Extended version of L4DC 2024 paper: 1–25

Random Features Approximation for Control-Affine Systems

Kimia Kazemian KK983@CORNELL.EDU
Yahya Sattar YSATTAR@CORNELL.EDU
Sarah Dean SDEAN@CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY, USA.

Abstract

Modern data-driven control applications call for flexible nonlinear models that are amenable to principled controller synthesis and realtime feedback. Many nonlinear dynamical systems of interest are *control affine*. We propose two novel classes of nonlinear feature representations which capture control affine structure while allowing for arbitrary complexity in the state dependence. Our methods make use of random features (RF) approximations, inheriting the expressiveness of kernel methods at a lower computational cost. We formalize the representational capabilities of our methods by showing their relationship to the Affine Dot Product (ADP) kernel proposed by Castañeda et al. (2021) and a novel Affine Dense (AD) kernel that we introduce. We further illustrate the utility by presenting a case study of data-driven optimization-based control using control certificate functions (CCF). Simulation experiments on a double pendulum empirically demonstrate the advantages of our methods.

Keywords: Random Features, Control-Affine Systems, Control Certificate Functions.

1. Introduction

Modern control applications require modelling systems with complex and nonlinear dynamics. Modern machine learning techniques offer a data-driven solution. From deep learning to kernel methods, learning-based approaches fit models to data. Highly expressive models can approximate arbitrary functions, and therefore model arbitrarily complex phenomena. However, this comes at a cost—they can be computationally expensive to train and difficult to use for the purpose of synthesizing a controller. This poses a challenge in real-time feedback systems.

Linear regression is a straightforward approach for learning dynamical models from data, so long as a suitable nonlinear feature representation, i.e., set of basis functions, is known (Mania et al., 2020). However, selecting proper basis functions is often challenging and requires modelling detailed properties of the unknown dynamics. One solution is to choose a set of random basis functions to generate feature vectors of fixed dimension. This approach, called *random features (RF)*, can achieve high expressiveness as long as the dimension of the feature vectors is large enough (Rahimi and Recht, 2008). Random features have proven useful for dynamical systems forecasting (Giannakis et al., 2023), receding horizon control (Lale et al., 2021), and policy learning (Lale et al., 2022).

We propose two novel classes of random feature representations suitable for principled data-driven control (Section 3). Our key insight is to leverage the control-affine structure of many nonlinear dynamics of interest, which enables principled optimization-based approaches for controller synthesis. We propose two distinct methods for incorporating this structure into random basis functions and formalize their representation guarantees by showing that they approximate functions in a

Reproducing Kernel Hilbert Space (RKHS). One of our methods approximates the *Affine Dot Product (ADP) kernel* proposed by Castañeda et al. (2021), while the other corresponds to a novel *Affine Dense (AD) kernel* that we propose. The RF methods significantly reduce the computational time and memory complexity compared to their kernel counterparts.

To showcase the utility of an explicit control-affine structure, we present a case study for nonlinear control in Section 4. Our data-driven approach is based on *Control Certificate Functions* (CCFs), which are utilized to synthesize controllers that provably achieve properties such as safety and stability (Taylor et al., 2021). CCFs have been used in a range of applications from robotics to multi-agent systems (Artstein, 1983; Ames et al., 2014; Nguyen et al., 2016; Pickem et al., 2017), including in a data-driven manner (Castañeda et al., 2021; Castañeda et al., 2021; Taylor et al., 2021; Choi et al., 2023). Simulations on a double inverted pendulum illustrate the benefits of our models when used in a *certainty-equivalent* manner. In appendix B.2, we additionally derive uncertainty estimates analogous to those of Gaussian process (GP) regression, and use them to propose a robust data-driven controller in appendix D. We highlight that the approximation methods that we propose may be broadly of interest for any control application which makes use of GPs (Koller et al., 2018; Caldwell and Marshall, 2021; Bradford et al., 2019; Hewing et al., 2020; Li et al., 2021).

2. Problem Setting and Preliminaries

In this work, we consider an affine modelling and prediction problem inspired by applications in data-driven control. We first define the general problem of interest, and then give several examples that arise in the context of learning for dynamics and control.

Definition 1 (Control-affine modelling problem) *For data of the form $\{(\boldsymbol{x}_i, \boldsymbol{u}_i, z_i)\}_{i=1}^N$, find a function $\hat{h} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ which i) is affine in its second argument and ii) accurately models the relationship between $(\boldsymbol{x}, \boldsymbol{u})$ and z, i.e. $\hat{h}(\boldsymbol{x}_i, \boldsymbol{u}_i)$ is not far from z_i.*

Such a modelling problem naturally arises in applications involving nonlinear control-affine systems. The dynamics are described in either continuous or discrete time:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} \quad \text{or} \quad \boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t) + \boldsymbol{g}(\boldsymbol{x}_t)\boldsymbol{u}_t. \tag{1}$$

Here, $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ is the system state, and $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. The nonlinear function $\boldsymbol{f}$ determines the evolution of the state in the absence of control inputs, while $\boldsymbol{g}$ models state-dependent actuation. Control-affine dynamics arise naturally from manipulator equations (Murray and Hauser, 1991; Tedrake, 2023), and are thus prevalent in applications like robotics. While many systems are known to follow dynamics of the form (1), the precise form of $\boldsymbol{f}$ and/or $\boldsymbol{g}$ may be unknown. Data-driven approaches enable the control of systems with entirely or partially unknown dynamics. There are many examples of modelling tasks that arise in such data-driven control settings.

Example 1 *Consider a model predictive control setting in which the evolution of the state itself must be predicted (Lale et al., 2021). For a discrete-time control-affine system (1) with unknown dynamics and direct state observation, a sequence of states and inputs $\{(\boldsymbol{x}_i, \boldsymbol{u}_i)\}_{i=0}^N$ defines n modelling problems of the form presented in Definition 1: one for each state dimension.*

Example 2 *Consider again model predictive control, now for continuous time control-affine dynamics (1). A sequence of sampled states $\{\boldsymbol{x}_i\}_{i=0}^N$ can be used to approximate $\{\dot{\boldsymbol{x}}_i\}_{i=1}^N$ with forward finite differencing and define n modelling problems of the form presented in Definition 1.*

Example 3 *Consider Certificate Function Control (Taylor et al., 2021), which enforces safety or stability using a known certificate function $C : \mathcal{X} \to \mathbb{R}$. For continuous time control-affine dynamics (1), such controllers require computing $\dot{C} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, which cannot be done directly when the dynamics are unknown. However, a sequence $\{C(\boldsymbol{x}_i)\}_{i=0}^N$ can be computed from a sequence of sampled states and the known function C. Then, finite differencing approximates the time derivative, resulting in a problem of the form presented in Definition 1.*

Example 4 *For any of the previous examples, suppose that an approximate model of the dynamics $\tilde{\boldsymbol{f}}$ and $\tilde{\boldsymbol{g}}$ is known. Then learning residual error dynamics also results in a problem of the form presented in Definition 1 (see e.g., Taylor et al. (2021); Castañeda et al. (2021)).*

The examples above serve to motivate the relevance of the modelling problem in Definition 1. We now turn to background and preliminaries on solving it. Our focus is on nonparametric techniques which can model phenomena of arbitrary complexity. We review kernel regression, which is both nonparametric and amenable to uncertainty quantification, and random features approximation, which allows for computational efficiency.

2.1. From Linear to Kernel Regression

We begin by reviewing regression approaches for general data containing input vectors $\{\boldsymbol{s}_i\}_{i=1}^N \subset \mathbb{R}^d$ and a target output variable $\{z_i\}_{i=1}^N \subset \mathbb{R}$. Our starting point is parametric linear regression, in which predictions depend linearly on a known nonlinear feature function of the inputs. Let $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^D$ map an input vector $\boldsymbol{s} \in \mathbb{R}^d$ to a feature vector $\boldsymbol{\phi}(\boldsymbol{s}) \in \mathbb{R}^D$. The feature function, also known as a basis function, maps the input vectors to a higher-dimensional feature space, where a linear relationship can be established more easily.

Linear least-squares regression (Watson, 1967) models the relationship as $\hat{h}(\boldsymbol{s}) = \hat{\boldsymbol{w}}^\top \boldsymbol{\phi}(\boldsymbol{s})$, where the parameter $\hat{\boldsymbol{w}} \in \mathbb{R}^D$ is learned from data by solving

$$\min_{\boldsymbol{w} \in \mathbb{R}^D} \sum_{i=1}^N (\boldsymbol{\phi}(\boldsymbol{s}_i)^\top \boldsymbol{w} - z_i)^2 + \lambda \|\boldsymbol{w}\|_2^2, \tag{2}$$

where $\lambda \geq 0$ is a regularization parameter. Let the matrix $\Phi \in \mathbb{R}^{N \times D}$ and the vector $\boldsymbol{z} \in \mathbb{R}^N$ be the aggregation of rows $\{\boldsymbol{\phi}(\boldsymbol{s}_i)^\top\}_{i=1}^N$ and $\{z_i\}_{i=1}^N$, respectively. Then the prediction is

$$\hat{h}(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^\top (\Phi^\top \Phi + \lambda \boldsymbol{I}_D)^{-1} \Phi^\top \boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{s})^\top \Phi^\top (\Phi \Phi^\top + \lambda \boldsymbol{I}_N)^{-1} \boldsymbol{z}. \tag{3}$$

The first equality is the closed-form solution to the least squares objective, and the second leverages the *kernel trick* (Scholkopf and Smola, 2018; Müller et al., 2018). The significance of this reformulation is that the vector $\Phi \boldsymbol{\phi}(\boldsymbol{s}) =: \boldsymbol{k}(\boldsymbol{s})$ and matrix $\Phi \Phi^\top =: K$ can be computed using only inner products of basis functions evaluated on training data. This is attractive because the class of basis functions determine the complexity and richness of the modelled relationship between inputs $\boldsymbol{s}$ and outputs z. While low-dimensional bases may suffice for highly structured processes, generally, a suitable compact basis may not be known a priori.

Kernel methods allow for expressive basis functions of arbitrarily high or infinite dimension. A kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ generalizes inner products between basis functions, and is used as a nonparametric approach for representing complex functions. Appropriately defined, kernel

ridge regression corresponds to regression in Reproducing Kernel Hilbert Spaces (RKHS) (Wendland, 2004). Many RKHS are dense in the set of continuous functions, enabling arbitrarily accurate representation of continuous functions via kernel regression. The following lemma presents a sufficient condition for checking that a kernel function defines a RKHS. It is a direct implication of the Moore–Aronszajn theorem and Lemma 1 in Berlinet and Thomas-Agnan (2011).

Lemma 2 *Let $\mathcal{H}$ be some Hilbert space with inner product $\langle \cdot, \cdot \rangle$. A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a reproducing kernel if there exists a mapping $\varphi : \mathbb{R}^d \to \mathcal{H}$ such that $k(\boldsymbol{s}, \boldsymbol{s}') = \langle \varphi(\boldsymbol{s}), \varphi(\boldsymbol{s}') \rangle$.*

In addition to their expressivity, kernel methods are amenable to theoretical guarantees and uncertainty characterization. Popularized from the Bayesian perspective as Gaussian Process (GP) regression (Williams and Rasmussen, 2006), confidence intervals on kernel predictions can be derived even in frequentist settings (Srinivas et al., 2009). We discuss this perspective further in appendix B.1 as it is useful for robust control. The drawback of kernel methods is computation. Algorithms have superlinear complexity in the number of data points. In particular, computing the kernel weights can be prohibitively expensive for large datasets. Solving (3) generally requires $O(N^3)$ time and $O(N^2)$ memory.

2.2. Random Feature Approximation

Rather than using kernel methods directly, we propose basis functions which are expressive, general purpose, and yet finite-dimensional. Consider a parametric family of basis functions $b : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}$. Then for parameters $\{\boldsymbol{\vartheta}_j\}_{j=1}^D \subset \mathbb{R}^p$ sampled i.i.d. from a fixed probability distribution $p(\boldsymbol{\vartheta})$, the random basis is defined as $\boldsymbol{\phi}(\boldsymbol{s}) = \begin{bmatrix} b(\boldsymbol{s}; \boldsymbol{\vartheta}_1) & b(\boldsymbol{s}; \boldsymbol{\vartheta}_2) & \dots & b(\boldsymbol{s}; \boldsymbol{\vartheta}_D) \end{bmatrix}^\top$. Random basis functions of this form approximate rich class of functions in the sense that $\boldsymbol{\phi}(\boldsymbol{s})^\top \boldsymbol{\phi}(\boldsymbol{s}')$ is a Monte-Carlo estimator which converges uniformly to a kernel $k(\boldsymbol{s}, \boldsymbol{s}')$ (Rahimi and Recht, 2008). The rate of convergence is controlled by the feature dimension D and the particular kernel depends on the definition of $b(\cdot; \boldsymbol{\vartheta})$ and $p(\boldsymbol{\vartheta})$.

The underlying observation behind random features is a simple consequence of Bochner's Theorem (Avron et al., 2017): For every normalized shift-invariant kernel (i.e., $k(0) = 1$), there is a probability density function $p(\cdot)$ on $\mathbb{R}^d$ such that

$$k(\boldsymbol{s}, \boldsymbol{s}') = \int_{\boldsymbol{\vartheta} \in \mathbb{R}^d} e^{-i 2\pi \boldsymbol{\vartheta}^\top (\boldsymbol{s} - \boldsymbol{s}')} p(\boldsymbol{\vartheta}) d\boldsymbol{\vartheta} =: \mathcal{F}(p(\boldsymbol{\vartheta})). \tag{4}$$

In other words, the inverse Fourier transform $\mathcal{F}^{-1}$ of the kernel $k(\cdot)$ is the probability density function $p(\cdot)$. This implies a one-to-one correspondence between any shift invariant kernel and a random features basis.

Example 5 (Random Fourier basis) *The random Fourier basis consists of sinusoidal nonlinearities of the form $b(\boldsymbol{s}; \boldsymbol{\vartheta}) = \begin{bmatrix} \cos(\boldsymbol{\vartheta}^\top \boldsymbol{s}) & \sin(\boldsymbol{\vartheta}^\top \boldsymbol{s}) \end{bmatrix}$. When $\boldsymbol{\vartheta}$ is sampled from a Gaussian distribution, i.e., $p(\boldsymbol{\vartheta}) \sim \mathcal{N}(0, 2\gamma \boldsymbol{I}_d)$, then the random Fourier basis approximates the radial basis function (RBF) kernel $k(\boldsymbol{s}, \boldsymbol{s}') = e^{-\frac{1}{\gamma} \|\boldsymbol{s} - \boldsymbol{s}'\|_2^2}$.*

The randomized nonlinear expansions provide a compact and computationally efficient alternative to the RKHS representations. This is particularly attractive when the number of data points is large. Recall from the prior section the feature matrix $\Phi \in \mathbb{R}^{N \times D}$ appearing in the prediction (3).

Since $\boldsymbol{\phi}(\boldsymbol{s}) \in \mathbb{R}^D$ and $\Phi^\top \Phi$ is a $D \times D$ matrix, the computation only depends on the dimension of our feature space. Hence, we can compute a random feature approximation in $O(ND^2)$ time and $O(ND)$ memory, which is computationally attractive when $D < N$.

3. Random Features for Control-Affine Modelling

In this section, we use ideas from kernel regression and random feature approximations to propose representations which capture the control-affine structure from Definition 1. We first present two general approaches for defining basis functions that are affine in the control variable. Then, we present RKHS representation guarantees by showing that the random basis approximates particular kernels. Finally, we present experiments which illustrate the predictive modelling capabilities of the proposed methods.

3.1. Control-Affine Basis Functions

The control-affine modelling problem (Definition 1) allows for complex dependence on the state variable, but imposes a restriction on the control variable. Given any arbitrary state-dependent bases $\boldsymbol{\psi}_i : \mathcal{X} \to \mathbb{R}^D$ for $i = 1, ..., m+1$, we propose the following two basis functions that are affine in the control variable $\boldsymbol{u}$.

Definition 3 (Affine dot product (ADP) bases) *The basis $\boldsymbol{\phi}_c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{D(m+1)}$, given by*

$$\boldsymbol{\phi}_c(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} u_1 \boldsymbol{\psi}_1(\boldsymbol{x})^\top & \dots & u_m \boldsymbol{\psi}_m(\boldsymbol{x})^\top & \boldsymbol{\psi}_{m+1}(\boldsymbol{x})^\top \end{bmatrix}^\top,$$

is the ADP basis of $m+1$ individual basis functions $\boldsymbol{\psi}_i : \mathcal{X} \to \mathbb{R}^D$, $i = 1, \dots, m+1$.

As we show in the following section (see Theorem 6), the ADP bases approximate the affine dot product (ADP) kernel, which was first proposed by Castañeda et al. (2021). Note that the ADP bases can also be written as the product of $\text{blkdiag}(\boldsymbol{\psi}_1(\boldsymbol{x}), ..., \boldsymbol{\psi}_{m+1}(\boldsymbol{x}))$ with the vector $[\boldsymbol{u}^\top \ 1]^\top$. This basis expands the feature dimension for every dimension of the control input, resulting in dimension which scales by $m+1$. This observation motivates a second proposed representation.

Definition 4 (Affine dense (AD) bases) *The basis $\boldsymbol{\phi}_d : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^D$, given by*

$$\boldsymbol{\phi}_d(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \boldsymbol{\psi}_1(\boldsymbol{x}) \dots \boldsymbol{\psi}_{m+1}(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix}$$

is the AD basis of $m+1$ individual basis functions $\boldsymbol{\psi}_i : \mathcal{X} \to \mathbb{R}^D$, $i = 1, \dots m+1$.

Compared with the ADP basis, the AD basis is more compact. For individual basis functions of dimension D, the AD basis will be of dimension D, whereas the ADP basis will be of dimension $D(m+1)$. Considering the linear regression use case, this means that AD has $O(ND^2)$ time and $O(ND)$ memory complexity, whereas for ADP it is $O(N(m+1)^2 D^2)$ and $O(N(m+1)D)$.

Leveraging ideas from random Fourier features, we propose control-affine basis functions constructed with state-dependent random Fourier basis functions $\boldsymbol{\psi}_i$ for $i = 1, \dots, m+1$:

$$\boldsymbol{\psi}_i(\boldsymbol{x}) := \sqrt{2/D} \begin{bmatrix} \sin(\boldsymbol{\vartheta}_{i,1}^\top \boldsymbol{x}) & \cos(\boldsymbol{\vartheta}_{i,1}^\top \boldsymbol{x}) & \dots & \sin(\boldsymbol{\vartheta}_{i,D/2}^\top \boldsymbol{x}) & \cos(\boldsymbol{\vartheta}_{i,D/2}^\top \boldsymbol{x}) \end{bmatrix}^\top, \quad (5)$$

where weights $\{\boldsymbol{\vartheta}_{i,j}\}_{j=1}^{D/2}$ are drawn i.i.d. from the distribution $p_i(\boldsymbol{\vartheta})$ for $i = 1, \ldots, m+1$. As described in Example 5, each of these individual basis functions approximates a shift invariant kernel corresponding to the Fourier transform of the density $p_i(\boldsymbol{\vartheta})$ (Rahimi and Recht, 2008, 2007). In other words, $\mathbb{E}_{\boldsymbol{\vartheta}\sim p_i(\cdot)}[\boldsymbol{\psi}_i(\boldsymbol{x})^\top \boldsymbol{\psi}_i(\boldsymbol{x}')] = k_i(\boldsymbol{x} - \boldsymbol{x}')$ where $k_i(\boldsymbol{v}) = \mathcal{F}(p_i(\boldsymbol{\vartheta}))[\boldsymbol{v}]$.

3.2. Representation Guarantees

We now develop representation guarantees for the compound bases by showing which kernels they approximate. The affine dot product (ADP) kernel was first proposed by Castañeda et al. (2021) for systems with control-affine dynamics.

Definition 5 (Affine dot product (ADP) kernel) *Define $k_c : \mathcal{X} \times U \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, given by*

$$k_c((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}')) := \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix} \operatorname{diag}(k_1(\boldsymbol{x}, \boldsymbol{x}'), \cdots, k_{m+1}(\boldsymbol{x}, \boldsymbol{x}')) \begin{bmatrix}\boldsymbol{u}'^\top & 1\end{bmatrix}^\top,$$

as the Affine Dot Product (ADP) compound kernel of $m+1$ individual kernels $k_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

The following theorem shows that the ADP basis approximates the ADP kernel.

Theorem 6 (ADP Approximation) *For $i = 1, ..., m+1$, suppose the basis functions $\boldsymbol{\psi}_i$ are defined according to (5) with p_i the inverse Fourier transform of a shift invariant kernel k_i. Let $\boldsymbol{\phi}_c$ be the ADP compound basis of $\boldsymbol{\psi}_i$ and let k_c the compound ADP kernel of k_i. Then*

$$\mathbb{E}[\boldsymbol{\phi}_c(\boldsymbol{x}, \boldsymbol{u})^\top \boldsymbol{\phi}_c(\boldsymbol{x}', \boldsymbol{u}')] = k_c((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}')).$$

The result follows by relating the dot product of features to the diagonal matrix in the ADP kernel. We defer all formal proofs to appendix A.

An alternative way to understand the ADP random feature approximation is to interpret the $(m+1) \times (m+1)$ diagonal matrix of kernels as an operator valued kernel. This operator valued kernel is the sum of $m+1$ *decomposable kernels*, as defined by Brault et al. (2016) (Definition 3). The ADP block diagonal matrix of basis functions can be interpreted through their framework as a random feature approximation for this operator valued kernel.

We now turn to the affine dense basis. First, we define a novel Affine Dense compound kernel.

Definition 7 (Affine dense (AD) kernel) *For $m+1$ individual kernels $k_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, let $\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{x}')$ be the diagonal matrix with i^{th} entry as $k_i(\boldsymbol{x} - \boldsymbol{x}')$, and $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{x}')$ a matrix with zero on the diagonal and $[\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{x}')]_{ij} = k_i(\boldsymbol{x}) k_j(\boldsymbol{x}')$ for $i \neq j \in [m+1]$. Then, define the Affine Dense (AD) compound kernel as $k_d : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \times U \to \mathbb{R}$, given by*

$$k_d((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}')) := \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix} \left(\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{A}(\boldsymbol{x}, \boldsymbol{x}')\right) \begin{bmatrix}\boldsymbol{u}'^\top & 1\end{bmatrix}^\top.$$

Notice that the diagonal matrix $\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{x}')$ in the AD kernel is similar to the ADP kernel. However, the AD kernel additionally includes the dense matrix $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{x}')$. Due to this second dense term, the AD compound kernel is not shift invariant in $\boldsymbol{x}$. As a result, it is not possible to view $\boldsymbol{A} + \boldsymbol{D}$ as a shift invariant operator-valued kernel, and thus the results of Brault et al. (2016) cannot be used to derive a random features approximation. Furthermore, it is not immediately clear whether the AD kernel is indeed a valid reproducing kernel. We therefore begin by showing that it is.

Theorem 8 (AD kernel) *Let $k_d((\boldsymbol{x},\boldsymbol{u}),(\boldsymbol{x}',\boldsymbol{u}'))$ be as in Definition 7. Suppose each $k_i(\boldsymbol{x},\boldsymbol{x}')$ is a normalized shift invariant reproducing kernel. Then, $k_d((\boldsymbol{x},\boldsymbol{u}),(\boldsymbol{x}',\boldsymbol{u}'))$ is a reproducing kernel.*

To prove this result, we use the crucial (but non-obvious) claim that if $k(\boldsymbol{x},\boldsymbol{x}')$ is a normalized shift invariant reproducing kernel, then $k(\boldsymbol{x},\boldsymbol{x}')-k(\boldsymbol{x})k(\boldsymbol{x}')$ is also a reproducing kernel. To prove that the claim is true, we construct an explicit feature mapping of the form required in Lemma 2. With the claim in hand, the proof follows by algebraic manipulations and the fact that the set of reproducing kernels is closed under addition. Therefore, the AD kernel k_d is a reproducing kernel and thus defines a RKHS. We next show that the AD basis functions approximate this RKHS.

Theorem 9 (Affine-dense kernel approximation) *Suppose that for $i=1,\ldots,m{+}1$ the basis functions ψ_i are defined according to (5) with p_i the inverse Fourier transform of a shift invariant kernel k_i. Let $\boldsymbol{\phi}_d$ be the AD compound basis of $\boldsymbol{\psi}_i$ and let k_d be the compound AD kernel of k_i. Then $\mathbb{E}[\boldsymbol{\phi}_d(\boldsymbol{x},\boldsymbol{u})^\top\boldsymbol{\phi}_d(\boldsymbol{x}',\boldsymbol{u}')]=k_d((\boldsymbol{x},\boldsymbol{u}),(\boldsymbol{x}',\boldsymbol{u}'))$.*

So far our results show that the basis functions we propose approximate kernel regression in expectation. When the dimension D is large enough, the approximation error can be bounded with high probability (Rahimi and Recht, 2007; Sutherland and Schneider, 2015). In particular, Sutherland and Schneider (2015) show conditions under which the pointwise approximation error is no more than ϵ with probability depending on D and ϵ. We therefore conclude with a result which shows that when the individual kernels have bounded approximation errors, so do the compound kernels. In Appendix B.3, we further derive bounds on the prediction errors and confidence intervals for use in robust control.

Proposition 10 (Kernel Approximation Errors) *Consider the ADP kernel $k_c(\boldsymbol{s},\boldsymbol{s}')$ and the AD kernel $k_d(\boldsymbol{s},\boldsymbol{s}')$ from Definitions 5 and 7, respectively. Consider $\{k_i(\boldsymbol{x})\}_{i=1}^{m+1}$ which are the individual kernels used to construct the ADP and the AD kernels. Recall $\boldsymbol{\psi}$ from 5. Suppose $|k_i(\boldsymbol{x})|\leq 1$ and $|k_i(\boldsymbol{x})-\boldsymbol{\psi}_i(\boldsymbol{x})^\top\boldsymbol{\psi}_i(\boldsymbol{x})|\leq\epsilon$ for all $\boldsymbol{x}\in\mathcal{X}$ and $i\in[m+1]$. Then, we have*

$$\max\{|k_c(\boldsymbol{s},\boldsymbol{s}')-\boldsymbol{\phi}_c(\boldsymbol{s})^\top\boldsymbol{\phi}_c(\boldsymbol{s}')|,\ |k_d(\boldsymbol{s},\boldsymbol{s}')-\boldsymbol{\phi}_d(\boldsymbol{s})^\top\boldsymbol{\phi}_d(\boldsymbol{s}')|\}\leq\epsilon(\boldsymbol{u}^\top\boldsymbol{u}'+1) \tag{6}$$

where $\boldsymbol{\phi}_c$ and $\boldsymbol{\phi}_d$ are defined in Theorems 6 and 9 respectively.

3.3. Numerical demonstration

In this section, we empirically[1] study the performance of the the two random features methods (ADP-RF and AD-RF) as well as the corresponding kernel methods (ADP-K and AD-K). We focus on performance in terms of prediction accuracy. In the next section, we also demonstrate the utility of these models for data-driven control.

We consider a prediction task relating to a double pendulum with actuation at both joints. The state of this system $\boldsymbol{x}\in\mathbb{R}^4$ consists of two angle variables and two angular velocities, while the control input $\boldsymbol{u}\in\mathbb{R}^2$ consists of the two joint actuation torques. In appendix E.1, we present a full derivation of the dynamics equation, which is affine in the control inputs. We simulate the system under closed-loop control and sample at 10 Hz. The controller is imperfectly designed to bring the system to an upright and balanced configuration; Further controller details are deferred to

1. Code is available at https:github.com/kimzemian/swift_affine_mastery

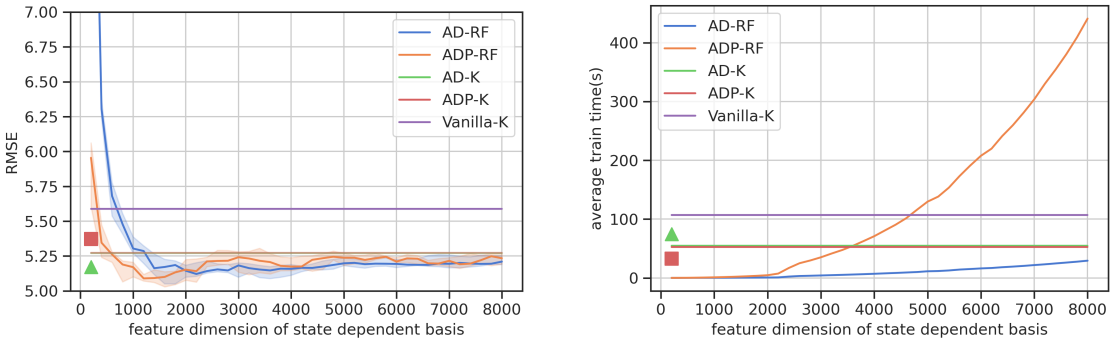

Figure 1: Evaluation of models, comparing prediction accuracy on test data (left) and training time on 8859 points (right) for a prediction problem on a double pendulum system. Horizontal lines and markers correspond to kernel methods. Random features are sampled 10 times at varying dimensions; the left panel displays median and quartiles over the trials while the right panel shows the mean. Increasing the feature dimension of each state-dependent basis $\boldsymbol{\psi}_i(x)$ results in lower RMSE but longer training time, especially for ADP-RF.

the following section. We collect a dataset containing 226 trajectories, each starting at a different initial point and lasting 5 seconds. The dataset if of the form $\{\{(\boldsymbol{x}_i^e, \boldsymbol{u}_i^e), z_i^e\}_{i=1}^L\}_{e=1}^E$ where z_i is the time derivative of a scalar function of the state (Example 3); details are described in the following section. We split the data into train and evaluation subsets with an $80/20$ split, so the train size is 8859 and test size is 2215, formulating a prediction task of the form in Definition 1.

We compare the performance of five models: three kernel methods (Vanilla-K, ADP-K, AD-K) and two random features methods (ADP-RF, AD-RF). Vanilla-K is an RBF kernel (Example 5) that operates on the concatenated state and input without any affine structure. ADP-K and AD-K (Definitions 5 and 7) use RBF kernel on the state variable, whereas, ADP-RF and AD-RF are as defined in Theorems 6 and 9 with the corresponding random Fourier bases (5) as in Example 5. For all models, $\gamma = 1$ and $\lambda = 1$. Figure 1 plots the performance in terms of test accuracy and training time. The left panel shows median and quartile RMSE on the evaluation split and the right panel shows the average training time. The kernels are represented by horizontal lines and markers. Vanilla-K performs worse than the affine kernels since it does not capture the affine structure, while AD-K and ADP-K have similar performance. For the RF models, we examine the effect of the random features approximation of state-dependent samples $\boldsymbol{\psi}_i(x)$ of dimension D. ADP-RF has lower error for smaller feature dimension, but both RF methods quickly approach the performance of the kernel methods. For small D, the RF methods are both much faster. Train time increases with D more quickly for ADP-RF than AD-RF, as training ADP-RF scales quadratically with $m+1$. Comparing the RF models, Figure 1 suggests that, although training with AD-RF is faster as compared to ADP-RF, the later has smaller RMSE. We attribute this to the higher dimensionality of the ADP compound basis, which allows for greater expressivity. In Appendix C, we present extensive experiments with synthetic data demonstrating the relationship between training time and RMSE. These show that for fixed training time, AD-RF outperforms ADP-RF on accuracy when D is sufficiently large, and this performance advantage grows as input dimension m increases.

4. Case Study: Certificate Function Control

A key motivation for our work is that the affine structure of our data-driven models is amenable for use in control tasks. We therefore describe how to incorporate these models into a particular approach to nonlinear control. We then evaluate closed loop performance of our models.

Background As a case study, we demonstrate a nonlinear control technique based on *control certificate functions* as proposed by Taylor et al. (2021), which we refer to for a more rigorous and precise introduction. This approach generalizes and unifies the use of control Lyapunov functions (CLFs) to guarantee stability (Galloway et al., 2015) and control barrier functions (CBFs) to guarantee safety (Ames et al., 2016). Certificate function control requires a continuously differentiable *certificate function* $C : \mathbb{R}^n \to \mathbb{R}$ satisfying certain properties (Taylor et al., 2021), along with a *comparison function* $\alpha : \mathbb{R} \to \mathbb{R}_+$. Then, an optimization-based state feedback controller can be defined which will guarantee desired properties such as stability or safety by construction (Ames et al., 2019). Given some "desired" control input $\boldsymbol{u}_d(\boldsymbol{x})$, the CCF quadratic program (QP) is:

$$\boldsymbol{u}^*(\boldsymbol{x}) = \underset{\boldsymbol{u}\in\mathbb{R}^m}{\arg\min} \|\boldsymbol{u}\|_2^2 + c_1 \|\boldsymbol{u} - \boldsymbol{u}_d(\boldsymbol{x})\|_2^2 \qquad \textbf{(CCF-QP)}$$
$$\text{s.t. } \underbrace{\nabla C(\boldsymbol{x})^\top (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u})}_{\dot{C}(\boldsymbol{x},\boldsymbol{u})} + \alpha(C(\boldsymbol{x})) \leq 0.$$

Data-driven Control We now suppose that the dynamics $\boldsymbol{f}$ and $\boldsymbol{g}$ are unknown, so the CCF-QP controller cannot be directly implemented. We assume that a valid CCF C and comparison function α for the unknown true system is given[2]. Given a sampled trajectory $\{(\boldsymbol{x}_i, \boldsymbol{u}_i)\}_{i=1}^N$, we construct a control affine modelling problem for $\dot{C} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ as described in Example 3. We therefore use methods discussed in Section 3 to create a control-affine model $\hat{h}(\boldsymbol{x}, \boldsymbol{u})$ which can be used in place of the unknown $\dot{C}(\boldsymbol{x}, \boldsymbol{u})$ function in (**CCF-QP**) in a *certainty equivalent* (CE) manner. Because the model $\hat{h}$ is affine in $\boldsymbol{u}$, the resulting optimization problem is still a QP. In Appendix E.2, we provide additional details on constructing this QP for both kernel and RF methods. We also discuss methods for *robust*, rather than CE, data-driven control. The robust approach requires estimates of uncertainty (e.g. as in Gaussian process regression) as well as the pointwise RF approximation errors, and results in a second order cone program (SOCP), see Appendix D.

Simulation experiments We simulate data-driven CCF control of the double pendulum introduced in Section 3.3, where the goal is to swing up and balance in the upright position $\boldsymbol{x} = 0$ with only an incorrect model of the dynamics $\tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}$. Knowing only its degree of actuation, we may conclude that the dynamics are *feedback linearizable* and therefore we can define a Control Lyapunov Function (CLF) without the exact dynamics model (Taylor et al., 2019). Specifically, we define $C(\boldsymbol{x}) = \boldsymbol{x}^\top P \boldsymbol{x}$, $\alpha(\boldsymbol{x}) = .725\boldsymbol{x}$, $\boldsymbol{u}_d(\boldsymbol{x})$ a feedback linearizing controller for the incorrect $\tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}$, and $c_1 = 25$. Full details are provided in Appendices E.3, E.4.

We first define a "nominal" QP controller which selects inputs according to (**CCF-QP**) with the nominal model $\tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}$. We use this controller to gather trajectories and define a dataset as described in Section 3.3. We subsample the data by $1/5$ and derive data-driven models of $\dot{C}(\boldsymbol{x}, \boldsymbol{u})$ as outlined in the paragraph above. We consider four data-driven QP controllers using the four affine models: AD-K, ADP-K, AD-RF, and ADP-RF. For the data-driven controllers, we augment the initial dataset

2. This assumption is met for feedback linearizable systems as long as the the degree of actuation of the true dynamics model is known (Taylor et al., 2019). For example, many robotic systems satisfy this assumption.

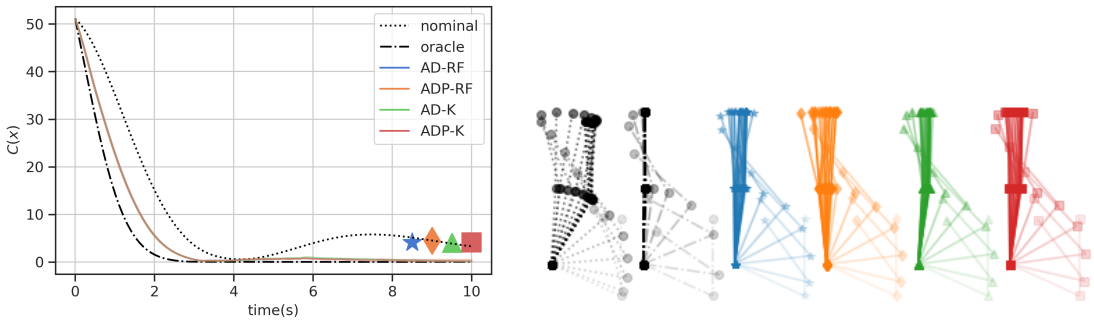

Figure 2: Left: The value of the Lyapunov function $C(\boldsymbol{x})$ over time for nominal, oracle, and data-driven controllers with initial state $[2, 0, 0, 0]$. Right: Illustration of the pendulum configurations over time. Nominal fails to balance the pendulum; data-driven methods succeed.

with episodic data collection: we run the controller for 10 seconds at 10 Hz, retrain, and repeat for ten episodes. The RF dimension is $D = N/5$ for N the size of the training data. Finally, we compare the performance of the the nominal and data-driven methods with an "oracle" controller that solves (**CCF-QP**) with the true dynamics. Figure 2 plots the system trajectory in terms of the Lyapunov function $C(\boldsymbol{x})$ and the pendulum configuration. While the nominal controller fails to balance the pendulum, the data-driven controllers succeed and are similar to each other.

5. Conclusion

This work considers a control affine modelling problem and proposes two classes of random basis functions as a solution: ADP and AD. The representation guarantees of these methods are made formal by connection to kernel regression in corresponding RKHSs. A case study in nonlinear control with CCF illustrates the utility of control affine models. Numerical experiments demonstrate the performance of the RF and kernel methods in terms of accuracy, computation time, and closed loop control performance. In Appendix B.3, we additionally present uncertainty estimates analogous to Gaussian process (GP) regression, as well as a corresponding robust data-driven control. We highlight that the approximation methods that we propose may be broadly of interest for any control application which makes use of GPs.

Our work opens the door to many future questions of interest. It would be interesting to develop kernels and random features tailored to particular control applications. One could explore the application of our methods to additional control techniques, like feedback linearization or model predictive control. It would be interesting to develop principled techniques for acquiring data, expanding from the simple warm start episodic approach that we used. Furthermore, additional methods for approximating kernels would provide alternatives to speeding up kernel and GP regression for data-driven control.

Acknowledgments

We thank Jason Choi for helping us understand their code base for ADP kernel. This work was partly funded by NSF CCF 2312774 and NSF OAC-2311521, a LinkedIn Research Award, and a gift from Wayfair.

References

Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2016.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

Zvi Artstein. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications*, 7(11):1163–1173, 1983.

Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *International conference on machine learning*, pages 253–262. PMLR, 2017.

Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

Eric Bradford, Lars Imsland, and Ehecatl Antonio del Rio-Chanona. Nonlinear model predictive control with explicit back-offs for gaussian process state space models. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 4747–4754, 2019. doi: 10.1109/CDC40024.2019.9029443.

Romain Brault, Markus Heinonen, and Florence Buc. Random fourier features for operator-valued kernels. In *Asian Conference on Machine Learning*, pages 110–125. PMLR, 2016.

Jack Caldwell and Joshua A. Marshall. Towards efficient learning-based model predictive control via feedback linearization and gaussian process regression. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4306–4311, 2021. doi: 10.1109/IROS51168.2021.9636755.

Fernando Castañeda, Jason J Choi, Bike Zhang, Claire J Tomlin, and Koushil Sreenath. Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics. In *2021 American Control Conference (ACC)*, pages 3683–3690. IEEE, 2021.

Fernando Castañeda, Jason J. Choi, Bike Zhang, Claire J. Tomlin, and Koushil Sreenath. Pointwise feasibility of gaussian process-based safety-critical control under model uncertainty. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6762–6769, 2021. doi: 10.1109/CDC45484.2021.9683743.

Jason J Choi, Fernando Castañeda, Wonsuhk Jung, Bike Zhang, Claire J Tomlin, and Koushil Sreenath. Constraint-guided online data selection for scalable data-driven safety filters in uncertain robotic systems. *arXiv preprint arXiv:2311.13824*, 2023.

Kevin Galloway, Koushil Sreenath, Aaron D Ames, and Jessy W Grizzle. Torque saturation in bipedal robotic walking through control lyapunov function-based quadratic programs. *IEEE Access*, 3:323–332, 2015.

Dimitrios Giannakis, Amelia Henriksen, Joel A Tropp, and Rachel Ward. Learning to forecast dynamical systems from streaming data. *SIAM Journal on Applied Dynamical Systems*, 22(2): 527–558, 2023.

Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2020. doi: 10.1109/TCST.2019.2949757.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6059–6066, 2018. doi: 10.1109/CDC.2018.8619572.

Sahin Lale, Kamyar Azizzadenesheli, Babak Hassibi, and Anima Anandkumar. Model learning predictive control in nonlinear dynamical systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 757–762. IEEE, 2021.

Sahin Lale, Yuanyuan Shi, Guannan Qu, Kamyar Azizzadenesheli, Adam Wierman, and Anima Anandkumar. Kcrl: Krasovskii-constrained reinforcement learning with guaranteed stability in nonlinear dynamical systems. *arXiv preprint arXiv:2206.01704*, 2022.

Fei Li, Huiping Li, and Yuyao He. Adaptive stochastic model predictive control of linear systems using gaussian process regression. *IET Control Theory & Applications*, 15(5):683–693, 2021.

Horia Mania, Michael I Jordan, and Benjamin Recht. Active learning for nonlinear system identification with guarantees. *arXiv preprint arXiv:2006.10277*, 2020.

Klaus-Robert Müller, Sebastian Mika, Koji Tsuda, and Koji Schölkopf. An introduction to kernel-based learning algorithms. In *Handbook of neural network signal processing*, pages 4–1. CRC Press, 2018.

Richard M Murray and John Edmond Hauser. *A case study in approximate linearization: The acrobat example*. Electronics Research Laboratory, College of Engineering, University of ..., 1991.

Quan Nguyen, Ayonga Hereid, Jessy W Grizzle, Aaron D Ames, and Koushil Sreenath. 3d dynamic walking on stepping stones with control barrier functions. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 827–834. IEEE, 2016.

Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. IEEE, 2017.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1177–1184, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th annual allerton conference on communication, control, and computing*, pages 555–561. IEEE, 2008.

Martin Schechter. *Principles of functional analysis*. Number 36. American Mathematical Soc., 2001.

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2018.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Dougal Sutherland and Jeff Schneider. On the error of random fourier features. *Uncertainty in Artificial Intelligence - Proceedings of the 31st Conference, UAI 2015*, 06 2015.

Andrew J Taylor, Victor D Dorobantu, Hoang M Le, Yisong Yue, and Aaron D Ames. Episodic learning with control lyapunov functions for uncertain robotic systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6878–6884. IEEE, 2019.

Andrew J. Taylor, Victor D. Dorobantu, Sarah Dean, Benjamin Recht, Yisong Yue, and Aaron D. Ames. Towards robust data-driven control synthesis for nonlinear systems with actuation uncertainty. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6469–6476, 2021. doi: 10.1109/CDC45484.2021.9683511.

Russ Tedrake. *Underactuated Robotics*. 2023. URL https://underactuated.csail.mit.edu.

Geoffrey S Watson. Linear least squares regression. *The Annals of Mathematical Statistics*, pages 1679–1699, 1967.

Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Appendix A. Omitted Proofs

A.1. Proof of Theorem 6

First note that for the ADP basis,

$$\boldsymbol{\phi}_c(\boldsymbol{x},\boldsymbol{u})^\top \boldsymbol{\phi}_c(\boldsymbol{x}',\boldsymbol{u}') = \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix} \operatorname{diag}(\boldsymbol{\psi}_1(\boldsymbol{x})^\top \boldsymbol{\psi}_1(\boldsymbol{x}'), \cdots, \boldsymbol{\psi}_{m+1}(\boldsymbol{x})^\top \boldsymbol{\psi}_{m+1}(\boldsymbol{x}')) \begin{bmatrix}\boldsymbol{u}'^\top & 1\end{bmatrix}^\top.$$

The result holds because by assumption, the expectation of $\boldsymbol{\psi}_i(\boldsymbol{x})^\top \boldsymbol{\psi}_i(\boldsymbol{x}')$ equals $k_i(\boldsymbol{x}-\boldsymbol{x}')$.

A.2. Proof of Theorem 8

Consider the following claim: if $k(\boldsymbol{x},\boldsymbol{x}')$ is a normalized shift invariant reproducing kernel, then $k(\boldsymbol{x},\boldsymbol{x}') - k(\boldsymbol{x})k(\boldsymbol{x}')$ is also a reproducing kernel.

For now we take the claim as true. Then notice that the AD kernel can also be written as

$$k_d((\boldsymbol{x},\boldsymbol{u}),(\boldsymbol{x}',\boldsymbol{u}')) = \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix} \tilde{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{x}') \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix}^\top + \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix} \tilde{\boldsymbol{A}}(\boldsymbol{x},\boldsymbol{x}') \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix}^\top,$$

for $\tilde{\boldsymbol{D}}(\boldsymbol{x},\boldsymbol{x}')$ a diagonal matrix with i^{th} entry as $k_i(\boldsymbol{x}-\boldsymbol{x}') - k_i(\boldsymbol{x})k_i(\boldsymbol{x}')$, and $\tilde{\boldsymbol{A}}(\boldsymbol{x},\boldsymbol{x}')$ a matrix whose entry at i,j is $k_i(\boldsymbol{x})k_j(\boldsymbol{x}')$ for $i,j \in [m+1]$. Since sums of kernels are also kernels, it suffices to show that each term is a kernel. If the claim above is true, then the first term is a special case of the ADP kernel, and is therefore a reproducing kernel by Lemma 3 of Castañeda et al. (2021). The second term can be directly written as an inner product $\begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix} \tilde{\boldsymbol{A}}(\boldsymbol{x},\boldsymbol{x}') \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix}^\top = \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix} \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}')^\top \begin{bmatrix}\boldsymbol{u}^\top & 1\end{bmatrix}^\top$ where $\boldsymbol{\phi}(\boldsymbol{x}) = [k_1(\boldsymbol{x}) \ \ldots \ k_{m+1}(\boldsymbol{x})]^\top$. Therefore, it is a kernel.

It now remains to prove the claim. We proceed by showing that $k(\boldsymbol{x},\boldsymbol{x}') - k(\boldsymbol{x})k(\boldsymbol{x}')$ can be written as the inner product of some explicit feature representation. Let $k(\boldsymbol{x},\boldsymbol{x}') = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}')\rangle$ for some feature function mapping to an arbitrary real Hilbert space $\mathcal{H}$, $\boldsymbol{\varphi} : \mathcal{X} \to \mathcal{H}$. This must exist since k is a reproducing kernel (Berlinet and Thomas-Agnan, 2011). Then

$$k(\boldsymbol{x},\boldsymbol{x}') - k(\boldsymbol{x})k(\boldsymbol{x}') = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}')\rangle - \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(0)\rangle\langle \boldsymbol{\varphi}(\boldsymbol{x}'), \boldsymbol{\varphi}(0)\rangle = \langle \boldsymbol{\varphi}(\boldsymbol{x}), (\mathcal{I}-\mathcal{P})\,\boldsymbol{\varphi}(\boldsymbol{x}')\rangle$$

where $\mathcal{I}$ is identity operator and $\mathcal{P} : \mathcal{H} \to \mathcal{H}$ is defined as the linear operator $\mathcal{P}\boldsymbol{v} = \boldsymbol{\varphi}(0)\langle \boldsymbol{\varphi}(0), \boldsymbol{v}\rangle$ for $\boldsymbol{v} \in \mathcal{H}$. We now argue that $\mathcal{I}-\mathcal{P}$ is a bounded self-adjoint positive operator; it is bounded, i.e. maps bounded subsets to bounded subsets, since $\mathcal{I}$ and $\mathcal{P}$(by Cauchy Schwarz) are bounded; it is positive, i.e., the quadratic form $\boldsymbol{v} \mapsto \langle \boldsymbol{v}, (\mathcal{I}-\mathcal{P})\,\boldsymbol{v}\rangle$ is positive semi-definite:

$$\langle \boldsymbol{v}, (\mathcal{I}-\mathcal{P})\,\boldsymbol{v}\rangle = \|\boldsymbol{v}\|_{\mathcal{H}}^2 - \langle \boldsymbol{\varphi}(0), \boldsymbol{v}\rangle^2 \geq \|\boldsymbol{v}\|_{\mathcal{H}}^2 - (\|\boldsymbol{\varphi}(0)\|_{\mathcal{H}}\|\boldsymbol{v}\|_{\mathcal{H}})^2 = 0.$$

The first inequality holds by Cauchy Schwarz, and the final equality holds because the kernel is normalized, i.e., $k(0) = \|\boldsymbol{\varphi}(0)\|_{\mathcal{H}}^2 = 1$. It is self-adjoint:

$$\begin{aligned}\langle \boldsymbol{v}, (\mathcal{I}-\mathcal{P})\,\boldsymbol{w}\rangle &= \langle \boldsymbol{v}, \boldsymbol{w}\rangle - \langle \boldsymbol{v}, \boldsymbol{\varphi}(0)\rangle\langle \boldsymbol{\varphi}(0), \boldsymbol{w}\rangle, \\ &= \langle \mathcal{I}\boldsymbol{v}, \boldsymbol{w}\rangle - \langle \boldsymbol{\varphi}(0)\langle \boldsymbol{\varphi}(0), \boldsymbol{v}\rangle, \boldsymbol{w}\rangle = \langle (\mathcal{I}-\mathcal{P})\,\boldsymbol{v}, \boldsymbol{w}\rangle.\end{aligned}$$

Therefore by theorem 13.15 in Schechter (2001), there exists a unique positive linear operator $\mathcal{L}$ such that $\mathcal{I}-\mathcal{P} = \mathcal{L}^2$, and therefore

$$k(\boldsymbol{x},\boldsymbol{x}') - k(\boldsymbol{x})k(\boldsymbol{x}') = \langle \boldsymbol{\varphi}(\boldsymbol{x}), (\mathcal{I}-\mathcal{P})\,\boldsymbol{\varphi}(\boldsymbol{x}')\rangle = \langle \mathcal{L}\boldsymbol{\varphi}(\boldsymbol{x}), \mathcal{L}\boldsymbol{\varphi}(\boldsymbol{x}')\rangle.$$

Therefore, we have constructed an explicit feature representation which proves the claim.

A.3. Proof of Theorem 9

First note that by assumption, the expectation of $\boldsymbol{\psi}_i(\boldsymbol{x})^\top \boldsymbol{\psi}_i(\boldsymbol{x}')$ is equal $k_i(\boldsymbol{x}-\boldsymbol{x}')$. Thus it remains to show that $\boldsymbol{\psi}_i(\boldsymbol{x})^\top \boldsymbol{\psi}_j(\boldsymbol{x}')$ estimates $k_i(\boldsymbol{x})k_j(\boldsymbol{x}')$ when $i \neq j$. Recall that by assumption, $\boldsymbol{\psi}_i$ is constructed using i.i.d. samples from the inverse Fourier transform of the kernel k_i, i.e., $p_i(\boldsymbol{\vartheta})$. Define $\zeta_{\boldsymbol{\vartheta}}(\boldsymbol{x}) = e^{i\boldsymbol{\vartheta}^\top \boldsymbol{x}}$. Then the expectation is given by,

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\vartheta},\boldsymbol{\vartheta}'}[\zeta_{\boldsymbol{\vartheta}}(\boldsymbol{x})\overline{\zeta_{\boldsymbol{\vartheta}'}(\boldsymbol{x}')}] &= \int_{\boldsymbol{\vartheta},\boldsymbol{\vartheta}'\in\mathbb{R}^n} p_i(\boldsymbol{\vartheta})p_j(\boldsymbol{\vartheta}')e^{i\boldsymbol{\vartheta}^\top \boldsymbol{x}}e^{-i\boldsymbol{\vartheta}'^\top \boldsymbol{x}'}d\boldsymbol{\vartheta}d\boldsymbol{\vartheta}', \\
&= \int_{\boldsymbol{\vartheta}\in\mathbb{R}^n} p_i(\boldsymbol{\vartheta})e^{i\boldsymbol{\vartheta}^\top \boldsymbol{x}}d\boldsymbol{\vartheta} \int_{\boldsymbol{\vartheta}'\in\mathbb{R}^n} p_j(\boldsymbol{\vartheta}')e^{-i\boldsymbol{\vartheta}'^\top \boldsymbol{x}'}d\boldsymbol{\vartheta}', \\
&= \mathbb{E}_{\boldsymbol{\vartheta}}[\zeta_{\boldsymbol{\vartheta}}(\boldsymbol{x})]\mathbb{E}_{\boldsymbol{\vartheta}'}[\overline{\zeta_{\boldsymbol{\vartheta}'}(\boldsymbol{x}')}] = k_i(\boldsymbol{x})k_j(\boldsymbol{x}').
\end{aligned}$$

A.4. Proof of Proposition 10

The proof of this proposition is given in the first two steps of the proof of Proposition 12.

Appendix B. Gaussian Process Regression

An important advantage of kernel methods is that they are amenable to theoretical guarantees and uncertainty characterization. Gaussian Process (GP) regression (Williams and Rasmussen, 2006) takes a Bayesian perspective, and provides posterior mean and variance estimates on function values. Confidence intervals of this form can be derived for kernel predictions even in frequentist settings (Srinivas et al., 2009). Prior works have developed robust CCF-based controllers for unknown models by incorporating Gaussian process (GP) regression (Castañeda et al., 2021; Castañeda et al., 2021). We thus describe how to use our approximation methods for GP regression. These results may be broadly of interest for any controller which makes use of GPs (Koller et al., 2018; Caldwell and Marshall, 2021; Bradford et al., 2019; Hewing et al., 2020; Li et al., 2021).

B.1. Background

This section presents an overview of regression methods which explicitly model the uncertainty. Bayesian linear regression is a probabilistic approach to regression analysis that models the relationship between a set of input vectors $\{\boldsymbol{s}_i\}_{i=1}^N \in \mathcal{S} \subseteq \mathbb{R}^d$ and target output variables $\{z_i\}_{i=1}^N \in \mathcal{Z} \subseteq \mathbb{R}$ as

$$z_i = h(\boldsymbol{s}_i) + \epsilon_i, \quad h(\boldsymbol{s}_i) = \boldsymbol{s}_i^\top \boldsymbol{w},$$

where $\boldsymbol{w} \in \mathbb{R}^d$ represents the linear model and $\{\epsilon_i\}_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_N^2)$ are the noise. In contrast to classical linear regression, Bayesian linear regression (BLR) not only estimates the parameters of a linear model, but also provides a probabilistic interpretation of the model's uncertainty. More specifically, BLR treats the regression coefficients as random variables with a prior distribution, and computes the posterior distribution over these coefficients given N input-output data pairs $\{(\boldsymbol{s}_i, z_i)\}_{i=1}^N$.

The Bayesian linear model suffers from limited expressiveness. A simple approach to overcome this problem is to map the input vectors $\{\boldsymbol{s}_i\}_{i=1}^N$ to a higher-dimensional feature space, where a linear relationship can be established more easily. If such a map exists, we call it a basis function.

Specifically, let $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^D$ maps an input vector $\boldsymbol{s} \in \mathbb{R}^d$ to a feature vector $\boldsymbol{\phi}(\boldsymbol{s}) \in \mathbb{R}^D$. Now the model becomes $h(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^\top \boldsymbol{w}$. Further, let the matrix $\Phi \in \mathbb{R}^{N \times D}$ and the vector $\boldsymbol{z} \in \mathbb{R}^N$ be the aggregation of rows $\{\boldsymbol{\phi}(\boldsymbol{s}_i)^\top\}_{i=1}^N$ and $\{z_i\}_{i=1}^N$, respectively. Assuming a Gaussian prior on the model weights, i.e., $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_w)$, the posterior distribution of $h(\boldsymbol{s}_*)$ at a query point $\boldsymbol{s}_*$ is $h(\boldsymbol{s}_*) \sim \mathcal{N}(\mu_*, \sigma_*^2)$ with,

$$\begin{aligned} \mu_* &= \boldsymbol{\phi}_*^\top \boldsymbol{\Sigma}_w \Phi^\top (\Phi \boldsymbol{\Sigma}_w \Phi^\top + \sigma_N^2 \boldsymbol{I}_N)^{-1} \boldsymbol{z}, \\ \sigma_*^2 &= \boldsymbol{\phi}_*^\top \boldsymbol{\Sigma}_w \boldsymbol{\phi}_* - \boldsymbol{\phi}_*^\top \boldsymbol{\Sigma}_w \Phi^\top (\Phi \boldsymbol{\Sigma}_w \Phi^\top + \sigma_N^2 \boldsymbol{I}_N)^{-1} \Phi \boldsymbol{\Sigma}_w \boldsymbol{\phi}_*, \end{aligned} \tag{7}$$

where we used the shorthand $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\boldsymbol{s}_*)$, $\mu_* = \mu(\boldsymbol{s}_*)$ and $\sigma_* = \sigma(\boldsymbol{s}_*)$ (Williams and Rasmussen, 2006). This distributional perspective characterizes the uncertainty in the model prediction. One simple way to express the uncertainty is through a confidence interval. For Gaussian distributions, confidence intervals take the form,

$$|\mu(\boldsymbol{s}_*) - h(\boldsymbol{s}_*)| \leq \beta \sigma(\boldsymbol{s}_*),$$

where $\beta \geq 0$ depends on the level of confidence. Below in Theorem 11, we present a formal version of this confidence bound that holds even in the frequentist setting, meaning that it does not depend on the distributional assumptions or Bayesian priors. However, the validity of the confidence interval does depend on the choice of basis functions, as this determines the complexity and richness of the modelled relationship between inputs $\boldsymbol{s}$ and outputs z. For data coming from a highly structured process, it may be reasonable to specify a basis of small dimension. However, in general a suitable compact basis may not be known a priori.

Kernel methods are used to allow for expressive basis functions of arbitrarily high or infinite dimension. Using the *kernel trick* (Scholkopf and Smola, 2018; Müller et al., 2018), the posterior (7) can be computed using only inner products of basis functions. A kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ generalizes the idea of inner products between basis functions. Using kernel functions in this context leads to the familiar Gaussian Process (GP) regression (Williams and Rasmussen, 2006). GPs are commonly used as a nonparametric approach for representing complex functions. GP regression corresponds to Bayesian regression in Reproducing Kernel Hilbert Spaces (RKHS), a specific class of function space denoted as $\mathcal{H}_k(\mathcal{S})$ (Wendland, 2004). The RKHS is equipped with the norm $\|h(\boldsymbol{s})\|_k := \sqrt{\langle h(\boldsymbol{s}), h(\boldsymbol{s}) \rangle}$ and is dense in the set of continuous functions, meaning that any continuous function can be represented arbitrarily well by kernel regression.

In the RKHS setting, Srinivas et al. (2009) establishes the following frequentist confidence interval.

Theorem 11 (Srinivas et al. (2009)) *Assume that the noise sequence $\{\epsilon_i\}_{i=1}^\infty$ is zero mean and uniformly bounded by σ_N. Let the target function $h : \mathcal{S} \to \mathbb{R}$ be a member of $\mathcal{H}_k(\mathcal{S})$ associated with a bounded kernel k, with its RKHS norm bounded by B. Then, with probability at least $1 - \delta$, the following holds for all $\boldsymbol{s} \in \mathcal{S}$ and $N \geq 1$:*

$$|\mu(\boldsymbol{s}_*) - h(\boldsymbol{s}_*)| \leq \sqrt{2B^2 + 300\gamma_{N+1} + \ln^3(\frac{N+1}{\delta})}\sigma(\boldsymbol{s}_*),$$

where γ_{N+1} is the maximum information gain after getting $N+1$ data points.

The drawback of kernel methods is computation. Algorithms for fitting functions in an RKHS to data have superlinear complexity in the number of data points. In particular, computing the kernel approximator can be prohibitively expensive for large datasets. Solving (7) generally requires $O(N^3)$ time and $O(N^2)$ memory.

B.2. Affine posterior

In this section, we explicitly show the affineness of predicted μ and σ in $\boldsymbol{u}$ as products of input-dependent and independent parts. We will use these calculation in the case study D.1 to control an acrobat using CCF functions. Throughout, we define $\boldsymbol{y} := [\boldsymbol{u}^\top \ 1]^\top$.

Kernel methods Under BLR assumptions, given a set of finite measurements of features and labels of the form $\{(\boldsymbol{s}_i, z_i)\}_{i=1}^N$, where $z_i = h(\boldsymbol{s}_i) + \epsilon_i$, and $\epsilon_i \sim \mathcal{N}(0, \lambda_N^2)$, a posterior distribution of $h(\boldsymbol{s})$ at a query point $\boldsymbol{s} := (\boldsymbol{x}, \boldsymbol{u})$ can be derived as follows: $h(\boldsymbol{x}, \boldsymbol{u}) \sim \mathcal{N}(\mu_x(\boldsymbol{u}), \sigma_x(\boldsymbol{u})^2)$ with

$$\mu_x(\boldsymbol{u}) := \mu(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{z}^\top (\boldsymbol{K} + \lambda_N^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_{(x,u)}, \tag{8}$$

$$\sigma_x(\boldsymbol{u})^2 := \sigma(\boldsymbol{x}, \boldsymbol{u})^2 = k\left((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}, \boldsymbol{u})\right) - \boldsymbol{k}_{(x,u)}^\top (\boldsymbol{K} + \lambda_N^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_{(x,u)}, \tag{9}$$

where $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is the Gram matrix whose entry at i, j is given by $[\boldsymbol{K}]_{i,j} = k((\boldsymbol{x}_i, \boldsymbol{u}_i), (\boldsymbol{x}_j, \boldsymbol{u}_j))$ for $i, j \in [N]$. Further, $\boldsymbol{k}_{(x,u)} = [k((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}_1, \boldsymbol{u}_1)) \ \cdots \ k((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}_N, \boldsymbol{u}_N))]^\top \in \mathbb{R}^N$, and $\boldsymbol{z} \in \mathbb{R}^N$ is the vector containing the output measurements $z_i = h(s_i) + \epsilon_i$ for $i \in [N]$.

In the following, we will show the affineness of $\mu_x(\boldsymbol{u})$ and $\sigma_x(\boldsymbol{u})^2$, when $k((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}'))$ is an AD kernel (Definition 7). We refer to Castañeda et al. (2021) section 5 for the case of ADP kernel (Definition 5). To proceed let $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ be the two sets containing the training data, where $\boldsymbol{y}_i = [\boldsymbol{u}^\top \ 1]^\top$. Let $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{x}')$ and $\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{x}')$ be as in Definition 7, and set $\boldsymbol{M}(\boldsymbol{x}, \boldsymbol{x}') := \boldsymbol{D}(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{A}(\boldsymbol{x}, \boldsymbol{x}')$. Further, define

$$\boldsymbol{k}_{train} = \operatorname{blkdiag}(\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_N^\top)[\boldsymbol{M}(\boldsymbol{x}_1, \boldsymbol{x})^\top \ \ldots \ \boldsymbol{M}(\boldsymbol{x}_N, \boldsymbol{x})^\top]^\top.$$

It's immediate that $\boldsymbol{k}_{(x,u)} = \boldsymbol{k}_{train} \boldsymbol{y}$. Therefore,

$$\begin{aligned}
\mu_x(\boldsymbol{u}) &= \underbrace{\boldsymbol{z}^\top (\boldsymbol{K} + \lambda_N^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_{train}}_{=: \boldsymbol{\Xi}_{ADK}(\mathcal{X}, \mathcal{Y})} \boldsymbol{y}, \\
&= \boldsymbol{\Xi}_{ADK}(\mathcal{X}, \mathcal{Y}) \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix}, \\
\sigma_x(\boldsymbol{u})^2 &= \boldsymbol{y}^\top \boldsymbol{M}(\boldsymbol{x}, \boldsymbol{x}) \boldsymbol{y} - \boldsymbol{y}^\top \boldsymbol{k}_{train}^\top (\boldsymbol{K} + \lambda_N^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_{train} \boldsymbol{y}, \\
&= \boldsymbol{y}^\top \underbrace{\left[\boldsymbol{M}(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_{train}^\top (\boldsymbol{K} + \lambda_N^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_{train}\right]}_{=: \boldsymbol{G}_{ADK}(\mathcal{X}, \mathcal{Y})} \boldsymbol{y}, \\
&= [\boldsymbol{u}^\top \ 1] \boldsymbol{G}_{ADK}(\mathcal{X}, \mathcal{Y}) \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix}.
\end{aligned}$$

Since every nonzero real vector can be scaled to have 1 as the last entry, and from above $\sigma_x(\boldsymbol{u})^2 = [\boldsymbol{u}^\top \ 1] \boldsymbol{G}_{ADK}(\mathcal{X}, \mathcal{Y}) \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix}$ for every $\boldsymbol{u}$, $\boldsymbol{G}_{ADK}(\mathcal{X}, \mathcal{Y})$ is positive semi-definite.

$$\implies \sigma_x(\boldsymbol{u}) = \left\| \boldsymbol{\Omega}_{ADK}(\mathcal{X}, \mathcal{Y}) \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix} \right\|_2.$$

Random basis methods Recall that using random features, the posterior mean and covariance can be approximated by

$$\hat{\mu}_x(\boldsymbol{u}) = \boldsymbol{\varphi}(\boldsymbol{x}, \boldsymbol{u})^\top (\Phi^\top \Phi + \lambda_N \boldsymbol{I}_D)^{-1} \Phi^\top \boldsymbol{z}, \tag{10}$$

$$\hat{\sigma}_x(\boldsymbol{u})^2 = \lambda_N \boldsymbol{\varphi}(\boldsymbol{x}, \boldsymbol{u})^\top (\Phi^\top \Phi + \lambda_N \boldsymbol{I}_D)^{-1} \boldsymbol{\varphi}(\boldsymbol{x}, \boldsymbol{u}). \tag{11}$$

ADP random features: From Definition 3, we know $\boldsymbol{\varphi}(\boldsymbol{x},\boldsymbol{u}) = \text{blkdiag}(\boldsymbol{\psi}_1(\boldsymbol{x}),\ldots,\boldsymbol{\psi}_{m+1}(\boldsymbol{x}))\boldsymbol{y}$. Define $\boldsymbol{\Psi}_{ADP}(\boldsymbol{x}) := \text{blkdiag}(\boldsymbol{\psi}_1(\boldsymbol{x}),...,\boldsymbol{\psi}_{m+1}(\boldsymbol{x}))$. As a result

$$\begin{aligned}
\hat{\mu}_x(\boldsymbol{u}) &= \boldsymbol{y}^\top \underbrace{\boldsymbol{\Psi}(\boldsymbol{x})^\top(\Phi^\top\Phi+\lambda_N \boldsymbol{I}_D)^{-1}\Phi^\top \boldsymbol{z}}_{\boldsymbol{\Xi}_{ADRF}(\mathcal{X},\mathcal{Y})},\\
&= [\boldsymbol{u}^\top \ \ 1]\boldsymbol{\Xi}_{ADRF}(\mathcal{X},\mathcal{Y}),\\
\hat{\sigma}_x(\boldsymbol{u})^2 &= \boldsymbol{y}^\top \underbrace{\lambda_N\boldsymbol{\Psi}(\boldsymbol{x})^\top(\Phi^\top\Phi+\lambda_N \boldsymbol{I}_D)^{-1}\boldsymbol{\Psi}(\boldsymbol{x})}_{\boldsymbol{G}_{ADPRF}(\mathcal{X},\mathcal{Y})}\boldsymbol{y},\\
&= [\boldsymbol{u}^\top \ \ 1]\lambda_N\boldsymbol{G}_{ADPRF}(\mathcal{X},\mathcal{Y})\begin{bmatrix}\boldsymbol{u}\\1\end{bmatrix},\\
\implies \hat{\sigma}_x(\boldsymbol{u}) &= \left\|\boldsymbol{\Omega}_{ADPRF}(\mathcal{X},\mathcal{Y})\begin{bmatrix}\boldsymbol{u}\\1\end{bmatrix}\right\|_2.
\end{aligned}$$

AD random features: From Definition 4, we know that $\boldsymbol{\varphi}(\boldsymbol{x},\boldsymbol{y}) = [\boldsymbol{\psi}_1(\boldsymbol{x}) \ \ldots \ \boldsymbol{\psi}_{m+1}(\boldsymbol{x})]\boldsymbol{y}$. Define $\boldsymbol{\Psi}_{AD}(\boldsymbol{x}) := [\boldsymbol{\psi}_1(\boldsymbol{x}) \ \ldots \ \boldsymbol{\psi}_{m+1}(\boldsymbol{x})]$. Then, similar to ADP random features, we have,

$$\begin{aligned}
\hat{\mu}_x(\boldsymbol{u}) &= [\boldsymbol{u}^\top \ \ 1]\boldsymbol{\Xi}_{ADRF}(\mathcal{X},\mathcal{Y}),\\
\hat{\sigma}_x(\boldsymbol{u})^2 &= [\boldsymbol{u}^\top \ \ 1]\lambda_N\boldsymbol{G}_{ADRF}(\mathcal{X},\mathcal{Y})\begin{bmatrix}\boldsymbol{u}\\1\end{bmatrix},\\
\implies \hat{\sigma}_x(\boldsymbol{u}) &= \left\|\boldsymbol{\Omega}_{ADRF}(\mathcal{X},\mathcal{Y})\begin{bmatrix}\boldsymbol{u}\\1\end{bmatrix}\right\|_2.
\end{aligned}$$

B.3. Error bounds

For the purposes of robust control, it is necessary to track how the approximation error accumulates in our computation of the posterior.

Sutherland and Schneider (2015) shows that with probability $1-\delta$, $\sup_{\boldsymbol{x}\in\mathcal{X}}|\boldsymbol{\psi}(\boldsymbol{x})^\top\boldsymbol{\psi}(\boldsymbol{x})-k(\boldsymbol{x})| \leq \epsilon$ for $D \geq \frac{8(d+2\alpha_\epsilon)}{\epsilon^2}\left[\frac{2}{1+\frac{2}{d}}\log\frac{\sigma_p l}{\epsilon} + log\frac{\beta_d}{\delta}\right]$, where l is the diameter of $\mathcal{X}$, $\sigma_p^2 = \mathbb{E}_p\|\boldsymbol{\omega}\|^2$ and β_d, α_e are defined in Proposition 1 of Sutherland and Schneider (2015).

Define $\boldsymbol{k}_s \in \mathbb{R}^N$ to be a vector containing the kernel $k(\boldsymbol{s}_i,\boldsymbol{s})$ for $i = 1,\ldots,N$. Let $\boldsymbol{U} \in \mathbb{R}^{N\times m}$ be a matrix with rows $\{\boldsymbol{u}_i^\top\}_{i=1}^N$. We get the following error bound.

Proposition 12 *Assume each i-th element of $\boldsymbol{\varphi}_C$ (14) is a member of $\mathcal{H}_{k_i}$ with bounded RKHS norm, for $i = 1,\ldots,m+1$. Assume either the AD or ADP kernel with bounded kernels k_i and that we have access to measurements $\boldsymbol{z}$. Assume $\|\boldsymbol{k}_s\| \leq \sqrt{N}\kappa$ and that $\lambda_n = n\lambda$. Let σ_{max} be the max singular value of $\boldsymbol{U}$. Then with a probability of $1-(\delta_1+\delta_2)$ we have:*

$$|\dot{C}(\boldsymbol{x},\boldsymbol{u}) - \hat{\mu}_{\boldsymbol{x}}(\boldsymbol{u})| \leq \beta\hat{\sigma}_{\boldsymbol{x}}(\boldsymbol{u}) + \epsilon(\nu\|\boldsymbol{u}_x\| + \iota v|u_x\|^2 + \Delta)$$

where $\nu := \frac{\sigma_{max}}{\sqrt{N\lambda}}(\sigma_n + \frac{2\beta\kappa}{\sqrt{N}} + 2\beta\epsilon), \iota = \frac{\beta\epsilon\sigma_{max}^2}{N\lambda}$,
$\Delta = \beta\Delta_\sigma + (\beta\kappa + \sqrt{N}\sigma_n)\Delta_\mu, \Delta_\mu = \frac{1}{\lambda\sqrt{N}}[1 + \frac{\kappa\sigma_{max}}{N\sqrt{N\lambda}} + \frac{\kappa}{\sqrt{N\lambda}}], \Delta_\sigma = 1+\epsilon+\frac{\kappa}{\sqrt{N\lambda}}$.

Proof we split the proof to several steps:

1. Approximating AD kernel pointwise:

$$\begin{aligned}|k_i(\boldsymbol{x})k_i(\boldsymbol{x}') - \hat{k}_i(\boldsymbol{x})\hat{k}_i(\boldsymbol{x}')| &= |k_i(\boldsymbol{x})(k_i(\boldsymbol{x}') - \hat{k}_i(\boldsymbol{x}')) + (k_i(\boldsymbol{x}) - \hat{k}_i(\boldsymbol{x}))\hat{k}_i(\boldsymbol{x})|,\\ &\leq \epsilon\, max(|k_i(\boldsymbol{x})|, |\hat{k}_i(\boldsymbol{x}')|),\\ &\leq \epsilon, \end{aligned} \tag{12}$$

where we used the assumption that $|k_i(\boldsymbol{x})| \leq 1$ for all $\boldsymbol{x} \in \mathcal{X}$ and all $i \in [m+1]$. Let $\boldsymbol{E} := \boldsymbol{A} + \boldsymbol{D} = [e_{ij}]_{\{i,j\}}$, that is, e_{ij} is the element in i^{th} row and j^{th} column of $\boldsymbol{E}$, where $\boldsymbol{A}, \boldsymbol{D}$ are as in Definition 7. Using (12), we conclude that $e_{ij} - \hat{e}_{ij} \leq \epsilon$ for all $1 \leq i, j \leq m+1$. This further implies,

$$\begin{aligned}|k_d((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}')) - \hat{k}_d((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}'))| &= |\sum_{1 \leq i,j \leq m+1} y_i(e_{ij} - \hat{e}_{ij})y'_j|\\ &\leq \epsilon(\boldsymbol{u}^\top \boldsymbol{u}' + 1),\end{aligned}$$

where $k_d((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}'))$ is in Definition 7 and y_i, y'_j denote the i^{th} and j^{th} elements of $\boldsymbol{y} := [\boldsymbol{u}^\top \ 1]^\top$ and $\boldsymbol{y}' := [\boldsymbol{u}'^\top \ 1]^\top$, respectively.

2. Approximating ADP Kernel pointwise:

$$\begin{aligned}|k_c((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}')) - \hat{k}_c((\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{x}', \boldsymbol{u}'))| &= |\sum_{1 \leq i \leq m+1} y_i(e_{ii} - \hat{e}_{ii})y'_i|\\ &\leq \epsilon(\boldsymbol{u}^\top \boldsymbol{u}' + 1),\end{aligned}$$

3. Approximating ADP, AD kernel matrices:
since we have the same bound on the estimation error of ADP kernel and AD kernel, the following proofs hold for both kernels:

$$\begin{aligned}\|\boldsymbol{K} - \hat{\boldsymbol{K}}\|_2 &\leq \epsilon\|[\boldsymbol{u}_i^\top \boldsymbol{u}_j + 1]_{i,j}\|_2 \leq \epsilon\sigma_{max}^2 + \epsilon(m+1)\\ \|\boldsymbol{k}_s - \hat{\boldsymbol{k}}_s\| &\leq \epsilon\|[\boldsymbol{u}_x^\top \boldsymbol{u}_i + 1]_i\| \leq \epsilon\|\boldsymbol{u}_x\|.\|[\boldsymbol{u}_1, ..., \boldsymbol{u}_N]\| + \epsilon\sqrt{N} \leq \epsilon\sigma_{max}\|\boldsymbol{u}_x\| + \epsilon\sqrt{N}\end{aligned}$$

4. Approximating the mean:

$$\begin{aligned}|\mu_x(\boldsymbol{u}) - \hat{\mu}_x(\boldsymbol{u})| &= \|\boldsymbol{z}^\top\|\|(\hat{\boldsymbol{K}} + \lambda_n \boldsymbol{I})^{-1}(\boldsymbol{k}_s - \hat{\boldsymbol{k}}_s) + ((\boldsymbol{K} + \lambda_n \boldsymbol{I})^{-1} - (\hat{\boldsymbol{K}} + \lambda_n \boldsymbol{I})^{-1})\boldsymbol{k}_s\|\\ &\leq \frac{\|\boldsymbol{z}\|}{\lambda_n}\|\hat{\boldsymbol{k}}_s - \boldsymbol{k}_s\| + \frac{\|\boldsymbol{z}\|.\|\hat{\boldsymbol{K}} - \boldsymbol{K}\|}{\lambda_n^2}\|\boldsymbol{k}_s\|\\ &\leq \frac{\sqrt{N}\sigma_n}{N\lambda}\|\hat{\boldsymbol{k}}_s - \boldsymbol{k}_s\| + \frac{\sqrt{N}\sigma_n\kappa}{n^2\lambda^2}\|\hat{\boldsymbol{K}} - \boldsymbol{K}\|\\ &\leq \frac{\sqrt{N}\sigma_n}{N\lambda}(\epsilon\sigma_{max}\|\boldsymbol{u}_x\| + \epsilon\sqrt{N}) + \frac{\sqrt{N}\sigma_n\kappa}{N^2\lambda^2}(\epsilon\sigma_{max}^2 + \epsilon N)\\ &\leq \epsilon\|\boldsymbol{u}_x\|\frac{\sigma_{max}\sigma_n}{\sqrt{N}\lambda} + \sqrt{N}\sigma_n\Delta_\mu\end{aligned}$$

Where we used $(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1} - (\hat{\boldsymbol{K}} + \lambda\boldsymbol{I})^{-1} = (\hat{\boldsymbol{K}} + \lambda\boldsymbol{I})^{-1}(\hat{\boldsymbol{K}} - \boldsymbol{K})(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}$, and that the smallest eigenvalue of $\hat{\boldsymbol{K}} + \lambda\boldsymbol{I}$ and $\boldsymbol{K} + \lambda\boldsymbol{I}$ is at least λ.

5. Approximating variance: Assume $\sigma_x(\boldsymbol{u}) + \hat{\sigma}_x(\boldsymbol{u}) \leq 1$

$$\begin{aligned}
|\sigma_x(\boldsymbol{u}) - \hat{\sigma}_x(\boldsymbol{u})| &\leq |\sigma_x(\boldsymbol{u})^2 - \hat{\sigma}_x(\boldsymbol{u})^2| \\
&\leq \epsilon + |\boldsymbol{k}_s[(\boldsymbol{K}+\lambda)^{-1}\boldsymbol{k}_s^\top - (\hat{\boldsymbol{K}}+\lambda)^{-1}\hat{\boldsymbol{k}}_s^\top] + [\boldsymbol{k}_s - \hat{\boldsymbol{k}}_s](\hat{\boldsymbol{K}}+\lambda)^{-1}\hat{\boldsymbol{k}}_s^\top| \\
&\leq \epsilon + \|\boldsymbol{k}_s\|\frac{|\mu_x(\boldsymbol{u}) - \hat{\mu}_x(\boldsymbol{u})|}{\|\boldsymbol{z}\|} + \frac{\epsilon\sigma_{max}\|\boldsymbol{u}_x\| + \epsilon\sqrt{N}}{N\lambda}\|\hat{\boldsymbol{k}}_s\| \\
&\leq \epsilon + \kappa(\epsilon\|\boldsymbol{u}_x\|\frac{\sigma_{max}^2}{N\lambda} + \Delta_\mu) \\
&\quad + \frac{\epsilon\sigma_{max}\|\boldsymbol{u}_x\| + \epsilon\sqrt{N}}{N\lambda}(\kappa + \epsilon\sigma_{max}^2\|\boldsymbol{u}_x\| + \epsilon\sqrt{N}) \\
&\leq \|\boldsymbol{u}_x\|\frac{2\epsilon\sigma_{max}^2}{\sqrt{N}\lambda}(\frac{\kappa}{\sqrt{N}} + \epsilon) + \|\boldsymbol{u}_x\|^2\frac{\epsilon^2\sigma_{max}^2}{N\lambda} + \Delta_\sigma + \kappa\Delta_\mu
\end{aligned}$$

6. Bounds on total error: with a probability of $1-(\delta_1+\delta_2)$:

$$\begin{aligned}
|\dot{C}_x(\boldsymbol{u}) - \hat{\mu}_x(\boldsymbol{u})| &\leq |\mu_x(\boldsymbol{u}) - \dot{C}_x(\boldsymbol{u})| + |\mu_x(\boldsymbol{u}) - \hat{\mu}_x(\boldsymbol{u})| \\
&\leq \beta\hat{\sigma}_x(\boldsymbol{u}) + \beta|\sigma_x(\boldsymbol{u}) - \hat{\sigma}_x(\boldsymbol{u})| + |\mu_x(\boldsymbol{u}) - \hat{\mu}_x(\boldsymbol{u})| \\
&\leq \beta\hat{\sigma}_x(\boldsymbol{u}) + \|\boldsymbol{u}_x\|\nu + \|\boldsymbol{u}_x\|^2\iota + \Delta_s
\end{aligned}$$

■

Appendix C. Empirical Compound Random Basis Comparison

In this Appendix, we perform experiments with synthetic data to demonstrate the relationship between training time and RMSE for ADP-RF and AD-RF models. Specifically, we use the following control-affine function to generate the data,

$$h_m(\boldsymbol{x}, \boldsymbol{u}) = 3\sin(2\pi\boldsymbol{x}^\top\boldsymbol{w}_1) - 2\sin(4\pi\boldsymbol{x}^\top\boldsymbol{w}_2) + \sum_{j=1}^{m}(\gamma_j\sin(2\pi\boldsymbol{x}^\top\boldsymbol{w}_{j+2}))u_j + \epsilon, \tag{13}$$

where $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^n$ parameterize $\boldsymbol{f}(\boldsymbol{x}) := 3\sin(2\pi\boldsymbol{x}^\top\boldsymbol{w}_1) - 2\sin(4\pi\boldsymbol{x}^\top\boldsymbol{w}_2)$, and $\{\boldsymbol{w}_{j+2}\}_{j=1}^m \in \mathbb{R}^n$ parameterize $\boldsymbol{g}(\boldsymbol{x}) := [\gamma_1\sin(2\pi\boldsymbol{x}^\top\boldsymbol{w}_3)\ \gamma_2\sin(2\pi\boldsymbol{x}^\top\boldsymbol{w}_4)\ \cdots\ \gamma_m\sin(2\pi\boldsymbol{x}^\top\boldsymbol{w}_{m+2})]$, such that $h_m(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} + \epsilon$ is affine in $\boldsymbol{u}$. All the (entries of) weights $\boldsymbol{w}$ and γ are sampled uniformly at random from $[0, 1)$.

We use the function (13) to generate data for varied input dimension $m = 1, ..., 20$. For each $m = k$, to generate $h_m(\boldsymbol{x}, \boldsymbol{u})$, we use the previous weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{k+1}, \gamma_1, \ldots, \gamma_{k-1}$ and only generate new weights for $j = k$, that is $\boldsymbol{w}_{k+2}$ and γ_k.

For each input dimension, we sample 1000 values of $\boldsymbol{x}_i \in \mathbb{R}^6$ and $\boldsymbol{u}_i \in \mathbb{R}^m$ uniformly at random from $[0, 1)$. For each i, we set the label $y_i = h_m(\boldsymbol{x}_i, \boldsymbol{u}_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, 0.01)$. Using this dataset, we train and evaluate AD-RF, ADP-RF on a 90/10 train/test split. For all methods, we use $\lambda = 1$ and rbf $\gamma = 1$. To choose feature dimensions, we vary the state-dependent basis dimension starting from 22, and is incremented by 22 for 10 steps; this is roughly chosen based on the widely considered best random features dimension of $\sqrt{n}\log n$. The ADP compound basis

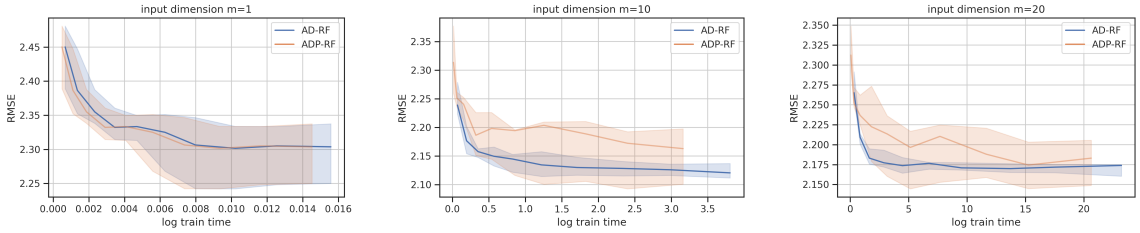

Figure 3: Evaluation of models, comparing prediction accuracy on 100 test data points against average training time on 900 data points for $m = 1, 10, 20$ respectively. Random features are sampled 10 times at matching compound dimensions; the panels display median and quartile RMSE over the trials. Increasing m demonstrates the advantage of AD-RF basis over ADP-RF in RMSE for fixed train-time.

dimension would be the multiplication of these dimensions by $m+1$. For fair comparison, we make sure that both the compound basis dimensions match. At each dimension, we resample the random features 10 times, and record the training time and test RMSE.

Figure 3 plots the median and quartile RMSE against the average training time. For lower dimensional inputs $\boldsymbol{u}$, at a given train time, ADP-RF and AD-RF perform similarly. However with increasing m, the performance gap between AD-RF and ADP-RF becomes larger. Specifically at larger m's, AD-RF converges faster at a lower RMSE; This means AD-RF reaches its optimal RMSE at a lower train-time, which implies that at lower feature dimensions, AD-RF captures the complexity of the model, whereas to ensure the same for ADP-RF, we need more complex and higher dimensional features.

Appendix D. Robust CCF Control

In Section 4, we present a *certainty-equivalent* approach to data-driven control using CCF control as a case study. Here, we additionally present a *robust* approach. It allows for synthesizing a data-driven control law which robustly enforces the constraint in **CCF-QP**.

D.1. Robust Bayesian Data-driven Controller

In this section, we show how to construct a robust data driven control law given the control-affine basis functions (introduced in Section 3.1). Similar to Section 4, we suppose that the dynamics $\boldsymbol{f}$ and $\boldsymbol{g}$ are unknown, so the robust CCF controller cannot be directly implemented. We assume that a valid CCF C and comparison function α for the unknown true system is given. Given a sampled trajectory $\{(\boldsymbol{x}_i, \boldsymbol{u}_i)\}_{i=1}^N$, we construct a control affine modelling problem for $\dot{C} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ as described in Example 3. Specifically, we use GP regression to model the uncertainty and compute the mean $\mu_{\boldsymbol{x}}(\boldsymbol{u}) := \mu(\boldsymbol{s})$ and variance $\sigma_{\boldsymbol{x}}(\boldsymbol{u}) := \sigma(\boldsymbol{s})$ according to (7). As first proposed in the GP context by Castañeda et al. (2021), we make use of the $1-\delta$ confidence interval presented in Theorem 11 to construct an optimization-based controller. The data-driven min-norm stabilizing

feedback control law $\boldsymbol{u}^*\colon \mathcal{X} \to \mathcal{U}$ is defined as

$$\begin{aligned} \boldsymbol{u}^*(\boldsymbol{x}) = \underset{\boldsymbol{u}\in\mathcal{U}}{\arg\min} \|\boldsymbol{u}\|_2^2 \qquad & \textbf{(BLR-CCF-SOCP)} \\ \text{s.t.} \quad \mu_{\boldsymbol{x}}(\boldsymbol{u}) + \beta\sigma_{\boldsymbol{x}}(\boldsymbol{u}) + \alpha(C(\boldsymbol{x})) \leq 0 & \end{aligned}$$

This optimization problem will be a Second-Order Cone Program (SOCP) as long as the constraint is a conic in $\boldsymbol{u}$. This follows from the form of the mean and variance function, and can be guaranteed as long as the basis function $\boldsymbol{\phi}$ is *affine* in $\boldsymbol{u}$. This is a natural requirement due to the affine structure of the dynamics.

$$\begin{aligned} \dot{C}(\boldsymbol{x},\boldsymbol{u}) &= \nabla C(\boldsymbol{x})^\top \boldsymbol{f}(\boldsymbol{x}) + (\nabla C(\boldsymbol{x})^\top \boldsymbol{g}(\boldsymbol{x}))\boldsymbol{u}, \\ &= \boldsymbol{\varphi}_C \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix} \end{aligned} \tag{14}$$

where $\boldsymbol{\varphi}_C \in \mathbb{R}^{1\times(m+1)}$.

When a random features approximation is used, there is an additional error term that much be accounted for in proper robust control. Leveraging the error analysis presented in Section B.3, we present (**RF-CCF-SOCP**) which is both computationally efficient and robust.

$$\begin{aligned} \boldsymbol{u}^*(\boldsymbol{x}) = \underset{\boldsymbol{u}\in\mathbb{R}^m}{\arg\min} \|\boldsymbol{u}\|_2^2 \qquad & \textbf{(RF-CCF-SOCP)} \\ \text{s.t.} \quad \hat{\mu}_x(\boldsymbol{u}) + \beta\hat{\sigma}_x(\boldsymbol{u}) + \epsilon(\nu\|\boldsymbol{u}_x\| + \iota\|\boldsymbol{u}_x\|^2 + \Delta) + \alpha(C(\boldsymbol{x})) \leq 0 & \end{aligned}$$

Appendix E. Experimental Details

E.1. Double pendulum dynamics derivation

We consider a frictionless two-link pendulum with torque τ_1 applied at a fixed base, where the first link is attached, and torque τ_2 applied at the opposite end, where the second link is attached. The links are modeled as point masses m_1 and m_2 at lengths l_1 and l_2 from the joints. We define θ_1 as the angle of the first link, measured from the upright positive, and θ_2 as the angle of the second link, measured from the first link. The corresponding angular rates are $\dot{\theta}_1$ and $\dot{\theta}_2$.

Letting $\boldsymbol{q} = [\theta_1, \theta_2]$, the total kinetic energy of the system is given by

$$\begin{aligned} T(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2}((m_1+m_2)l_1^2 + 2m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\dot{\theta}_1^2 + (m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\dot{\theta}_1\dot{\theta}_2 \\ + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2, \end{aligned} \tag{15}$$

and the potential energy of the system is given by

$$U(\boldsymbol{q}) = (m_1+m_2) g l_1 \cos\theta_1 + m_2 g l_2 \cos(\theta_1+\theta_2). \tag{16}$$

As a result, the Lagrangian of the system takes the following form,

$$\begin{aligned} L(\boldsymbol{q},\dot{\boldsymbol{q}}) &= T(\boldsymbol{q},\dot{\boldsymbol{q}}) - U(\boldsymbol{q}), \\ &= \frac{1}{2}((m_1+m_2)l_1^2 + 2m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\dot{\theta}_1^2 + (m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\dot{\theta}_1\dot{\theta}_2, \\ &\quad + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 - (m_1+m_2) g l_1 \cos\theta_1 - m_2 g l_2 \cos(\theta_1+\theta_2). \end{aligned} \tag{17}$$

Now we can write the Lagrange equations as follows

$$\frac{d}{dt}\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \boldsymbol{q}} = \boldsymbol{\tau}. \tag{18}$$

The partial derivatives are given by

$$\begin{aligned}
\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\theta}_1} &= ((m_1+m_2)l_1^2 + 2m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\dot{\theta}_1 + (m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\dot{\theta}_2,\\
\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \theta_1} &= (m_1+m_2) g l_1 \sin\theta_1 + m_2 g l_2 \sin(\theta_1+\theta_2),\\
\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\theta}_2} &= (m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\dot{\theta}_1 + m_2 l_2^2 \dot{\theta}_2,\\
\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \theta_2} &= -m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1^2 - m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 g l_2 \sin(\theta_1+\theta_2).
\end{aligned}$$

This further implies,

$$\begin{aligned}
\frac{d}{dt}\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\theta}_1} &= ((m_1+m_2)l_1^2 + 2m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\ddot{\theta}_1 + (m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\ddot{\theta}_2\\
&\qquad - 2m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2^2,\\
\frac{d}{dt}\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\theta}_2} &= (m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 - m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 \dot{\theta}_2.
\end{aligned}$$

With these results, we can write the following Lagrange equations

$$\begin{aligned}
&\frac{d}{dt}\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\theta}_1} - \frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \theta_1} = \tau_1,\\
\implies &((m_1+m_2)l_1^2 + 2m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\ddot{\theta}_1 + (m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\ddot{\theta}_2\\
&- 2m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2^2 - (m_1+m_2) g l_1 \sin\theta_1 - m_2 g l_2 \sin(\theta_1+\theta_2) = \tau_1.
\end{aligned}$$

Similarly,

$$\begin{aligned}
&\frac{d}{dt}\frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\theta}_2} - \frac{\partial L(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \theta_2} = \tau_2,\\
\implies &(m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 - m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1^2\\
&+ m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 \dot{\theta}_2 - m_2 g l_2 \sin(\theta_1+\theta_2) = \tau_2,\\
\implies &(m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2)\ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1^2 - m_2 g l_2 \sin(\theta_1+\theta_2) = \tau_2.
\end{aligned}$$

From the above Lagrange equations, we write the following manipulator equation,

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \boldsymbol{\tau}_g(\boldsymbol{q}) + \boldsymbol{B}\boldsymbol{u}. \tag{19}$$

where

$$\boldsymbol{M}(\boldsymbol{q}) := \begin{bmatrix} (m_1+m_2)l_1^2 + 2m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2 & m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2 \\ m_2 l_1 l_2 \cos\theta_2 + m_2 l_2^2 & m_2 l_2^2 \end{bmatrix}, \tag{20}$$

$$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) := \begin{bmatrix} -2m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2 & -m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2 \\ m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1 & 0 \end{bmatrix}, \tag{21}$$

$$\boldsymbol{\tau}_g(\boldsymbol{q}) := \begin{bmatrix} (m_1+m_2) g l_1 \sin\theta_1 + m_2 g l_2 \sin(\theta_1+\theta_2) \\ m_2 g l_2 \sin(\theta_1+\theta_2) \end{bmatrix}, \tag{22}$$

$$\boldsymbol{B} := \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{u} := \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}, \quad \boldsymbol{q} := \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}, \quad \dot{\boldsymbol{q}} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}, \quad \ddot{\boldsymbol{q}} = \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}. \tag{23}$$

Therefore, the state of the acrobat is $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}}) = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$, where, $\theta_1, \theta_2 \in [0, \pi]$ and $\dot{\theta}_1, \dot{\theta}_2 \in \mathbb{R}$. The input $\boldsymbol{u} = [\tau_1, \tau_2]$ is of 2 dimensions. The manipulator equation can be written as the following control affine dynamics.

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}(\boldsymbol{q})^{-1}\left(-\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q})\right) \end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x})} + \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{M}(\boldsymbol{q})^{-1}\boldsymbol{B} \end{bmatrix}}_{\boldsymbol{g}(\boldsymbol{x})} \boldsymbol{u} \tag{24}$$

E.2. CCF Modelling Problem

We consider a control affine modelling problem in the setting of learning the residual errors from a nominal dynamics model (Example 4) for a CCF (Example 3). The true dynamics model $\boldsymbol{f}, \boldsymbol{g}$ is defined according to the equations above with $m_1 = m_2 = l_1 = l_2 = 1$ while the nominal dynamics model $\tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}$ is defined with incorrect values of mass and length, $\tilde{m}_1 = \tilde{m}_2 = \tilde{l}_1 = \tilde{l}_2 = 0.6$. The CCF is a CLF and is defined to ensure stability to the origin:

$$C(\boldsymbol{x}) = \boldsymbol{x}^\top P \boldsymbol{x}, \quad P = \begin{bmatrix} 12 & 0 & 3.16 & 0 \\ 0 & 12 & 0 & 3.16 \\ 3.16 & 0 & 4.04 & 0 \\ 0 & 3.16 & 0 & 4.04 \end{bmatrix}.$$

Using the nominal model, $\dot{\tilde{C}}(\boldsymbol{x}, \boldsymbol{u}) = \nabla C(\boldsymbol{x})^\top (\tilde{\boldsymbol{f}}(\boldsymbol{x}) + \tilde{\boldsymbol{g}}(\boldsymbol{x})\boldsymbol{u})$. The goal of the modelling problem is to learn the residual $\dot{C}(\boldsymbol{x}, \boldsymbol{u}) - \dot{\tilde{C}}(\boldsymbol{x}, \boldsymbol{u})$. Given a sampled trajectory $\{\boldsymbol{x}_i, \boldsymbol{u}_i\}_{i=1}^{L+1}$, we compute $\{C(\boldsymbol{x}_i)\}_{i=1}^{L+1}$ and use forward finite differencing to approximate $\{\hat{\dot{C}}_i\}_{i=1}^{L}$. Then the regression targets are defined as $z_i = \hat{\dot{C}}_i - \dot{\tilde{C}}(\boldsymbol{x}_i, \boldsymbol{u}_i)$.

E.3. Nominal Control Data Collection

We collect data using a controller designed with the nominal dynamics models. The control law is given by **CCF-QP** with $c_1 = 25$, $\boldsymbol{u}_d(\boldsymbol{x})$ a feedback linearizing controller designed for nominal dynamics $\tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}$, $C(\boldsymbol{x}) = \boldsymbol{x}^\top P \boldsymbol{x}$ defined above, $\tilde{\boldsymbol{f}}, \tilde{\boldsymbol{g}}$ used in place of $\boldsymbol{f}, \boldsymbol{g}$, and $\alpha(c) = 0.725c$. Additionally, the hard constraint is replaced with a slack variable with penalty coefficient $1e6$.

The nominal control law is simulated in closed-loop with the true dynamics using Runge-Kutta 4(5) at 10 Hz. We collect $E = 226$ trajectories starting from different initial conditions. The initial

conditions comprise of a meshgrid of coordinates of the different initial states. Each trajectory is 5 seconds long, resulting in $L + 1 = 50$ sampled points, the final datasize is of size 11074.

For the prediction experiments, the data is split into a test and train set shuffled at random.

E.4. Closed-Loop Experiments

We evaluate six controllers starting from initial state $\boldsymbol{x}_0 = [2, 0, 0, 0]$. All simulations during data collection and evaluation use Runge-Kutta 4(5) at 10 Hz.

The nominal controller is described above. The oracle controller is given by **CCF-QP** with the same parameters as the nominal, except that the constraint uses the true dynamics $\boldsymbol{f}, \boldsymbol{g}$. Each of the four data-driven controllers is defined using an affine model of the residual $\hat{h}$. The control law is defined by **CCF-QP** where $\dot{C}(\boldsymbol{x}, \boldsymbol{u})$ is replaced with $\dot{\hat{C}}(\boldsymbol{x}, \boldsymbol{u}) + \hat{h}(\boldsymbol{x}, \boldsymbol{u})$, the slack penalty is $1e6 \cdot (t + 1)$ where t is the time in seconds, and otherwise the parameters are the same as for the nominal/oracle controllers. Section B.2 presents the precise affine form in terms of the training data.

Each data-driven model $\hat{h}$ is trained in an episodic manner. The process is warm started with subsampling the nominal grid data at a rate of 1/5 resulting in 2215 data points. This initial training dataset defines an affine model (AD-K, ADP-K, AD-RF, or ADP-RF) which in turn defines a data-driven controller. We simulate the data-driven controller in closed loop starting from $\boldsymbol{x}_0$ for 10 seconds, add the resulting data to the training set, and retrain. We repeat for 10 episodes, resulting in a training set size of 3215, and report the performance of the final controller.