{t-1} as:

$$z_t = F(x_{t-1}) + G(x_{t-1})u_{t-1}. \tag{4.12}$$

Further substituting for x_{t-1} from Eq. (4.7) as $x_{t-1} = \Psi(\mathcal{Z}^q_{t-1})$, we can write the z_t in terms of the information state as:

$$z_t = F(\Psi(\mathcal{Z}^q_{t-1})) + G(\Psi(\mathcal{Z}^q_{t-1}))u_{t-1}, \tag{4.13}$$

and finally, the entire equation can trivially be written in terms of the information state as:

$$\mathcal{Z}^q_t = \tilde{F}(\mathcal{Z}^q_{t-1}) + \tilde{G}(\mathcal{Z}^q_{t-1})u_{t-1}. \tag{4.14}$$

This shows that the system dynamics in information state is affine in controls similar to the underlying system (Eq. (4.8)), and thus, the original partially observed control problem:

$$\begin{aligned} \bar{u}_t &= \arg\min_{u_t} \sum_{t=0}^{T-1} c(z_t, u_t) + c_T(z_T), \\ s.t.\ \ x_{t+1} &= x_t + f(x_t)\Delta t + g(x_t)u_t\Delta t, \\ z_t &= h(x_t), \end{aligned} \tag{4.15}$$

where the instantaneous cost function is quadratic in control: $c(z_t, u_t) = (l(z_t) + \frac{1}{2}u_t^T R u_t)\Delta t$,

can equivalently be posed as the following "fully observed" optimal control problem in terms of information state:

$$\bar{u}_t = \arg\min_{u_t} \sum_{t=0}^{T-1} c(z_t, u_t) + c_T(z_T), \tag{4.16}$$

$$s.t.\ \ \mathcal{Z}_{t+1}^q = \mathcal{Z}_t^q + \mathcal{F}(\mathcal{Z}_t^q)\Delta t + \mathcal{G}(\mathcal{Z}_t^q)u_t\Delta t. \tag{4.17}$$

Theorem 3. *Let the cost functions $l(\cdot)$, $c_T(\cdot)$, the drift $f(\cdot)$ and the input influence function $g(\cdot)$ be $\mathcal{C}^2$, i.e., twice continuously differentiable. The unique global minimum of the open-loop problem (Eq.* (4.16)*) starting at some initial information state $\mathcal{Z}_0$ satisfies:*

$$\bar{u}_t = -R^{-1}\bar{\mathcal{G}}_t^T G_t,\ G_t = \bar{L}_t^{\mathcal{Z}} + \mathbb{A}_t^T G_{t+1}, \tag{4.18}$$

$$K_t = -(R_t + \mathbb{B}_t^T P_{t+1}\mathbb{B}_t)^{-1}[\sum_{i=1}^{n} \bar{\mathcal{G}}_{t,i}^{\mathcal{Z},T} G_{t+1}^i + \mathbb{B}_t^T P_{t+1}\mathbb{A}_t], \tag{4.19}$$

$$P_t = \mathbb{A}_t^T P_{t+1}\mathbb{A}_t + \bar{L}_t^{\mathcal{Z}\mathcal{Z}} + \sum_{i=1}^{n}[\bar{\mathcal{F}}_{t,i}^{\mathcal{Z}\mathcal{Z}} + \sum_{j=1}^{p} \bar{\Gamma}_{t,i}^{j,\mathcal{Z}\mathcal{Z}}\bar{u}_t^j]G_{t+1}^i - K_t^T(R_t + \mathbb{B}_t^T P_{t+1}\mathbb{B}_t)K_t, \tag{4.20}$$

where $\mathbb{A}_t = I + \left(\bar{\mathcal{F}}_t^{\mathcal{Z}} + \sum_{j=1}^{p} \bar{\Gamma}_t^{j,\mathcal{Z}}\bar{u}_t^j\right)\Delta t,\ \mathbb{B}_t = \bar{\mathcal{G}}\Delta t,\ \bar{\mathcal{G}}_t = \mathcal{G}(\bar{\mathcal{Z}}_t)$, and $\{\bar{\mathcal{Z}}_t\}$ represents the optimal nominal trajectory, $\bar{L}_t^{\mathcal{Z}} = \nabla_{\mathcal{Z}} l|_{\bar{\mathcal{Z}}_t}$, $G_T = \nabla_{\mathcal{Z}} c_T|_{\bar{\mathcal{Z}}_T}$, $\bar{u}_t = [\bar{u}_t^1 \cdots \bar{u}_t^p]^T$, the control influence matrix: $\mathcal{G} = \left[\Gamma^1(\mathcal{Z}) \cdots \Gamma^p(\mathcal{Z})\right]$, and Γ^j and $\bar{u}_t^j$ represents the control influence vector and optimal control vector corresponding to the j^{th} input. Finally, $\bar{\mathcal{F}}_t^{\mathcal{Z}} = \nabla_{\mathcal{Z}}\mathcal{F}|_{\bar{\mathcal{Z}}_t}$ and $\bar{\Gamma}_t^{j,\mathcal{Z}} = \nabla_{\mathcal{Z}}\Gamma^j|_{\bar{\mathcal{Z}}_t}$ gives the Jacobians of the system dynamics and $\bar{\mathcal{F}}_{t,i}^{\mathcal{Z}\mathcal{Z}}$ and $\bar{\Gamma}_{t,i}^{j,\mathcal{Z}\mathcal{Z}}$ gives the Hessians of the system dynamics along the nominal trajectory, the initial information state is defined as $\mathcal{Z}_0 = [z_0', z_0', \cdots, z_0', 0, 0, \cdots, 0]'$.

Proof. We make use of the result that under the regularity conditions above, for a fully observed system, satisfying the Minimum Principle is sufficient to obtain the global optimal trajectory for the problem formulated in Eq. (4.16), the proof for which is given in Section 3.4. Given that the information state construct turns the partially observed problem into a fully observed one (Lemma 5), and that Eq. (4.18) is simply the minimum principle for the information state with Eq. (4.19)

giving the optimal linear feedback gain K_t, the result follows directly. □

Remark 7. *The above result shows that we do not need global knowledge of the dynamics $\mathcal{F}(\cdot)$ and $\mathcal{G}(\cdot)$, rather it is sufficient if we iteratively find the locally linearized model (the estimates of Jacobians) around nominal Information State trajectories in order to converge to the optimum.*

4.4 Open-Loop Trajectory Design using POD-ILQR

This subsection details the algorithm for open-loop trajectory design using POD-iLQR. The advantage of iLQR is that the equations involved are given explicitly in terms of the LTV dynamics. These LTV dynamics can be calculated for the information state in a data-based fashion, using our proposed LTV-ARMA identification algorithm.

4.4.1 Information State for the LTV System

Let the nominal state be denoted by $\bar{x}_t$ and the deviations from the nominal state as δx_t, which can be modeled as the following LTV system linearized around the nominal trajectory as:

$$\delta x_t = A_{t-1}\delta x_{t-1} + B_{t-1}\delta u_{t-1},\ \delta z_t = C_t \delta x_t. \tag{4.21}$$

Let us also form a notion of nominal information state evolution and deviations as $\bar{\mathcal{Z}}_t$ and $\delta\mathcal{Z}_t = (\delta z_t, \delta z_{t-1}, \cdots, \delta z_{t-q+1}, \delta u_{t-1}, \cdots, \delta u_{t-q+1})$, where $\delta z_t = z_t - \bar{z}_t$ and $\delta u_t = u_t - \bar{u}_t$ are the deviation from the nominal observation at time t. We now provide the information state formulation for the linearized version of the nonlinear system.

Lemma 6. *The state and observation at current time δx_t and δz_t has a unique map to past q observations δZ_t^q, and control inputs δU_t^q for the linear time-varying model (Eq.* (4.21)*), given the observability matrix $O^q = \begin{bmatrix} A_{t-q}^T ... A_{t-2}^T C_{t-1}^T, & \cdots, & A_{t-q}^T C_{t-q+1}^T, & C_{t-q}^T \end{bmatrix}^T$ has full column rank.*

Proof. Let us start by writing the output equation for past q time-steps as Eq. (4.26), which in simplified form can be written as:

$$O^q \delta x_{t-q} = \delta Z_t^q - G^q \delta U_t^q. \tag{4.22}$$

A unique solution for δx_{t-q} exist if the matrix O^q is full column rank matrix and then the solution can be written as:

$$\delta x_{t-q} = O^{q^+}(\delta Z_t^q - G^q \delta U_t^q). \tag{4.23}$$

Now, the system output at time t can be written as:

$$\delta z_t = C_t A_{t-1} ... A_{t-q} \delta x_{t-q} + \begin{bmatrix} C_t B_{t-1} & C_t A_{t-1} B_{t-2} & \cdots & C_t A_{t-1} ... B_{t-q} \end{bmatrix} \delta U_t^q, \tag{4.24}$$

where the unique solution for δx_{t-q} is substituted to get:

$$\delta z_t = C_t A_{t-1} ... A_{t-q} O^{q^+}(\delta Z_t^q - G^q \delta U_t^q) + \begin{bmatrix} C_t B_{t-1} & C_t A_{t-1} B_{t-2} & \cdots & C_t A_{t-1} ... B_{t-q} \end{bmatrix} \delta U_t^q, \tag{4.25}$$

which shows a unique map to the information state $\delta \mathcal{Z}_t^q$. □

The above result allows us to write the unique mapping of the past measurements and control inputs to the present state, which can be captured using the ARMA models. The following result gives the exact solution for the ARMA parameters.

$$\underbrace{\begin{bmatrix} \delta z_{t-1} \\ \delta z_{t-2} \\ \vdots \\ \delta z_{t-q+1} \\ \delta z_{t-q} \end{bmatrix}}_{\delta Z_t^q} = \underbrace{\begin{bmatrix} C_{t-1} A_{t-2} ... A_{t-q} \\ C_{t-2} A_{t-3} ... A_{t-q} \\ \vdots \\ C_{t-q+1} A_{t-q} \\ C_{t-q} \end{bmatrix}}_{O^q} \delta x_{t-q} + \underbrace{\begin{bmatrix} 0 & C_{t-1} B_{t-2} & \cdots & C_{t-1} A_{t-2} ... B_{t-q+1} & C_{t-1} A_{t-2} ... B_{t-q} \\ 0 & 0 & C_{t-2} B_{t-3} & \cdots & C_{t-2} A_{t-3} ... B_{t-q} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & C_{t-q+1} B_{t-q} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{G^q} \underbrace{\begin{bmatrix} \delta u_{t-1} \\ \delta u_{t-2} \\ \vdots \\ \delta u_{t-q+1} \\ \delta u_{t-q} \end{bmatrix}}_{\delta U_t^q}, \tag{4.26}$$

Proposition 7. *An ARMA model of the order q given by: $\delta z_t = \alpha_{t-1} \delta z_{t-1} + \cdots + \alpha_{t-q} \delta z_{t-q} + \beta_{t-1} \delta u_{t-1} + \cdots + \beta_{t-q} \delta u_{t-q}$, exactly fits the LTV system given in Eq.* (4.21) *if matrix $O^q =$*

$\left[A_{t-q}^T ... A_{t-2}^T C_{t-1}^T, \quad \cdots, \quad A_{t-q}^T C_{t-q+1}^T, \quad C_{t-q}^T\right]^T$ *is full column rank. The exact parameters that matches the LTV system can then be written as:*

$$[\alpha_{t-1} \mid \alpha_{t-2} \mid \cdots \mid \alpha_{t-q}] = C_t A_{t-1} ... A_{t-q} O^{q^+},$$
$$[\beta_{t-1} \mid \beta_{t-2} \mid \cdots \mid \beta_{t-q}] = -C_t A_{t-1} ... A_{t-q} O^{q^+} G^q + \begin{bmatrix} C_t B_{t-1} & C_t A_{t-1} B_{t-2} & \cdots & C_t A_{t-1} ... B_{t-q} \end{bmatrix}.$$

Notice that if there exists a number $\bar{q}$ for which the matrix $O^{\bar{q}}$ is full column rank, then there always exists an exact fit for the ARMA model with sufficiently large enough $q > \bar{q}$. This allows us to write linearized models at each step along the nominal trajectory in terms of ARMA parameters.

Corollary 1. *For the case of mechanical systems with all the position/DOFs as the output feedback, the minimum value for q would be $q = 2$, which would allow for the exact fit for the ARMA model to be with only 2 past observations and inputs.*

4.4.2 Linear Time-Varying System Identification using ARMA Model

Here we use the standard least square method to estimate the linear ARMA parameters from input-output experiment data.

First we start from the perturbed linear system about the nominal trajectory and estimate the system parameters α_{t-i} and β_{t-i} for $i = 1, \cdots, q$ from $\delta z_t^{(j)} = \alpha_{t-1}\delta z_{t-1}^{(j)} + \cdots + \alpha_{t-q}\delta z_{t-q}^{(j)} + \beta_{t-1}\delta u_{t-1}^{(j)} + \cdots + \beta_{t-q}\delta u_{t-q}^{(j)}$, where $\delta z_t^{(j)}$ is the observed output and $\delta u_t^{(j)} \sim \mathcal{N}(0, \sigma I)$ is the control input perturbation we feed to the system at step t for the j^{th} rollout. All the perturbations are zero-mean, i.i.d, Gaussian noise. The covariance σ is a $o(u)$ small value selected by the user. After N rollouts, we can write the linear mapping between input-output perturbation as:

$$\mathbb{Z} = [\alpha_{t-1} \mid \alpha_{t-2} \mid \cdots \mid \alpha_{t-q} \mid \beta_{t-1} \mid \beta_{t-2} \mid \cdots \mid \beta_{t-q}]\mathbb{X}, \tag{4.27}$$

and write out the components:

$$\mathbb{Z} = \begin{bmatrix} \delta z_t^{(1)} & \delta z_t^{(2)} & \cdots & \delta z_t^{(N)} \end{bmatrix}, \mathbb{X} = \begin{bmatrix} \delta z_{t-1}^{(1)} & \delta z_{t-1}^{(2)} & \cdots & \delta z_{t-1}^{(N)} \\ \delta z_{t-2}^{(1)} & \delta z_{t-2}^{(2)} & \cdots & \delta z_{t-2}^{(N)} \\ \vdots & \vdots & \ddots & \vdots \\ \delta z_{t-q}^{(1)} & \delta z_{t-q}^{(2)} & \cdots & \delta z_{t-q}^{(N)} \\ \delta u_{t-1}^{(1)} & \delta u_{t-1}^{(2)} & \cdots & \delta u_{t-1}^{(N)} \\ \delta u_{t-2}^{(1)} & \delta u_{t-2}^{(2)} & \cdots & \delta u_{t-2}^{(N)} \\ \vdots & \vdots & \ddots & \vdots \\ \delta u_{t-q}^{(1)} & \delta u_{t-q}^{(2)} & \cdots & \delta u_{t-q}^{(N)} \end{bmatrix}. \tag{4.28}$$

Finally, using the standard least square method, the ARMA parameters of the linearized system are estimated as:

$$[\alpha_{t-1} \mid \cdots \mid \alpha_{t-q} \mid \beta_{t-1} \mid \cdots \mid \beta_{t-q}] = \mathbb{Z}\mathbb{X}^{\mathsf{T}}(\mathbb{X}\mathbb{X}^{\mathsf{T}})^{-1}. \tag{4.29}$$

4.4.3 LTV System Dynamics in Information State

After identifying the system parameters $\alpha_{t-1}, \cdots, \alpha_{t-q}$ and $\beta_{t-1}, \cdots, \beta_{t-q}$ for $t = \{0 \cdots T\}$, now, we write the perturbation LTV system in the information state as given in Eq. (4.31), which is written again with observation model as:

$$\delta\mathcal{Z}_t = \mathcal{A}_{t-1}\delta\mathcal{Z}_{t-1} + \mathcal{B}_{t-1}\delta u_{t-1},\ \delta z_t = \mathcal{C}_t\delta\mathcal{Z}_t, \tag{4.30}$$

with $\mathcal{C}_t = [I_{n_z}\ 0]$, where n_z is the number of measured outputs. Now, we can use the identified $\mathcal{A}_{t-1}$, $\mathcal{B}_{t-1}$ to find the optimal nominal trajectory using iLQR, and design the optimal linear feedback gain, i.e., $\delta u_t = K_t\delta\mathcal{Z}_t$ (Eq. (4.19) in Theorem 3) for the information state, where note that the current control input depends on the past observations as well as the control inputs. The feedback design procedure is described in Section 4.5.1

$$\underbrace{\begin{bmatrix} \delta z_t \\ \delta z_{t-1} \\ \delta z_{t-2} \\ \vdots \\ \delta z_{t-q+1} \\ \hline \delta u_{t-1} \\ \delta u_{t-2} \\ \delta u_{t-3} \\ \vdots \\ \delta u_{t-q+1} \end{bmatrix}}_{\delta \mathcal{Z}_t} = \underbrace{\left[\begin{array}{ccccc|ccccc} \alpha_{t-1} & \alpha_{t-2} & \cdots & \alpha_{t-q+1} & \alpha_{t-q} & \beta_{t-2} & \beta_{t-3} & \cdots & \beta_{t-q+1} & \beta_{t-q} \\ 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots & \vdots & & \ddots & \vdots & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{array}\right]}_{\mathcal{A}_{t-1}} \underbrace{\begin{bmatrix} \delta z_{t-1} \\ \delta z_{t-2} \\ \vdots \\ \delta z_{t-q+1} \\ \delta z_{t-q} \\ \hline \delta u_{t-2} \\ \delta u_{t-3} \\ \vdots \\ \delta u_{t-q+1} \\ \delta u_{t-q} \end{bmatrix}}_{\delta \mathcal{Z}_{t-1}} + \underbrace{\begin{bmatrix} \beta_{t-1} \\ 0 \\ \vdots \\ 0 \\ 0 \\ \hline 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathcal{B}_{t-1}} \delta u_{t-1} \tag{4.31}$$

4.4.4 Matching Markov Parameters for the Identified LTV System in Information State

Most system identification approaches make use of the generalized Markov parameters to match the impulse response characteristics of the plant i.e. they represent the input-output relation of the true plant in the time domain. This subsection shows that the identified LTV system in the information state matches the impulse response of the underlying nonlinear system linearized around some nominal trajectory.

Theorem 4. *The generalized Markov parameters of the LTV system (Eq.* (4.30)*) represented in Information State, $\mathcal{H}_{t,\tau}$ exactly matches the generalized Markov parameters of the underlying LTV system (Eq.* (4.21)*), $H_{t,\tau}$, i.e.,*

$$\mathcal{H}_{t,\tau} = H_{t,\tau}, \; \forall \, t, \tau. \tag{4.32}$$

where $\mathcal{H}_{t,\tau}$ is defined as:

$$\mathcal{H}_{t,\tau} = \begin{cases} \mathcal{C}_t \mathcal{A}_{t-1} \mathcal{A}_{t-2} \cdots \mathcal{A}_{\tau+1} \mathcal{B}_\tau; & \forall \, \tau < t-1 \\ \mathcal{C}_t \mathcal{B}_\tau; & \tau = t-1 \\ 0; & \forall \, \tau > t \end{cases}, \tag{4.33}$$

where $H_{t,\tau}$ is defined as:

$$H_{t,\tau} = \begin{cases} C_t A_{t-1} A_{t-2} \cdots A_{\tau+1} B_\tau; & \forall\, \tau < t-1 \\ C_t B_\tau; & \tau = t-1 \\ 0; & \forall\, \tau > t \end{cases} . \tag{4.34}$$

Proof. We here provide a brief proof for the linear time-invariant (LTI) system due to space constraints, which can be similarly extended to the LTV system. Let us write the Markov parameter $\mathcal{H}_{t,t-1}$ from Information State system as:

$$\mathcal{CB} = [I_{n_z}\ 0]\mathcal{B}_{t-1} = \beta_{t-1} = CB, \tag{4.35}$$

where the β_{t-1} (ARMA parameter) in the last equation was calculated from Proposition 7.

Similarly, we write $\mathcal{H}_{t,t-2}$ as:

$$\begin{aligned} \mathcal{CAB} &= \begin{bmatrix} \alpha_{t-1} & \cdots & \alpha_{t-q} & \beta_{t-2} & \cdots & \beta_{t-q} \end{bmatrix} \mathcal{B} \\ &= \alpha_{t-1}\beta_{t-1} + \beta_{t-2} \\ &= CAB - CA^q O^{q^+} \begin{bmatrix} CB \\ 0 \\ \vdots \\ 0 \end{bmatrix} + CA^q O^{q^+} \begin{bmatrix} CB \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= CAB, \end{aligned} \tag{4.36}$$

where $X(:,j)$ represents the j^{th} column of matrix X. Next, we write $\mathcal{H}_{t,t-3}$ as:

$$\mathcal{C}\mathcal{A}^2\mathcal{B} = \begin{bmatrix} \alpha_{t-1} & \cdots & \alpha_{t-q} & \beta_{t-2} & \cdots & \beta_{t-q} \end{bmatrix} \mathcal{A}\mathcal{B} \tag{4.37}$$

$$= \alpha_{t-1}CAB + \alpha_{t-2}CB + \beta_{t-3} \tag{4.38}$$

$$= CA^qO^{q^+} \begin{bmatrix} CAB \\ CB \\ 0 \\ \vdots \\ 0 \end{bmatrix} + CA^2B - CA^qO^{q^+} \begin{bmatrix} CAB \\ CB \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{4.39}$$

$$= CA^2B. \tag{4.40}$$

The rest of the Markov parameters can be similarly expanded to prove the result. □

4.5 Partially Observed Decoupled Data-based Control (POD2C) Algorithm

In the following, we detail the linear feedback design using the information state-LQG and present the complete POD2C algorithm that allows for the generation of the full closed-loop output feedback policy, i.e., the feedback as a function of the information state. The main observation is that the partially observed case might be treated similarly to the fully observed case by noting that the information state comprises past q observations and control inputs as detailed earlier.

4.5.1 Closed-Loop Control Design with Specific LQG Method

Given the estimated LTV perturbation model in the information state and the measurement model described as:

$$\delta\mathcal{Z}_t = \mathcal{A}_{t-1}\delta\mathcal{Z}_{t-1} + \mathcal{B}_{t-1}\delta u_{t-1} + \mathcal{D}_{t-1}w_{t-1}, \tag{4.41}$$

$$\delta\mathcal{Y}_t = \delta\mathcal{Z}_t + v_t, \tag{4.42}$$

Algorithm 5: Complete POD2C Algorithm

$\Rightarrow$ ***Open-loop trajectory design via POD-iLQR***

Initialization: Set initial state x_0, initial guess $u^0_{0:T-1}$, line search parameter $\alpha = 1$, regularization $\mu = 10^{-6}$, iteration counter $k = 0$, convergence coefficient $\epsilon = 0.001$, line search threshold $\sigma_1 = 0.3$.

while $(cost_k/cost_{k-1}) < 1 - \epsilon$ **do**

  `/* backward pass */`

  $\{k_{0:N-1}, K_{0:N-1}\} = backward\_pass(u^k_{0:T-1}, z^k_{0:T-1})$.

  `/* forward pass */`

  $\zeta = 0$.

  while $\zeta < \sigma_1$ **do**

    Reduce α,

    $u^{k+1}_{0:T-1}, z^{k+1}_{0:T-1}, cost_k, \Delta cost(\alpha) =$
    $forward\_pass(u^k_{0:T-1}, z^k_{0:T-1}, \{k_{0:T-1}, K_{0:T-1}\})$.

    $\zeta = (cost_k - cost_{k-1})/\Delta cost(\alpha)$.

  end while

  $k = k + 1$.

end while

$\bar{u}_{0:T-1} = u^{k+1}_{0:T-1}$.

$\bar{z}_{0:T-1} = z^{k+1}_{0:T-1}$.

$\Rightarrow$ ***Closed-loop feedback design***

1. $\mathcal{A}_t, \mathcal{B}_t = ARMA\_fitting(\bar{u}_{0:T-1}, \bar{z}_{0:T-1})$.

2. Calculate observer and feedback gains $L_{0:N-1}$ and $K_{0:N-1}$ using the specified LQG from Eq. (3.19).

3. Final POD2C control policy:

$u_t = \bar{u}^*_t - K_t \delta\hat{\mathcal{Z}}_t$,

where $\delta\hat{\mathcal{Z}}_t = \mathcal{A}_{t-1}\delta\hat{\mathcal{Z}}_{t-1} + \mathcal{B}_{t-1}\delta u_{t-1} + L_t(\delta\mathcal{Y}_t - \mathcal{A}_{t-1}\delta\hat{\mathcal{Z}}_{t-1} - \mathcal{B}_{t-1}\delta u_{t-1})$.

where w_t and v_t represent the process and measurement noise, we design a finite horizon, discrete time LQG along the trajectory for each timestep to minimize the cost function:

$$J = E\left[\delta\mathcal{Z}_T^T Q_T \delta\mathcal{Z}_T + \sum_{t=0}^{T-1}(\delta\mathcal{Z}_t^T Q_t \delta\mathcal{Z}_t + \delta u_t^T R_t \delta u_t)\right], \tag{4.43}$$

$$s.t. \quad \delta\mathcal{Z}_t = A_{t-1}\delta\mathcal{Z}_{t-1} + B_{t-1}\delta u_{t-1} + D_{t-1}w_{t-1},$$

$$\delta\mathcal{Y}_t = \delta\mathcal{Z}_t + v_t,$$

where δY is the noisy measurement. Notice that this is a specific version of LQG where state is estimated in the presence of process and measurement noise only as opposed to estimating the states from smaller number of noisy measurements, i.e. ($C = I$) and $L_t = P_t(P_t + V_t)^{-1}$, where P_t is solved in a forward propagation fashion from the Riccati equation:

$$P_{t+1} = A_t[P_t - P_t(P_t + V_t)^{-1}P_t]A_t^T + D_t W_t D_t^T, \tag{4.44}$$

with the initial condition $P_0 = E[\tilde{x}\tilde{x}^T]$.

The feedback gain for the above problem is calculated as:

$$K_{t-1} = (R + B_{t-1}^T S_t B_{t-1})^{-1}(B_{t-1}^T S_t A_{t-1}), \tag{4.45}$$

where S_t is solved in a back propagation fashion from the Riccati equation:

$$S_t = A_t^T S_{t+1}[I - B_t(R_t + B_t^T S_{t+1} B_t)^{-1} B_t^T S_{t+1}]A_t + Q_t, \tag{4.46}$$

with final condition as $S_T = Q_T$.

Then, the closed-loop linear feedback term is $u_t = -K_t\delta\hat{\mathcal{Z}}_t$, where $\delta\hat{\mathcal{Z}}_t = A_{t-1}\delta\hat{\mathcal{Z}}_{t-1} + B_{t-1}\delta u_{t-1} + L_t(\delta\mathcal{Y}_t - A_{t-1}\delta\hat{\mathcal{Z}}_{t-1} - B_{t-1}\delta u_{t-1})$.

Simplified LQR design: For the case of noiseless measurements, a simplified LQR design can be used for the closed-loop linear feedback term with $u_t = -K_t\delta\mathcal{Z}_t$ where K_t is calculated using

Eq. (4.45) and (4.46) only.

4.5.2 Complete POD2C Algorithm

Notice that assuming sufficient smoothness, any feedback law for the information state problem can be represented as $\pi_t(\mathcal{Z}_t) = \bar{u}_t + K_t\delta\mathcal{Z}_t + S_t(\delta\mathcal{Z}_t)$, where $\bar{u}_t$ is the nominal control sequence arising from $\pi_t(\cdot)$, K_t is the optimal linear feedback term (Eq. (4.19)) and $S_t(\cdot)$ are the higher order terms in the feedback law. The POD2C algorithm then proposes a 2 step procedure to approximate the solution for the original optimization problem. First, a noiseless open-loop optimization problem is solved to find an optimal control sequence, $\bar{u}_t^*$ via the POD-iLQR scheme. Then, an LQG controller is synthesized as described above. Finally, the control applied to the system is given by $u_t = \bar{u}_t^* - K_t\delta\hat{\mathcal{Z}}_t$, where K_t is the time-varying feedback gain.

Algorithm 6: Forward Pass

Input: Previous iteration nominal trajectory - $u_{0:T-1}^k, z_{0:T-1}^k$, iLQR gains - $\{k_{0:T-1}, K_{0:T-1}\}$, line search parameter α.
Start from $t = 0$, $cost = 0$, $\bar{x}_0 = x_0$, $\Delta cost(\alpha) = 0$.
while $t < T$ **do**

$$
\begin{aligned}
z_t^{k+1} &= C_t x_t^{k+1}, \\
u_t^{k+1} &= u_t^k + \alpha k_t + K_t(\mathcal{Z}_t^{k+1} - \mathcal{Z}_t^k), \\
x_{t+1}^{k+1} &= simulate\_forward\_step(x_t^{k+1}, u_t^{k+1}), \\
cost &= cost + incremental\_cost(\mathcal{Z}_t^{k+1}, u_t^{k+1})
\end{aligned}
$$

  $t = t + 1$.
end while
$cost = cost + terminal\_cost(\mathcal{Z}_T^{k+1})$,
$\Delta cost(\alpha) = -\alpha \sum_{t=0}^{T-1} k_t' Q_{u_t} - \frac{\alpha^2}{2} \sum_{t=0}^{T-1} k_t' Q_{u_t u_t} k_t$.
return $u_{0:T-1}^{k+1}, z_{0:T-1}^{k+1}, cost, \Delta cost(\alpha)$.

The complete POD2C algorithm to determine the optimal nominal trajectory in a data-based fashion along with closed-loop feedback law is summarized together in Algorithm 5, 6 and 7. As shown in [14], we use line search parameter α to find a good step size for a policy update.

Algorithm 7: Backward Pass

Compute $J_{\mathcal{Z}_\mathbf{T}^\mathbf{k}}$ and $J_{\mathcal{Z}_\mathbf{T}^\mathbf{k}\mathcal{Z}_\mathbf{T}^\mathbf{k}}$ using boundary conditions.
$t = T - 1$.
while $t >= 0$ **do**
`/* obtain the augmented Jacobians using ARMA model from Eq. (4.31) */`
$\mathcal{A}_t, \mathcal{B}_t = ARMA\_fitting(u^k_{0:T-1}, z^k_{0:T-1})$,
`/* obtain the partials of the Q function as follows */`

$$
\begin{aligned}
Q_{\mathcal{Z}_\mathbf{t}} &= c_{\mathcal{Z}_\mathbf{t}} + \mathcal{A}_t^T J'_{\mathcal{Z}_\mathbf{t+1}}, \\
Q_{\mathbf{u_t}} &= c_{\mathbf{u_t}} + \mathcal{B}_t^T J'_{\mathcal{Z}_\mathbf{t+1}}, \\
Q_{\mathcal{Z}_\mathbf{t}\mathcal{Z}_\mathbf{t}} &= c_{\mathcal{Z}_\mathbf{t}\mathcal{Z}_\mathbf{t}} + \mathcal{A}_t^T J'_{\mathcal{Z}_\mathbf{t+1}\mathcal{Z}_\mathbf{t+1}} \mathcal{A}_t, \\
Q_{\mathbf{u_t}\mathcal{Z}_\mathbf{t}} &= c_{\mathbf{u_t}\mathcal{Z}_\mathbf{t}} + \mathcal{B}_t^T (J'_{\mathcal{Z}_\mathbf{t+1}\mathcal{Z}_\mathbf{t+1}} + \mu I_{n_x \times n_x}) \mathcal{A}_t, \\
Q_{\mathbf{u_t}\mathbf{u_t}} &= c_{\mathbf{u_t}\mathbf{u_t}} + \mathcal{B}_t^T (J'_{\mathcal{Z}_\mathbf{t+1}\mathcal{Z}_\mathbf{t+1}} + \mu I_{n_x \times n_x}) \mathcal{B}_t.
\end{aligned}
$$

if $Q_{\mathbf{u_t}\mathbf{u_t}}$ is positive-definite **then**

$$
k_t = -Q^{-1}_{\mathbf{u_t}\mathbf{u_t}} Q_{\mathbf{u_t}}, K_t = -Q^{-1}_{\mathbf{u_t}\mathbf{u_t}} Q_{\mathbf{u_t}\mathcal{Z}_\mathbf{t}}.
$$

Decrease μ.
else
Increase μ.
Redo backward pass for current timestep.
end if
`/* obtain the partials of the value function` J_t `*/`

$$
\begin{aligned}
J_{\mathcal{Z}_\mathbf{t}} &= Q_{\mathcal{Z}_\mathbf{t}} + K_t^T Q_{\mathbf{u_t}\mathbf{u_t}} k_t + K_t^T Q_{\mathbf{u_t}} + Q^T_{\mathbf{u_t}\mathcal{Z}_\mathbf{t}} k_t, \\
J_{\mathcal{Z}_\mathbf{t}\mathcal{Z}_\mathbf{t}} &= Q_{\mathcal{Z}_\mathbf{t}\mathcal{Z}_\mathbf{t}} + K_t^T Q_{\mathbf{u_t}\mathbf{u_t}} K_t + K_t^T Q_{\mathbf{u_t}\mathcal{Z}_\mathbf{t}} + Q^T_{\mathbf{u_t}\mathcal{Z}_\mathbf{t}} K_t.
\end{aligned}
$$

$t = t - 1$.
end while
return $\{k_{0:N-1}, K_{0:N-1}\}$.

4.6 Empirical Results

For the following results, we use MuJoCo as a blackbox to provide the data to design the nominal trajectory and closed-loop feedback gain. First, we list the details of the MuJoCo models with their initial configuration shown in Figure 4.2.

Cart-Pole: The state of a four-dimension under-actuated cart-pole comprises the angle of the pole, the cart's horizontal position, and their rates. Within a given horizon, the task is to swing up the pole and balance it in the middle of the rail by applying a horizontal force on the cart.

15-link Swimmer: The 15-link swimmer model has 17 degrees of freedom and together with their rates, the system is described by 34 state variables. The task is to solve the planning and control problem from a given initial state to the goal position located at the center of the ball. Controls can only be applied in the form of torques to the 14 joints. Hence, it is under-actuated by 3 DOF.

Fish: The fish model moves in 3D space, the torso is a rigid body with 6 DOF. The system is described by 27 dimensions of states (including a set of quaternions) and 6 control channels. The task is to swim into the target ball. Controls are applied in the form of torques to the joints that connect the fins and tails with the torso. The rotation of the torso is described using quaternions.

T2D1 Robotic Arm: The T_2D_1 tensegrity model is a 3D robotic tensegrity arm. It consists of 33 bars and 46 strings. The bars are connected by ball joints. The task is to reach the top node into the target ball from the given initial configuration. Controls are applied in the form of tension in the strings, while the feedback is based on the coordinates of some of the nodes. The bars are shown as orange objects and the strings are shown as grey objects.

The final configuration of all the four models is given in Figs. 4.1 and 4.3 which are obtained at the end of the horizon with the POD2C algorithm.

The output number in Table 4.1 represents the minimum number of measurements needed to fit a good q^{th} order ARMA model. Note that a relatively smaller number of measurements are needed as the dimension of the system increases. The cart's horizontal position and angle of the pole are used as feedback for the classic cart-pole. The 15-link swimmer and fish are both

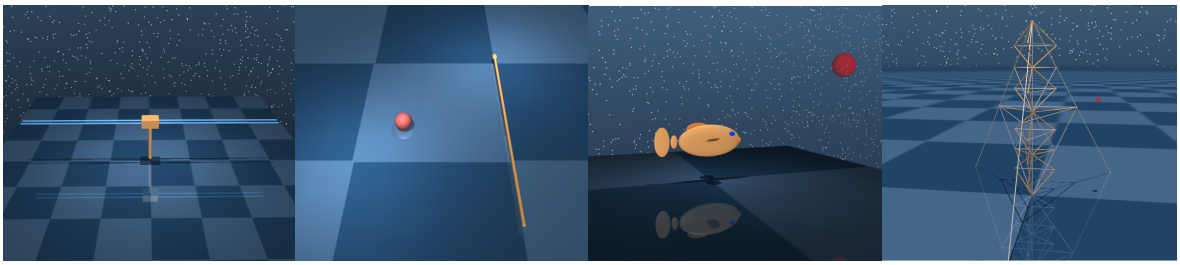


Figure 4.2: Models simulated in MuJoCo in their initial states.

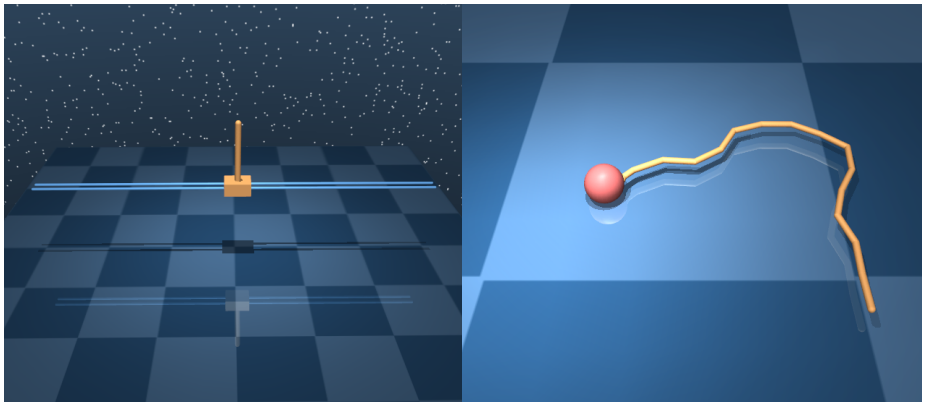
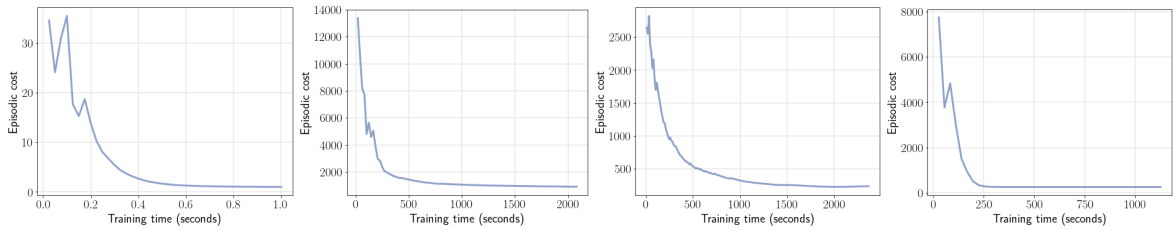


Figure 4.3: Models simulated in MuJoCo in their terminal states.

high-dimensional multi-body robots in a fluid environment. The 15-link swimmer requires only angular positions of every other joint and the fish needs only the angles of the fin and tail joints. The T_2D_1 arm is a high-dimensional soft-robotic arm where the positions of 8 evenly chosen nodes are needed out of the total 25 nodes. The rule of thumb for measurement selection is that we only measure the positions that contain the most information and avoid redundant information. Note that velocity feedback is not needed in the control design as the rates can be calculated from the past 2 observations of positions.

Open-loop training with POD-iLQR: As described in Section 4.5, we obtain the nominal trajectory from POD-iLQR training. As shown in Fig. 4.4, the cart-pole training converged quickly in 1.2 seconds consisting of 40 iterations. The 15-link swimmer and the fish take ≈ 2100 seconds

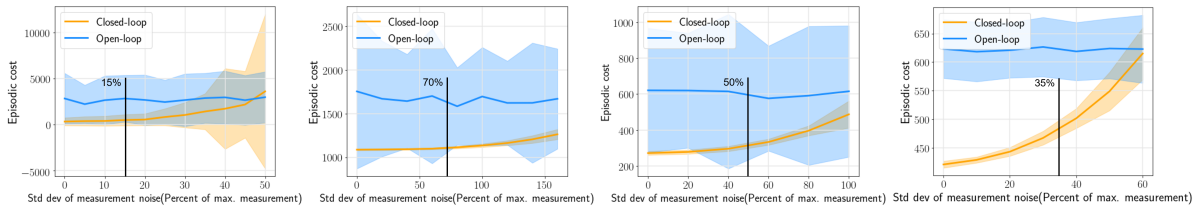
with 100 iterations and ≈ 2400 seconds with 200 iterations respectively. The 15-link swimmer and the fish take more iterations to converge as they have higher non-linearity due to the fluid-structure interaction in the swimming motion. However, the training is much more time-efficient compared with the first order gradient descent method, as well as the DDPG RL method as discussed in previous sections. The T2D1 arm system also takes smaller time and iterations ≈ 500 seconds with 25 iterations to converge despite the high dimensionality of the model and limited outputs. The convergence of all the tested cases is efficient even for systems with high non-linearity and high dimensionality. The time taken and iteration numbers during POD2C algorithm execution for the above cases are given in Table 4.1. The results are obtained using MatLab and MuJoCo on a Ryzen 3700 PC. The most time-consuming procedure is running simulations to collect data for fitting the ARMA model, which is in serial for now but can be in parallel as the rollouts are independent of each other and further improve the time efficiency and have the potential for real-time operation. The obtained nominal control trajectory is then used as the open-loop nominal control policy in the following steps.



(a) Cart-Pole (b) 15-link Swimmer (c) Fish (d) T2D1 Robotic Arm

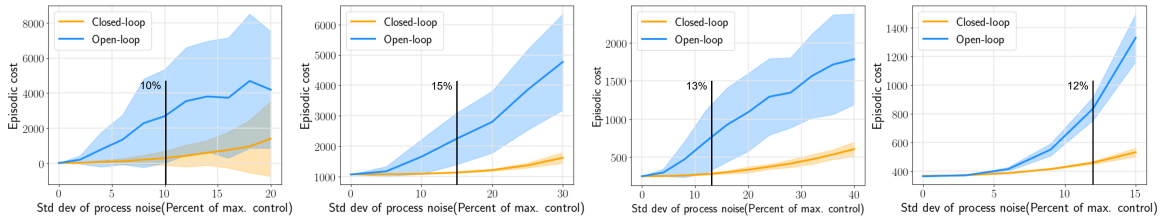
Figure 4.4: Convergence of episodic cost during training.

Robustness to measurement noise: Figure 4.5 shows the plots for the episodic cost with the variation in the measurement noise. The figure compares the open-loop and the closed-loop control policies under different measurement noise levels, while the process noise standard deviation is set to 10% of the maximal nominal control. The measurement noise level on the x-axis is the



(a) Cart-Pole (b) 15-link Swimmer (c) Fish (d) T2D1 Robotic Arm

Figure 4.5: Averaged episodic reward vs measurement noise level for fixed 10% process noise with LQG as closed-loop feedback.



(a) Cart-Pole (b) 15-link Swimmer (c) Fish (d) T2D1 Robotic Arm

Figure 4.6: Averaged episodic reward vs process noise level for fixed 10% measurement noise with LQG as closed-loop feedback.

percentage of the measurement noise standard deviation w.r.t. the maximal nominal measurement. Note that both the measurement and process noise is added as zero-mean Gaussian i.i.d. noise to all measurement and control channels at each step.

As the measurement noise does not influence the open-loop, the open-loop cost curves are almost flat with invariant variance. The closed-loop cost has a significantly smaller mean and variance than the open-loop cost, which proves the robustness to measurement noise of the closed-loop policy. Note that even for a large measurement noise, the closed-loop policy can successfully finish the task with smaller noise levels than what is indicated by the black threshold lines in the figure. Also, the variance of the open-loop policy, as well as the variance of the closed-loop policy at zero measurement noise, come from the fixed 10% process noise. The spikes in the open-loop curves are due to numerical error from the Monte-Carlo simulations.

Table 4.1: Simulation results of POD2C.

System	Steps per episode	Total Time (sec.)	Iteration Number	q	State Number	Output Number
Cart-pole	30	1.2	40	2	4	2
Swimmer	2400	2110.0	100	5	34	10
Fish	1200	2420.5	200	2	27	11
Robotic Arm	300	1155.8	40	3	150	24

Robustness to process noise: Figure 4.6 shows similar plots as shown in Fig. 4.5 except we vary the process noise level along the x-axis with fixed 10% measurement noise. Under the open-loop policy, the process noise drives the model off the nominal trajectory and results in high episodic cost, while the closed-loop feedback can help the system stay close to the nominal trajectory. This can be seen from the figure as the episodic cost mean and variance of the closed-loop policy is much smaller than the open-loop policy on the entire tested noise range, although both policies fail the task when the process noise becomes larger than what the black threshold line indicates. The threshold values of process and measurement noise in all the examples are found by constantly increasing the noise and finding the threshold noise level that will drive the system too far away from the nominal trajectory such that the final goal can not be achieved. The above analysis regarding the performance of control policy under noise proves that the LQG closed-loop feedback wrapped around the nominal trajectory makes the full closed-loop policy robust to measurement and process noise.

4.7 Conclusions

This section proposes an optimal motion planning/trajectory design algorithm for partially observed systems by transforming them into fully observed problems using the information state. The algorithm uses q^{th}-order ARMA models that are generated using the input-output data and are used to model LTV systems in the information state which allows designing the optimal nominal trajectory using iLQR with partial state observations. The constructed LTV system in the information state is shown to exactly match the Markov parameters of the underlying linear state-space model.

Finally, we presented the extended D2C 2.0 algorithm (POD2C) to control complex robotic systems by designing the closed-loop feedback law with partial state observations. Empirical results are shown for complex robotic systems, including challenging cases of hard-to-model soft contact constraints and dynamic fluid-structure interactions, under motion as well as sensing uncertainty. In our opinion, the POD2C approach is a highly efficient method for RL in partially observed problems, however, questions regarding optimality remain and shall be explored in future work. Also, there are still obstacles that keep D2C from real-world applications, such as how to learn the control policy under uncertainty. In the next chapter, we will look into the problem of learning to control under process and sensing noise.

5. LEARNING TO CONTROL UNDER UNCERTAINTY

5.1 Introduction

The optimal control of a nonlinear dynamical system is computationally intractable for complex high-dimensional systems due to the “curse of dimensionality" associated with dynamic programming [4]. The problem becomes more challenging when the analytical model of the system is unknown and even more formidable when only some of the states are available for measurement, i.e., under partial state observation. In fact, most real-world problems tend to be partially observed with the existence of uncertainty in the dynamics and the measurements. This has been recognized as one of the major gaps that keep controllers designed in simulation from applying successfully to real-world applications [77].

There has been a significant body of work in the field of learning to control unknown dynamical systems through RL, with great progress in overcoming the inability to create accurate dynamical models for complex robots [78, 24]. However, despite excellent performance on several tasks [25, 26], most of the work is in simulation, and applying RL to real robots remains a challenging problem [77]. The performance of policies trained in simulation directly applied to real robots can be poor due to the sim-to-real gap. This gap is caused by numerous reasons. First, the process and sensing uncertainties are not considered in the simulation. It is impossible to capture all the physics details in a simulation model and the simulated sensor data can look very different from its real-world counterpart. Second, MDP is assumed in simulated dynamics, while the control and measurement can be asynchronous due to computation and communication delays on real robots. This latency violates the fundamental assumption of MDP, thus can cause failure to some RL algorithms [79]. Third, safety constraints are necessary for real robots but may not be considered in simulation, which leads to differences between the feasible state and action spaces. Also, since the ground-true state may not be available in real-world robots, the policy has to be trained with partial observation, which poses challenges to the RL algorithms. Although the sim-to-real gap remains to

be filled, it is still a promising direction to prototype the policy in simulation and generalize it to real-world applications with a small amount of training.

Another direction is to train directly on real robots, where some of the sim-to-real challenges disappear. Here the policy is trained with real sensor data under uncertainty. Once the policy is successfully trained, its performance is guaranteed. The ability to train on the robot also enables continuous improvement after deployment. With the initial policy trained in simulation, it is important for the robots to learn continuously in the real world to adapt to new environments [77]. Also, learning on real robots is probably the most similar way to human learning [80]. However, it is still a difficult task as resetting may not be feasible, human intervention may be required heavily and the reward can be evaluated only from onboard perception. Compared to learning in simulation, the speed of data collection can be much slower in the real world and the ability to train in parallel is limited due to the budget and human intervention required.

RL researchers have been making great progress to solve the challenges of applying RL to the real world in both sim-to-real and learning on real robot directions. Previous work such as [81] adopted domain randomization (DR) to address uncertainty and latency. Selected system parameters are randomly sampled and noise is added during training, which improves the robustness, but the optimality is lost. In addition, [82, 83] parallelized the simulations on multiple agents to decrease total training time as well as improve the robustness on different terrains. For partially observed systems, a belief space variant of iLQR has been proposed which finds a locally optimal solution under motion and sensing uncertainty [84]. However, computational complexity is increased due to the typically high belief state dimension. To reduce the difference between simulation and the real world, simulated sensor data are used instead of the true states [83]. Adapter networks can be used to convert simulated images to look like their real-world counterparts [85] or the other way around [86]. With these improvements, the trained policy can be directly applied to the real robots with comparable performance [82]. In the optimal control community, previous work [87] conducts a deterministic optimization in simulation and uses an adaptive fuzzy estimator to estimate and compensate for the uncertainty.

In the direction of on-robot training, multiple learning processes can be run with different hyper-parameters on the same robot [88]. Algorithms can be developed to automatically tune the hyper-parameters [51]. These techniques can greatly shorten the tuning process. To tackle the resetting issue, one can design a controller that automatically resets the robot to a random position, to ensure enough exploration of the environment and reduce human intervention [80, 89]. This method requires expert knowledge to design the resetting controller and can be challenging for complex robots. In fact, on-robot training still requires the resetting ability heavily due to the RL algorithm itself. And in many cases, the robots have to be designed in a way that resetting can be easily done [90, 80]. For the delay between measurements and control actions, a recurrent neural network could be trained to extrapolate the observation to when the action is applied from past observations [77], or the observation space can be augmented with previous observations and actions [90]. This idea is similar to the information state method we proposed in POD2C. With the ability to learn online, it may be a good strategy to prototype the policy in simulation, and improve it with a relatively small amount of online training [91] to take the advantages of both simulation and on-robot training.

As the initial work of applying the data-based learning to control approach POD2C we proposed in Section 4.5 to real-world applications, in this chapter, we focus on the problem of learning to control under uncertainty. As shown in Chapter 4, the POD2C algorithm is a generalization from iLQR that converts partially observed problems to “fully observed" problems using the information state. It achieves high training efficiency by decoupling the open-loop and feedback design as shown in Section 4.4. The information state-based LTV-ARMA system identification method is used to estimate the “fully observed" linearized model, which allows us to solve the optimal control problem while the system model is unknown. We have shown that POD2C considers some of the challenges mentioned above such as training efficiency and partial observation in Chapter 4. Here, its performance under process and sensing uncertainties will be studied. The main contributions include: we study the learning under noise problem with our proposed POD2C method. We analyze the efficacy of the POD2C algorithm when the training is carried out under noise. We prove the

convergence of POD2C to the global minimum in the full state observation case. We show that POD2C constructs biased LTV systems and can not converge to the true optimum in the partially observed case, where multiple rollouts need to be averaged to recover the global convergence and optimality.

The rest of the chapter is organized as follows: Section 5.2 provides the optimal control problem formulation. Section 5.3 emphasizes the main focus of this chapter and defines the fully observed and partially observed cases under noise. Section 5.4 analyzes the performance of POD2C directly applied in fully observed and partially observed problems under uncertainty. Empirical results are shown to support the results in Section 5.5.

5.2 Problem Formulation

Let us start by writing the stochastic nonlinear dynamics in discrete time state space form as follows:

$$x_{t+1} = f(x_t, u_t) + \omega_t, \tag{5.1}$$

where x_t is the state, u_t is the control input of the system and ω_t is the process noise. Let us assume the observation model to be of form:

$$z_t = h(x_t) + v_t, \tag{5.2}$$

where z_t is the measurement and v_t is the sensor noise. Let us now define a finite horizon objective function as:

$$J(z_0) = E\left[\sum_{t=0}^{T-1} c(z_t, u_t) + c_T(z_T)\right], \tag{5.3}$$

where $c(z_t, u_t)$ denotes a running incremental cost and $c_T(z_T)$ denotes a terminal cost function. POD2C is proposed to find an observation based feedback policy to minimize the cost function above with deterministic system dynamics and measurements, i.e., ω_t and z_t are zero. The goal of this work is to apply POD2C while the uncertainty exists and analyze its efficacy.

5.3 Learning to Control under Uncertainty with POD2C

In Chapter 4, POD2C is implemented in simulation and we use a noiseless blackbox simulation model to collect data for training. In the real-world environment, there is always noise. Thus the main focus of this work is to solve the partially observed optimal control problem with noise and unknown system dynamics using POD2C. There are two main steps in the POD2C algorithm: open-loop nominal trajectory design using POD-iLQR and closed-loop linear feedback design. Data collection is needed in the forward pass and backward pass of POD-iLQR to evaluate the cost function and estimate the ARMA parameters. In the closed-loop feedback design step, we use the AMRA model fitted in the last backward pass of POD-iLQR, thus no extra data are collected. As data collection is only needed in the open-loop trajectory design, in the following, we study how the noise-corrupted data affect the POD-iLQR algorithm in two cases. In the fully observed case, we assume that there is process noise in the system dynamics but the full state measurements are perfect, i.e.,

$$x_{t+1} = f(x_t, u_t) + \omega_t,\ z_t = x_t. \tag{5.4}$$

In the partially observed case, we assume that both the system dynamics and the sensors are corrupted with noise. In addition, only a subset of the states is measured, i.e.,

$$x_{t+1} = f(x_t, u_t) + \omega_t,\ z_t = C_t x_t + v_t, \tag{5.5}$$

where $C_t \in \mathcal{R}^{n_z \times n_x}, n_z < n_x$ and n_z is the number of outputs.

5.4 Optimality Analysis of POD-iLQR under Uncertainty

According to the results in Section 4.3.1, the POD-iLQR algorithm in the deterministic environment is guaranteed to find the unique global minimum of the open-loop problem in Eq. (4.17). In this section, we analyze the convergence and optimality of POD-iLQR in the two cases described above.

5.4.1 Global Convergence of POD-iLQR in the Fully Observed Case

To identify the LTV model for a nonlinear system under noise, it is important to keep the trajectory close to the nominal trajectory. Thus we implement a feedback term in control using the feedback gain K_t solved from the last backward pass, i.e., $u_t = \bar{u}_t + \delta u_t + K_t(\delta \mathcal{Z}_t)$, where $\delta \mathcal{Z}_t = \mathcal{Z}_t - \bar{\mathcal{Z}}_t$ and δu_t is the input perturbation sampled from a zero-mean i.i.d. process. Then we have the following result:

Lemma 7. *The ARMA model identified in the fully observed case using the method described in Section 4.4.2 is unbiased from the true LTV model.*

Proof. Due to full state observation, the matrix $O^q = \begin{bmatrix} A_{t-q}^T ... A_{t-2}^T C_{t-1}^T, & \cdots, & A_{t-q}^T C_{t-q+1}^T, & C_{t-q}^T \end{bmatrix}^T$ is full column rank. Thus an ARMA model with $q = 1$ can exactly fit the LTV system in Eq. (4.21) according to Proposition 7. Then from Eq. (4.21) and (5.4), we have the following LTV system:

$$\bar{z}_{t+1} + \delta z_{t+1} = A_t(\bar{z}_t + \delta z_t) + B_t(\bar{u}_t + \delta u_t + K_t \delta z_t) + \omega_t, \tag{5.6}$$

which leads to:

$$\delta z_{t+1} = (A_t + B_t K_t)\delta z_t + B_t \delta u_t + \omega_t. \tag{5.7}$$

After n_s number of rollouts, we have:

$$\begin{aligned}
\mathbb{Z} &= \begin{bmatrix} \delta z_{t+1}^{(1)} & \delta z_{t+1}^{(2)} & \dots & \delta z_{t+1}^{(n_s)} \end{bmatrix}, \\
\mathbb{X} &= \begin{bmatrix} \delta z_t^{(1)} & \delta z_t^{(2)} & \dots & \delta z_t^{(n_s)} \\ \delta u_t^{(1)} & \delta u_t^{(2)} & \dots & \delta u_t^{(n_s)} \end{bmatrix}, \\
\mathbb{Z} &= \begin{bmatrix} A_t + B_t K_t \vdots B_t \end{bmatrix} \mathbb{X} + \begin{bmatrix} \omega_t^{(1)} & \omega_t^{(2)} & \dots & \omega_t^{(n_s)} \end{bmatrix}.
\end{aligned} \tag{5.8}$$

Notice that ω_t is uncorrelated with δz_t and δu_t. Thus in the least square calculation, the second term on the RHS goes to zero as the number of rollouts n_s increases. Then the least square result of the

above equation can be written as:

$$\left[\hat{A}_t \vdots \hat{B}_t\right] = \mathbb{Z}\mathbb{X}^\intercal(\mathbb{X}\mathbb{X}^\intercal)^{-1} = \left[A_t + B_t K_t \vdots B_t\right]. \tag{5.9}$$

As K_t is known from the last backward pass, it is trivial to recover A_t, B_t. Thus the fitted ARMA model equals the true LTV model in Eq. (4.21). □

Next, we can show the following result:

Theorem 5. *Let the instantaneous cost function be quadratic in control and let the cost functions $l(\cdot)$, $c_T(\cdot)$, the drift $f(\cdot)$ and the input influence function $g(\cdot)$ be $\mathcal{C}^2$, i.e., twice continuously differentiable. The POD-iLQR algorithm started at a feasible initial information state $\mathcal{Z}_0$ converges to the unique global minimum of the open-loop problem in Eq.* (4.17) *when applied to the fully observed system in Eq.* (5.4) *with small zero-mean noise ω_t.*

Proof. From Lemma 7, we know that the identified LTV model in the fully observed case is the same as in the noiseless case for small enough noise. Thus the results from the backward pass are also identical. We can write the identified LTV-ARMA model as in Eq. (4.41). Let us denote the update direction calculated in the forward pass under process noise as $d_{s,t} = [\delta\mathcal{Z}'_{t+1} \;\; \delta\mathcal{Z}'_t \;\; \delta u'_t]'$, gradient of the system dynamics constraint function as $\nabla h(\bar{\mathcal{Z}}_{t+1}, \bar{\mathcal{Z}}_t, \bar{u}_t) = [I \;\; -\mathcal{A}_t \;\; -\mathcal{B}_t]$. Let $\mathcal{F}^t$ denote the history of the algorithm till time t. Then, due to the zero mean noise ω_t, it is easy to see that the expected descent direction conditioned on the history $\mathcal{F}^t$, $E[d_{s,t}/\mathcal{F}^t] = \bar{d}_{s,t}$, where $\bar{d}_{s,t}$ denotes the true update direction (without noise). We know from Lemma 4, that $\bar{d}_{s,t}$ is a descent direction of the cost function, i.e., $\bar{d}'_{s,t}\nabla J'_t$ is always negative. Thus the expected update direction is a descent direction. Then using the line search condition of ILQR, similar to Theorem 1, it can be shown that:

$$E[J_{t+1}/\mathcal{F}^t] \leq J_t - \beta_t||\nabla J_t||||\bar{d}_{s,t}||, \tag{5.10}$$

for some $\beta_t > 0$. Then , using the Supermartingale Convergence Theorem [92], it follows that,

almost surely:

$$\sum_t \beta_t ||\nabla J_t|| < \infty, \tag{5.11}$$

which implies that $\nabla J_t \to 0$ almost surely, i.e., the algorithm converges to a stationary point of the cost function. Next, using Theorem 2 in Chapter 2 and Theorem 3 in Section 4.3.1, POD-iLQR is guarenteed to converge to the global minimum of the open-loop problem in Eq. (4.17). □

5.4.2 Biased Update Direction of POD-iLQR in the Partially Observed Case

For the partially observed case, we need to choose a large enough q such that matrix O^q is full column rank. Also, we use the feedback gain K_t from the last backward pass to keep the trajectory close to the nominal trajectory. Then we have the following negative result:

Lemma 8. *If directly applied to the partially observed system in Eq.* (5.5)*, the backward pass shown in Algorithm 4 generates a biased update direction.*

Proof. Starting from Eq. (5.5), we have the following LTV system:

$$z_{t+1} = C_t A_t x_t + C_t B_t u_t + C_t \omega_t + v_{t+1}. \tag{5.12}$$

Then we can write the output equation for past q time-steps as Eq. (4.26). Assuming that the q we choose satisfies Assumption 1, we can solve for the unique solution of δx_{t-q} as:

$$\delta x_{t-q} = O^{q^+}(\delta Z_t^q - G^q \delta U_t^q - G_\omega^q \Omega_t^q - V_t^q). \tag{5.13}$$

Now, the system output at time t can be written as:

$$\begin{aligned}\delta z_t = C_t A_{t-1} ... A_{t-q} \delta x_{t-q} + \begin{bmatrix} C_t B_{t-1} & C_t A_{t-1} B_{t-2} & \cdots & C_t A_{t-1} ... B_{t-q} \end{bmatrix} \delta U_t^q \\ + \begin{bmatrix} C_t & C_t A_{t-1} & \cdots & C_t A_{t-1} ... A_{t-q+1} \end{bmatrix} \Omega_t^q + v_t.\end{aligned} \tag{5.14}$$

Let us denote

$$\begin{aligned}
\alpha_t &= C_t A_{t-1} ... A_{t-q} O^{q^+}, \\
\beta_t &= \begin{bmatrix} C_t B_{t-1} & C_t A_{t-1} B_{t-2} & \cdots & C_t A_{t-1} ... B_{t-q} \end{bmatrix} - \alpha_t G^q, \\
\beta_t^D &= \begin{bmatrix} C_t & C_t A_{t-1} & \cdots & C_t A_{t-1} ... A_{t-q+1} \end{bmatrix} - \alpha_t G_\omega^q,
\end{aligned} \tag{5.15}$$

and the unique solution for δx_{t-q} is substituted to get:

$$\delta z_t = \alpha_t \delta Z_t^q + \beta_t \delta U_t^q + \beta_t^D \Omega_t^q - \alpha_t V_t^q + v_t. \tag{5.16}$$

After taking n_s number of rollouts and apply the linear least square as described in Section 4.4.2, the estimated ARMA model parameters can be written as:

$$\begin{aligned}
\begin{bmatrix} \hat{\alpha}_t & \hat{\beta}_t \end{bmatrix} = \begin{bmatrix} \alpha_t & \beta_t \end{bmatrix} + (\beta_t^D \begin{bmatrix} \Omega_t^{q,(1)} & \Omega_t^{q,(2)} & \cdots & \Omega_t^{q,(n_s)} \end{bmatrix} - \alpha_t \begin{bmatrix} V_t^{q,(1)} & V_t^{q,(2)} & \cdots & V_t^{q,(n_s)} \end{bmatrix} \\
+ \begin{bmatrix} v_t^{(1)} & v_t^{(2)} & \cdots & v_t^{(n_s)} \end{bmatrix}) \mathbb{X}^\mathsf{T} (\mathbb{X}\mathbb{X}^\mathsf{T})^{-1},
\end{aligned} \tag{5.17}$$

where in this case,

$$\mathbb{X} = \begin{bmatrix} \delta Z_t^{q,(1)} & \delta Z_t^{q,(2)} & \dots & \delta Z_t^{q,(n_s)} \\ \delta U_t^{q,(1)} & \delta U_t^{q,(2)} & \dots & \delta U_t^{q,(n_s)} \end{bmatrix}. \tag{5.18}$$

As δZ_t^q is correlated with Ω_t^q and V_t^q, the second term on the RHS of Eq. (5.17) is nonzero. Thus the estimated ARMA parameters are biased from the true values in $\begin{bmatrix} \alpha_t & \beta_t \end{bmatrix}$. Further, the update direction generated from Algorithm 4 is biased. □

With the biased update direction, one can expect that the POD-iLQR algorithm can no longer converge to the true minimum. In this case, multiple rollouts have to be averaged to recover the convergence to the true minimum. In the next section, we show empirical evidence for Theorem 5 and Lemma 8.

5.5 Empirical Results

We use MuJoCo, a physics engine [38], as a blackbox to collect the data needed for the open-loop nominal trajectory design and the closed-loop feedback gain. We verify our results on two nonlinear systems with their initial configuration shown in Fig. 5.1. All simulations are done on a machine with the following specifications: AMD Ryzen 3700X 8-Core CPU@3.59 GHz, with 16 GB RAM, with no multi-threading. The algorithms are implemented in MatLab.

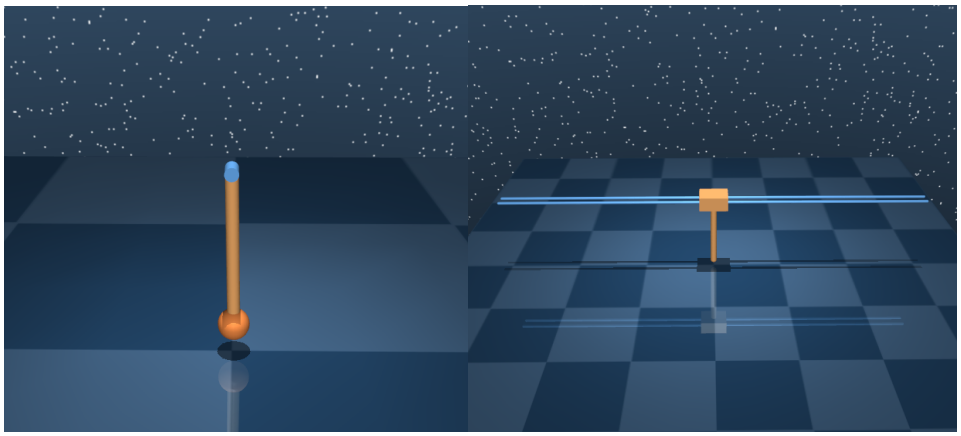


Figure 5.1: Models simulated in MuJoCo in their initial states.

5.5.1 Model Description

Here we provide details of the MuJoCo models used in our simulations [93].

Pendulum: The single pendulum model is a pole hinged to a fixed point. There are two state variables: angle and angular rate of the pole. The task is to swing up and balance the pole in the upright position.

Cart-Pole: The four-dimensional under-actuated cart-pole model includes a cart moving on the x-axis and a pole linked to it with a hinge joint. The only actuation is the force on the cart. The state comprises the angle of the pole, the cart's horizontal position, and their rates. The task is to

swing up and balance the pole in the middle of the rail within a given horizon.

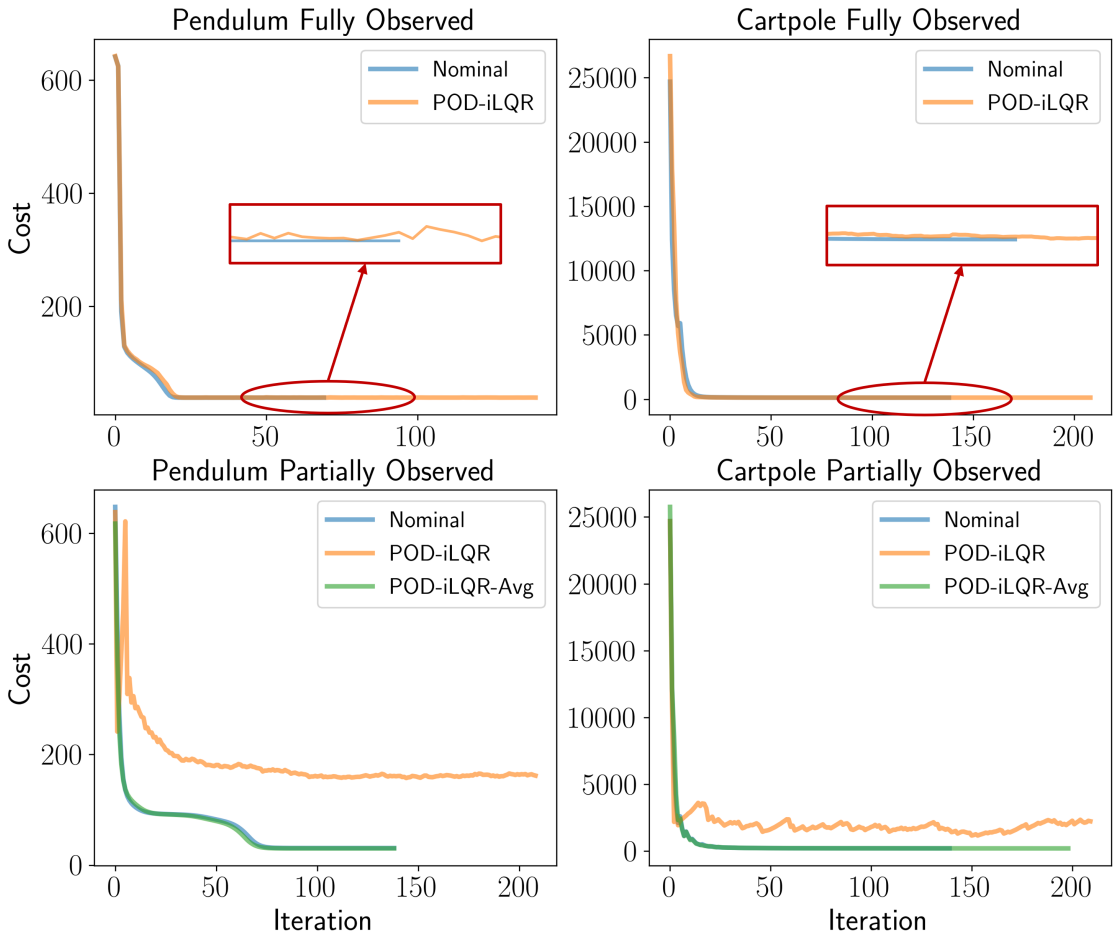


Figure 5.2: Convergence comparison in fully and partially observed cases.

5.5.2 POD-iLQR in the Fully Observed Case

As we assume perfect measurements in the fully observed case, the process noise is the only uncertainty we add to the dynamics. In addition to that, we sample the initial state deviation δx_0 from a zero-mean random process. The standard deviation of the process noise ω_t is 10% w.r.t. the standard deviation of the initial state deviation δx_0. For each system tested, we run POD-iLQR as it is in the noiseless system as well as in the fully observed system with process noise. The number

of steps in the horizon is fixed so in the data collection step, each rollout takes the same number of steps. In the cost function, the running cost $l(x_t) = x_t' Q x_t$, where x_t is the error between the current state and the target state at time t. The cost parameter Q remains the same throughout the horizon except for the terminal step. The termination criterion is that the convergence rate is lower than a threshold or it reaches the max iteration number. We compare the cost convergence curves in the first row of Fig. 5.2. The "nominal" curve shows the cost convergence of applying POD-iLQR on the full state noiseless case. The curve labeled "POD-iLQR" shows the cost convergence in the full state case under process noise. From the plot, both curves almost overlap each other and converge to the same result. The "POD-iLQR" curves in the zoomed-in view have some ups and downs due to the process noise in the forward pass. This verifies the proof in Theorem 5 which shows that the expected update direction, not the actual update direction is a descending direction. Thus in the fully observed case, POD-iLQR can be directly applied without modification and it is guaranteed to converge to the global minimum. One thing to notice is that the total number of rollouts needed under noise is larger than in the noiseless case to make sure that the correlation goes to zero as shown in Lemma 7.

5.5.3 POD-iLQR in the Partially Observed Case

In the partially observed case, we only measure the positions, not their rates. For the pendulum example, the observed state is just the angle of the pole. In the cart-pole, we only measure the position of the cart and the angle of the pole. To simulate the measurements in simulation, we add sensor noise v_t to the measurements. Thus there are both process and measurement noise in the environment to simulate model and sensor uncertainties. The standard deviations of the process noise ω_t and the measurement noise v_t are both 10% w.r.t. the standard deviation of the initial state deviation δx_0. We found that in this observation setting, $q = 2$ satisfies the observability assumption. So in the LTV-ARMA system identification step, we fit an ARMA model with $q = 2$ for both the pendulum and cart-pole. To evaluate the cost function, we only use the measurements instead of the states. Similar to the full state observation case, the cost function is quadratic in the information state and the control input. Three cost curves are shown in the second row of Fig. 5.2. The curve

labeled “nominal" shows the cost convergence of POD-iLQR in the partially observed but noiseless environment. Then in the case labeled “POD-iLQR", we directly apply the algorithm to the partially observed case with both process and measurement noise. In the case labeled “POD-iLQR-Avg", we run multiple rollouts for each set of control perturbation δu_t and used the mean trajectory in the ARMA model fitting step to average out the noise. From the plots, it is shown that in both the pendulum and the cart-pole, the cost curve of the averaging method matches the nominal cost curve and they converge to the same result. The outlier curve is from the experiment where we directly applied POD-iLQR without averaging. Due to the noise corrupted system dynamics and measurements, the cost curve has more oscillation and failed to converge to the true minimum. Thus if directly implemented on real robots without averaging, POD-iLQR will generate a biased result without the optimality guarantee anymore. To make sure the noise is averaged out in the averaging method, the total number of rollouts needed in the ARMA model fitting is increased from n_s to $n_s \times n_{avg}$, where n_s is the number of rollouts in one ARMA model fitting without averaging and n_{avg} is the number of rollouts needed for each control perturbation set. And the total time taken during training will be much higher than the case without averaging.

5.6 Conclusions

This chapter introduces an optimal motion planning/trajectory design algorithm for partially observed systems by transforming them into fully observed problems using the information state. The main focus is to analyze its performance under uncertainty and pave the way for future implementation on real-world robots. The algorithm is proved to converge to the global minimum in the fully observed case with process noise only. It is shown that the algorithm is biased and does not converge to the global minimum for partially observed systems with process and measurement noise. In this case, multiple rollouts can be averaged to recover optimality and convergence in the ARMA model identification step at the cost of a longer training time. The empirical results are shown to verify the analysis. In our opinion, this algorithm has advantages in optimality and training efficiency when applied to real-world robots. The actual performance on real robots will be explored in future work.

6. CONCLUSIONS

In this dissertation, we study the problem of designing an optimal feedback law in a data-based fashion. We first derive a decoupling principle which reveals the near-optimality of the deterministic optimal control policy applied to the stochastic system. A highly efficient decoupled data-based framework (D2C) is developed based on the decoupling principle. We show that the closed-loop control policy generated by D2C is within $O(\epsilon^4)$ to the true stochastic optimal policy, where ϵ is the noise coefficient. The advantages of D2C on training efficiency, energy efficiency and closed-loop robustness to noise are demonstrated by comparing to DDPG and a model-based shape control method on several nonlinear, high-dimensional robotic systems and tensegrity structures. By implementing a data-based generalized iLQR in the open-loop optimal trajectory design step, we further improve the training efficiency of the D2C approach and prove its convergence to the global optimum of the optimal control problem. Simulation experiments on high-dimensional nonlinear complex systems show the advantages of the improved D2C on 1) training efficiency, 2) closed-loop performance and 3) reliability of training over state-of-the-art RL methods. The D2C approach is extended to partially observed problems where only a subset of the states can be observed. The extended method (POD2C) includes a partially observed generalization of iLQR using the LTV-ARMA system identification method for the open-loop optimal trajectory design and a specific LQG on the information state-based ARMA model for the closed-loop feedback design. We prove that the POD2C solution satisfies the generalized minimum principle, thus it is the global optimality of the partially observed problem. The efficacy of POD2C is tested on partially observed, high-dimensional, highly nonlinear complex robotic systems. We also study the learning under uncertainty problem. We show that in the fully observed case where process noise exists in the system dynamics, the POD2C algorithm still converges to the global optimum. However, in the partially observed case where both the system dynamics and the measurements are corrupted by noise, we have a negative result that POD2C will generate a biased solution and multiple rollouts need to be averaged to recover the optimality.

We propose four directions that can be considered for future development.

1. We have shown that POD2C generates a biased result when directly applied to the partially observed system with the process and sensing noise. What other information is needed and how to modify POD2C to eliminate this bias are left unknown. Also, there are other sources of uncertainty such as biased model parameters and disturbance. These uncertainties may not be modeled as zero-mean noise. It is interesting to explore how to learn and compensate for these uncertainties as well.

2. The systems we have considered so far have no hard constraints such as the contact with the ground. The system dynamics with hard contact may not be smooth and the dynamics before contract, at the contact point and after contact may be different. Thus, the system linearization and identification can be problematic. For the fixed timestep discretization we use, the contact point may be hard to capture accurately when the contact happens within a timestep. A generalized D2C framework for solving the optimal control problems with unsmooth and changing dynamics may be proposed to cover this class of applications.

3. Although D2C is shown to be highly efficient, the current time taken during training is still too long to run in real-time. If training in real-time is made possible, the D2C with replanning can be a practical global control policy. There are several potential directions such as parallelization of rollouts and more efficient parametrization.

4. The hyperparameters needed to be tuned in D2C are the cost parameters. It usually takes a few training to find a set of good parameters. If a systematic auto-tuning method can be proposed to learn the best hyperparameters during training, the expertise needed to use D2C can be further reduced.

REFERENCES

[1] R. E. Skelton and M. C. de Oliveira, *Tensegrity Systems*. Springer US, 2009.

[2] M. Chen, R. Goyal, M. Majji, and R. E. Skelton, "Design and analysis of a growable artificial gravity space habitat," *Aerospace Science and Technology*, vol. 106, p. 106147, 2020.

[3] M. Chen, R. Goyal, M. Majji, and R. E. Skelton, "Deployable tensegrity lunar tower," in *Earth and Space 2021*, pp. 1079–1092, American Society of Civil Engineers, 2021.

[4] P. R. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*, vol. 75. SIAM, 2015.

[5] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Courier Corporation, 2012.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Two Volume Set*. Athena Scientific, 2nd ed., 1995.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[8] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *International Conference for Robotics and Automation (ICRA)*, 2016.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[10] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," *arXiv preprint arXiv:1502.05477*, 2017.

[11] D. Jacobsen and D. Mayne, *Differential Dynamic Programming*. Elsevier, 1970.

[12] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of American Control Conference*, pp. 300 – 306, 2005.

[13] W. Li and E. Todorov, "Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system," *International Journal of Control*, vol. 80, no. 9, pp. 1439–1453, 2007.

[14] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, 2012.

[15] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill, 2015.

[16] K. S. Parunandi and S. Chakravorty, "T-pfc: A trajectory-optimized perturbation feedback control approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3457–3464, 2019.

[17] R. Skelton and G. Shi, "Iterative identification and control using a weighted q-markov cover with measurement noise," *Signal Processing*, vol. 52, no. 2, pp. 217–234, 1996. Subspace Methods, Part II: System Identification.

[18] S. Chen, S. A. Billings, and P. M. Grant, "Non-linear system identification using neural networks," *International Journal of Control*, vol. 51, no. 6, pp. 1191–1214, 1990.

[19] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[20] Z. Littlefield, D. Klimenko, H. Kurniawati, and K. E. Bekris, "The importance of a suitable distance function in belief-space planning," in *Robotics Research*, pp. 683–700, Springer, 2018.

[21] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observatoins," in *Proceedings of Robotics: Science and Systems (RSS)*, 2010.

[22] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*. Routledge, New York, 1975.

[23] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[24] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[26] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[27] W. Yuhuai, M. Elman, L. Shun, G. Roger, and B. Jimmy, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *arXiv preprint arXiv:1708.05144*, 2017.

[28] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] E. Theoddorou, Y. Tassa, and E. Todorov, "Stochastic differential dynamic programming," in *Proceedings of American Control Conference*, 2010.

[30] S. Levine and K. Vladlen, "Learning complex neural network policies with trajectory optimization," in *Proceedings of the International Conference on Machine Learning*, 2014.

[31] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2007.

[32] M. Deisenroth and C. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *International Conference on Machine Learning (ICML)*, 2011.

[33] D. Mitrovic, S. Klanke, and S. Vijayakumar, *Adaptive Optimal Feedback Control with Learned Internal Dynamics Models, in From Motor Learning to Interaction Learning in Robots. Studies in Computational Intelligence, vol 264*. Springer, Berlin, 2010.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[35] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade, "Towards generalization and simplicity in continuous control," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[36] H. Kushner and G. Yin, *Stochastic Approximation and Recursive Algorithms and applications*. Springer, NY, 2003.

[37] R. Versyhnin, *High Dimensional Probability: An Introduction with Application to Data Science*. Cambridge University Press, Cambridge, UK, 2018.

[38] T. Emanuel, E. Tom, and Y. Tassa, "Mujoco: A physics engine for model-based control," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[39] T. Yuval and *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[40] M. Plappert, "keras-rl." https://github.com/keras-rl/keras-rl, 2016.

[41] D. Rus, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 5, pp. 467–475, 2015.

[42] C. Majidi, "Soft robotics: A perspective — current trends and prospects for the future," *Soft Robotics*, vol. 1, pp. 5–11, 2014.

[43] R. Goyal and R. E. Skelton, "Tensegrity system dynamics with rigid bars and massive strings," *Multibody System Dynamics*, vol. 46(3), pp. 203–228, 2019.

[44] R. E. Skelton and M. C. de Oliveira, "Optimal tensegrity structures in bending: The discrete michell truss," *Journal of the Franklin Institute*, vol. 347(1), pp. 257–283, 2010.

[45] K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of The Royal Society Interface*, vol. 11, no. 98, 2014.

[46] A. P. Sabelhaus, A. K. Akella, Z. A. Ahmad, and V. SunSpiral, "Model-predictive control of a flexible spine robot," in *2017 American Control Conference (ACC)*, pp. 5051–5057, 2017.

[47] H. Karnan, R. Goyal, M. Majji, R. E. Skelton, and P. Singla, "Visual feedback control of tensegrity robotic systems," *2017 IEEE/RSJ-IROS*, pp. 2048–2053, 2017.

[48] A. G. Tibert and S. Pellegrino, "Deployable tensegrity mast," *In: 44th AIAA/ASME/ASCE, Structures, Structural Dynamics and Materials Conference and Exhibit, Norfolk , VA, USA.*, p. 1978, 2003.

[49] C. Paul, F. J. Valero-Cuevas, and H. Lipson, "Design and control of tensegrity robots for locomotion," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 944–957, 2006.

[50] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 634–641, May 2017.

[51] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, PMLR, 10–15 Jul 2018.

[52] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596, PMLR, 10–15 Jul 2018.

[53] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, pp. 2967–2986, 2014.

[54] M. N. G. Mohamed, S. Chakravorty, and R. Wang, "Optimality and tractability in stochastic nonlinear control," *arXiv preprint arXiv:2004.01041*, 2020.

[55] W. Heemels, K. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 3270–3285, 2012.

[56] H. Li, Y. Shi, W. Yan, and R. Cui, "Periodic event-triggered distributed receding horizon control of dynamically decoupled linear systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10066–10071, 2014. 19th IFAC World Congress.

[57] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, p. 1–51, 1995.

[58] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[59] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[60] S. M. Allen and J. W. Cahn, "A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening," *Acta Metallurgica*, vol. 27, no. 6, pp. 1085–1095, 1979.

[61] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup, "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control," *Reproducibility in Machine Learning Workshop, ICML'17*, 2017.

[62] S. Chakravorty, R. Wang, and M. N. G. Mohamed, "On the convergence of reinforcement learning," *arXiv preprint arXiv:2011.10829*, 2020.

[63] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, vol. 734. John Wiley & Sons, 2011.

[64] A. Bryson and H. Y.-C., *Applied Optimal Control: Optimization, Estimation and Control*. Washington: Hemisphere Pub. Corp., 1975.

[65] Z. Cheng, J. Ma, X. Zhang, F. L. Lewis, and T. H. Lee, "Neural network ilqr: A reinforcement learning architecture for trajectory optimization," *arXiv preprint arXiv:2011.10737*, 2020.

[66] M. Schimel, T.-C. Kao, K. T. Jensen, and G. Hennequin, "Ilqr-vae: control-based learning of input-driven dynamics with applications to neural data," *bioRxiv*, 2021.

[67] T. Zong, L. Sun, and Y. Liu, "Reinforced ilqr: A sample-efficient robot locomotion learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5906–5913, IEEE, 2021.

[68] J. van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31(11), pp. 1263–1278, 2012.

[69] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee, "Motion planning under uncertainty for robotic tasks with long time horizons," *International Journal of Robotics Research*, vol. 30, pp. 308–323, 2010.

[70] R. Platt, "Convex receding horizon control in non-gaussian belief space," in *Algorithmic Foundations of Robotics X*, pp. 443–458, Springer, 2013.

[71] M. Ghahramani and A. Thavaneswaran, "Financial applications of arma models with garch errors," *The Journal of Risk Finance*, 2006.

[72] I. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems part i: Deterministic non-linear systems," *International Journal of Control*, vol. 41, no. 2, pp. 303–328, 1985.

[73] I. J. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems part ii: Stochastic non-linear systems," *International Journal of Control*, vol. 41, no. 2, pp. 329–344, 1985.

[74] F. N. Chowdhury, "Input-output modeling of nonlinear systems with time-varying linear models," *IEEE Transactions on Automatic Control*, vol. 45, no. 7, pp. 1355–1358, 2000.

[75] E. Hernandez and Y. Arkun, "Control of nonlinear systems using polynomial arma models," *AIChE Journal*, vol. 39, no. 3, pp. 446–460, 1993.

[76] M. Majji, J.-N. Juang, and J. L. Junkins, "Time-varying eigensystem realization algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 1, pp. 13–28, 2010.

[77] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[78] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[79] T. Xiao, E. Jang, D. Kalashnikov, S. Levine, J. Ibarz, K. Hausman, and A. Herzog, "Thinking while moving: Deep reinforcement learning with concurrent control," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[80] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real world robotic reinforcement learning," in *International Conference on Learning Representations*, 2020.

[81] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, 2021.

[82] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, "Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments," *arXiv preprint arXiv:2005.13857*, 2020.

[83] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," *arXiv preprint arXiv:2109.11978*, 2021.

[84] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[85] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11154–11163, 2020.

[86] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12619–12629, 2019.

[87] A. Arab and Y. Mousavi, "Optimal control of wheeled mobile robots: From simulation to real world," in *2020 American Control Conference (ACC)*, pp. 583–589, 2020.

[88] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, and K. Tumer, "Collaborative evolutionary reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 3341–3350, PMLR, June 2019.

[89] M. Bloesch, J. Humplik, V. Patraucean, R. Hafner, T. Haarnoja, A. Byravan, N. Y. Siegel, S. Tunyasuvunakool, F. Casarini, N. Batchelor, F. Romano, S. Saliceti, M. Riedmiller, S. M. A. Eslami, and N. Heess, "Towards real robot learning in the wild: A case study in bipedal locomotion," in *Proceedings of the 5th Conference on Robot Learning*, vol. 164 of *Proceedings of Machine Learning Research*, pp. 1502–1511, PMLR, 2022.

[90] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning.," in *Robotics: Science and Systems*, 2019.

[91] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*, pp. 651–673, PMLR, 2018.

[92] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1st ed., 1996.

[93] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm_control: Software and tasks for continuous control," *Software Impacts*, vol. 6, p. 100022, 2020.

APPENDIX

A.1 Dynamics of General Tensegrity Structures

This section provides a brief overview of the non-linear dynamic model of a general tensegrity structure. The final form of class-1 tensegrity dynamics is formulated as a second order matrix differential equation [43].

$$\ddot{N}M_s + NK_s = W, \tag{A.1}$$
$$M_s = \begin{bmatrix} C_{nb}^{\mathsf{T}}(C_b^{\mathsf{T}}\hat{J}C_b + C_r^{\mathsf{T}}\hat{m}_b C_r) & C_{ns}^{\mathsf{T}}\hat{m}_s \end{bmatrix},$$
$$K_s = \begin{bmatrix} C_s^{\mathsf{T}}\hat{\gamma}C_{sb} - C_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\hat{\lambda}C_b & C_s^{\mathsf{T}}\hat{\gamma}C_{ss} \end{bmatrix},$$

where λ represents the force density (compressive force per unit length) in the bar, given by:

$$\hat{\lambda} = -\hat{J}\hat{l}^{-2}\lfloor \dot{B}^{\mathsf{T}}\dot{B} \rfloor - \frac{1}{2}\hat{l}^{-2}\lfloor B^{\mathsf{T}}(W - S\hat{\gamma}C_s)C_{nb}^{\mathsf{T}}C_b^{\mathsf{T}} \rfloor, \tag{A.2}$$

and $N = \begin{bmatrix} n_1 & n_2 & \cdots & n_{2\beta+\sigma} \end{bmatrix} \in \mathbb{R}^{3\times(2\beta+\sigma)}$ represents the matrix containing the node position vectors $n_i \in \mathbb{R}^{3\times 1}$, β represents the number of bars, and σ represents the number of string-to-string nodes. The acceleration vector corresponding to the i^{th} node is represented by $\ddot{n}_i$, which forms the matrix $\ddot{N} = \begin{bmatrix} \ddot{n}_1 & \ddot{n}_2 & \cdots & \ddot{n}_{2\beta+\sigma} \end{bmatrix}$. The bar matrix $B = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \cdots & \vec{b}_\beta \end{bmatrix} \in \mathbb{R}^{3\times\beta}$ and string matrix $S = \begin{bmatrix} \vec{s}_1 & \vec{s}_2 & \cdots & \vec{s}_\alpha \end{bmatrix} \in \mathbb{R}^{3\times\alpha}$ contain bar vectors $\vec{b}_i$ and string vectors $\vec{s}_i$, respectively. The diagonal matrices $\hat{m}_b$ and $\hat{m}_s$ are formed by arranging bar masses and string point masses along the diagonal elements, respectively, and $\hat{J}$ is a diagonal matrix with $J_i = \frac{m_{b_i}}{12} + \frac{m_{b_i} r_{b_i}^2}{l_i^2}$ as the diagonal element that allows to accommodate the inertia in the bars. The connectivity matrices $C_{nb}, C_b, C_r, C_{ns}, C_s, C_{sb}$, and C_{ss} define the connections between different nodes to form bar vector, string vectors and string-to-string node positions. The external force matrix $W =$

$\begin{bmatrix} w_1 & w_2 & \cdots & w_{2\beta+\sigma} \end{bmatrix}$ represents the matrix containing external force vector $\vec{w}_i$ corresponding to each node position vector $\vec{n}_i$. The term "force density in strings" is denoted by γ_i to describe tensile force per unit length in the i^{th} string member and $\hat{\gamma}$ represents the diagonal matrix formed from γ_i as its diagonal elements. Equation (A.2) provides the analytical formula to calculate the diagonal matrix $\hat{\lambda}$ where $\lfloor \circ \rfloor$ operator sets every off-diagonal element of the square matrix to zero. The diagonal matrix $\hat{l}$ is formed by the length of the bar members where l_i denotes the length of the i^{th} bar. Please refer to [43] for the detailed derivation of this dynamics model.

Dynamics Model for Class-k Tensegrity Structures

The model presented above is formulated for class-1 tensegrity structures. Any class-k structure can be described as a special case of class-1 structures by adding constraints on k nodes to have the same node positions. These constraints are written in the linear form as:

$$NP = D, \tag{A.3}$$

where matrix $P \in \mathbb{R}^{(2\beta+\sigma)\times c}$ and matrix $D \in \mathbb{R}^{3\times c}$ can be written from the observation to provide constraints and c is the number of added constraints [43]. The added constraints can be accommodated in the class-1 tensegrity dynamics by introducing Lagrange multiplier $\Omega \in \mathbb{R}^{3\times c}$ that constraints the motion to a certain space by adding Lagrange constraint forces of the form ΩP^{T}. The full-order dynamics model for class-k tensegrity structure is written as:

$$\ddot{N}M_s + NK_s = W + \Omega P^{\mathsf{T}}, \tag{A.4}$$

with a modification for force density in the bars as:

$$\hat{\lambda} = -\hat{J}\hat{l}^{-2}\lfloor \dot{B}^{\mathsf{T}}\dot{B} \rfloor - \frac{1}{2}\hat{l}^{-2}\lfloor B^{\mathsf{T}}(W + \Omega P^{\mathsf{T}} - S\hat{\gamma}C_s)C_{nb}^{\mathsf{T}}C_b^{\mathsf{T}} \rfloor. \tag{A.5}$$

The degree of freedom for the entire structure is reduced because of the above-mentioned

constraints. This reduction in the number of allowed displacement dimensions can be captured in a reduced dimensional space by reducing the order of the model. The following second order matrix differential equation provides the reduced order dynamic model as:

$$\ddot{\eta}_2 M_2 + \eta_2 K_2 = \widetilde{W}, \tag{A.6}$$

where $M_2 = U_2^\mathsf{T} M_s U_2$, $K_2 = U_2^\mathsf{T} K_s U_2$, $\widetilde{W} = W U_2 - \eta_1 U_1^\mathsf{T} K_s U_2$, and the remaining variables are generated from the singular value decomposition (SVD) of the full-column rank matrix P as:

$$NP = NU\Sigma V^\mathsf{T} = N[U_1\ U_2] \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix} \begin{bmatrix} V^\mathsf{T} \end{bmatrix} = D, \tag{A.7}$$

where we use $[\eta_1\ \ \eta_2] \triangleq NU$ to get:

$$\eta_1 = DV\Sigma_1^{-1}, \ \ \dot{\eta}_1 = 0, \ \ \ddot{\eta}_1 = 0. \tag{A.8}$$

Notice that η_1 represents the constraint space in the transformed coordinates and η_2 represents the reduced order space where the motion is present. The Lagrange multiplier is calculated at each step to introduce constraint forces by solving the following algebraic equation:

$$\eta_1 U_1^\mathsf{T} K_s M_s^{-1} U_1 + \eta_2 U_2^\mathsf{T} K_s M_s^{-1} U_1 = W M_s^{-1} U_1 + \Omega V \Sigma_1^\mathsf{T} U_1^\mathsf{T} M_s^{-1} U_1. \tag{A.9}$$

Please refer to Dr. Raman Goyal's work [43] for the analytic solution to this linear algebra problem.

A.2 Model-Based Shape Control for Class-k Tensegrity Systems

An important step in writing the control of this non-linear dynamic system into a linear programming problem is to be able to write the force densities in the bar $\lambda = [\lambda_1\ \lambda_2\ \cdots\ \lambda_\beta]^\mathsf{T}$ in terms of the linear function of force densities in the strings $\gamma = [\gamma_1\ \gamma_2\ \cdots\ \gamma_\alpha]^\mathsf{T}$. Let us write the i^{th} diagonal

element of the matrix $\hat{\lambda}$ from Eq. (A.5) as:

$$\lambda_i = -J_i l_i^{-2} e_i^{\mathsf{T}} \lfloor \dot{B}^{\mathsf{T}} \dot{B} \rfloor e_i - \frac{1}{2} l_i^{-2} e_i^{\mathsf{T}} \lfloor B^{\mathsf{T}} (W + \Omega P^{\mathsf{T}} - S\hat{\gamma} C_s) C_{nb}^{\mathsf{T}} C_b^{\mathsf{T}} \rfloor e_i, \tag{A.10}$$

which is written using the identity $\hat{x}y = \hat{y}x$, for x and y being the column vectors, and stacking all the scalars into a column vector as:

$$\lambda = \Lambda\gamma + \tau, \tag{A.11}$$

where

$$\Lambda = \begin{bmatrix} \Lambda_1^{\mathsf{T}} \; \Lambda_2^{\mathsf{T}} \; \cdots \; \Lambda_\beta^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \; \tau = \begin{bmatrix} \tau_1^{\mathsf{T}} \; \tau_2^{\mathsf{T}} \; \cdots \; \tau_\beta^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{A.12}$$

$$\tau_i = -J_i l_i^{-2} ||\dot{b}_i||^2 - \frac{1}{2} l_i^{-2} b_i^{\mathsf{T}} (W + \Omega P^{\mathsf{T}}) C_{nb}^{\mathsf{T}} C_b^{\mathsf{T}} e_i, \tag{A.13}$$

$$\Lambda_i = \frac{1}{2} l_i^{-2} b_i^{\mathsf{T}} \widehat{S(C_s C_{nb}^{\mathsf{T}} C_b^{\mathsf{T}} e_i)}, \; for \; i = 1, 2 \; \cdots \; \beta. \tag{A.14}$$

A.2.1 Controller for Reduced Order Dynamics Model

In this section, we write down the control algorithm to control the position of certain nodes in the structure. Let us define the position of those nodes by $Y = LNR$, where L is a matrix that defines the x, y or z coordinates of the node and R matrix defines which nodes to be controlled. $\bar{Y}$ defines the final desired location of the nodes that we want to move from Y to $\bar{Y}$. Therefore, the error in the positions at any time is written as:

$$E = LNR - \bar{Y} = L(\eta_1 U_1^{\mathsf{T}} + \eta_2 U_2^{\mathsf{T}})R - \bar{Y}, \tag{A.15}$$

and the first and second derivatives of the error with respect to time is written as:

$$\dot{E} = L\dot{\eta}_2 U_2^{\mathsf{T}} R, \quad \ddot{E} = L\ddot{\eta}_2 U_2^{\mathsf{T}} R, \tag{A.16}$$

where $\dot{\eta}_1 = \ddot{\eta}_1 = 0$ was used from the dynamics model formulation. Now, a second order differential equation in the error dynamics is used to move the nodes from the current position to the desired position by aptly choosing the control gain parameters matrices Ψ and Θ:

$$\ddot{E} + \dot{E}\Psi + E\Theta = 0, \tag{A.17}$$

$$L\ddot{\eta}_2 U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta = 0. \tag{A.18}$$

Further substituting for $\ddot{\eta}_2$ from Eq. (A.6), the following equation is obtained:

$$L(WU_2 - \eta_1 U_1^\mathsf{T} K_s U_2 - \eta_2 U_2^\mathsf{T} K_s U_2) M_2^{-1} U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta = 0. \tag{A.19}$$

Rearranging the above equation to collect all the known and unknown terms together, we get:

$$LWU_2 M_2^{-1} U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta = L(\eta_1 U_1^\mathsf{T} K_s U_2 + \eta_2 U_2^\mathsf{T} K_s U_2) M_2^{-1} U_2^\mathsf{T} R. \tag{A.20}$$

Let us define the known left side of the equation as:

$$\mathcal{C} \triangleq LWU_2 M_2^{-1} U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta, \tag{A.21}$$

and write the right hand side of Eq. (A.20) as:

$$L(\eta_1 U_1^\mathsf{T} K_s U_2 + \eta_2 U_2^\mathsf{T} K_s U_2) M_2^{-1} U_2^\mathsf{T} R = LNK_s \underbrace{U_2 M_2^{-1} U_2^\mathsf{T}}_{M_{sn}} R. \tag{A.22}$$

We now take the i^{th} column of the above matrix and substitute for $K_s = C_s^{\mathsf{T}}\hat{\gamma}C_s - C_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\hat{\lambda}C_bC_{nb}$ to obtain:

$$LNK_sM_{sn}Re_i = LNC_s^{\mathsf{T}}\hat{\gamma}C_sM_{sn}Re_i - LNC_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\hat{\lambda}C_bC_{nb}M_{sn}Re_i, \tag{A.23}$$

and using the identity $\hat{x}y = \hat{y}x$ for the right-hand side terms gives:

$$LNK_sM_{sn}Re_i = LNC_s^{\mathsf{T}}\widehat{(C_sM_{sn}Re_i)}\gamma - LNC_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\widehat{(C_bC_{nb}M_{sn}Re_i)}\lambda. \tag{A.24}$$

Substituting for λ in terms of γ from Eq. (A.11) gives:

$$\begin{aligned} LNK_sM_{sn}Re_i = &-LNC_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\widehat{(C_bC_{nb}M_{sn}Re_i)}\tau \\ &+ \Big(LNC_s^{\mathsf{T}}\widehat{(C_sM_{sn}Re_i)} - LNC_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\widehat{(C_bC_{nb}M_{sn}Re_i)}\Lambda\Big)\gamma. \end{aligned} \tag{A.25}$$

Now, substituting it back to the vector equation of Eq. (A.20), and stacking up all these matrices on left and vectors on right, we get:

$$\begin{bmatrix} \Gamma_1^{\mathsf{T}} & \Gamma_2^{\mathsf{T}} & \cdots & \Gamma_{n_r}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \gamma = \begin{bmatrix} \mu_1^{\mathsf{T}} & \mu_2^{\mathsf{T}} & \cdots & \mu_{n_r}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{A.26}$$

where

$$\Gamma_i = LNC_s^{\mathsf{T}}\widehat{(C_sM_{sn}Re_i)} - LNC_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\widehat{(C_bC_{nb}M_{sn}Re_i)}\Lambda, \tag{A.27}$$

$$\mu_i = \mathcal{C}e_i + LNC_{nb}^{\mathsf{T}}C_b^{\mathsf{T}}\widehat{(C_bC_{nb}M_{sn}Re_i)}\tau, \tag{A.28}$$

$$\mathcal{C} = LWM_{sn}R + L\dot{\eta}_2U_2^{\mathsf{T}}R\Psi + [L(\eta_1U_1^{\mathsf{T}} + \eta_2U_2^{\mathsf{T}})R - \bar{Y}]\Theta, \tag{A.29}$$

$$M_{sn} = U_2M_2^{-1}U_2^{\mathsf{T}}, \quad for \quad i = 1, 2 \cdots n_r. \tag{A.30}$$

A.2.2 Controlling the Velocity and Acceleration

For controlling the node positions, we write the error in position as $E_p = L_p N R_p - \bar{Y}_p$ where subscript p is used for the position and write the final linear equation for the force densities in the string in a compact form as (refer Eq. (A.26)):

$$\Gamma_p \gamma = \mu_p, \quad \gamma \geq 0. \tag{A.31}$$

For controlling the velocity, we define the error in velocity of certain nodes as:

$$E_v = L_v \dot{\eta}_2 U_2^{\mathsf{T}} R_v - \bar{\dot{Y}}_v, \quad \dot{E}_v + E_v \Psi_v = 0, \tag{A.32}$$

and use a first order differential equation to derive the error in velocity to zero. Only first derivative of error E_v is required as the control variable, force density γ in the strings come out at the same level of time derivative. Following the same derivation as used in the previous subsections, we write the linear equation to control the nodal velocities as:

$$\Gamma_v \gamma = \mu_v, \quad \gamma \geq 0. \tag{A.33}$$

To control the acceleration of the nodes, the error is defined as:

$$E_a = L_a \ddot{\eta}_2 U_2^{\mathsf{T}} R_a - \bar{\ddot{Y}}_a, \tag{A.34}$$

which can be directly converted to a linear equation in control variable by equating it to zero as $E_a = 0$. Following the same procedure, we get the linear algebra equation to solve for control variable as:

$$\Gamma_a \gamma = \mu_a, \quad \gamma \geq 0. \tag{A.35}$$

Finally, combining the Eqs. (A.31), (A.33), and (A.35) allows to simultaneously control the position, velocity and acceleration of different nodes in the structure.

$$\begin{bmatrix} \Gamma_p^\mathsf{T} & \Gamma_v^\mathsf{T} & \Gamma_a^\mathsf{T} \end{bmatrix}^\mathsf{T} \gamma = \begin{bmatrix} \mu_p^\mathsf{T} & \mu_v^\mathsf{T} & \mu_a^\mathsf{T} \end{bmatrix}^\mathsf{T}, \quad \gamma \geq 0. \tag{A.36}$$

A.3 Estimation of Hessians: Linear Least Squares by Central Difference (LLS-CD)

Using the same Taylor expansion as described in Section 3.4.2, we obtain the following central difference equation:

$$F(\bar{x}_t + \delta x_t, \bar{u}_t + \delta u_t) + F(\bar{x}_t - \delta x_t, \bar{u}_t - \delta u_t) = 2F(\bar{x}_t, \bar{u}_t) + \begin{bmatrix} \delta x_t{}' & \delta u_t{}' \end{bmatrix} \begin{bmatrix} F_{x_t x_t} & F_{x_t u_t} \\ F_{u_t x_t} & F_{u_t u_t} \end{bmatrix} \begin{bmatrix} \delta x_t \\ \delta u_t \end{bmatrix} + O(\|\delta x_t\|^4 + \|\delta u_t\|^4), \tag{A.37}$$

where $F_{x_t x_t} = \left.\dfrac{\partial^2 F}{\partial x^2}\right|_{x_t}$, similar for $F_{u_t x_t}$ and $F_{u_t u_t}$. Denote $z_t = F(\bar{x}_t + \delta x_t, \bar{u}_t + \delta u_t) + F(\bar{x}_t - \delta x_t, \bar{u}_t - \delta u_t) - 2F(\bar{x}_t, \bar{u}_t)$. The Hessian is a $(n_s + n_u)$ by n_s by $(n_s + n_u)$ tensor, where n_s is the number of states and n_u is the number of actuators. Let's seperate the tensor into 2D matrices w.r.t. the second dimension and neglect time t for simplicity of notations:

$$
\begin{aligned}
z_i &= \begin{bmatrix} \delta x' & \delta u' \end{bmatrix} \begin{bmatrix} F_{xx}^{(i)} & F_{xu}^{(i)} \\ F_{ux}^{(i)} & F_{uu}^{(i)} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} \\
&= \sum_{j=1}^{n_s} \sum_{k=1}^{n_s} \frac{\partial^2 F_i}{\partial x_j \partial x_k} \delta x_j \delta x_k + 2 \sum_{j=1}^{n_u} \sum_{k=1}^{n_s} \frac{\partial^2 F_i}{\partial u_j \partial x_k} \delta u_j \delta x_k + \sum_{j=1}^{n_u} \sum_{k=1}^{n_u} \frac{\partial^2 F_i}{\partial u_j \partial u_k} \delta u_j \delta u_k \\
&= \underbrace{\begin{bmatrix} \delta x_1{}^2 & \delta x_1 \delta x_2 & \cdots & \delta x_1 \delta u_{n_u} & \delta x_2{}^2 & \delta x_2 \delta x_3 & \cdots & \delta u_{n_u}^2 \end{bmatrix}}_{\delta M} \underbrace{\begin{bmatrix} \dfrac{\partial^2 F_i}{\partial x_1^2} \\ 2\dfrac{\partial^2 F_i}{\partial x_1 \partial x_2} \\ \vdots \\ 2\dfrac{\partial^2 F_i}{\partial x_1 \partial u_{n_u}} \\ \dfrac{\partial^2 F_i}{\partial x_2^2} \\ 2\dfrac{\partial^2 F_i}{\partial x_2 \partial x_3} \\ \vdots \\ \dfrac{\partial^2 F_i}{\partial u_{n_u}^2} \end{bmatrix}}_{H_i},
\end{aligned}
\tag{A.38}
$$

where $F_{xx}^{(i)} = \left.\dfrac{\partial^2 F_i}{\partial x^2}\right|_{x_t}$, z_i is the i^{th} element of vector z_t and F_i is the i^{th} element of the dynamics vector $F(x_t, u_t)$. Multiplying on both sides by $\delta M'$ and apply standard Least Square method: $H_i = (\delta M^T \delta M)^{-1} \delta M^T z_i$. Then repeat for $i = 1, 2, ..., n_s$ to get the estimation for the Hessian tensor.