NORTHWESTERN UNIVERSITY

Dynamical System Segmentation:
Data-Driven Symbolic Abstractions for Motion Analysis and Control

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

MASTER OF SCIENCE

Field of Mechanical Engineering

By

Thomas A. Berrueta

EVANSTON, ILLINOIS

August 2020

ABSTRACT

Dynamical System Segmentation:

Data-Driven Symbolic Abstractions for Motion Analysis and Control

Thomas A. Berrueta

Complexity abounds in both natural and engineered dynamical systems—from bipedal walkers to disorganized swarms of robots. It is the intrinsic complexity of such systems that makes first-principles reasoning often intractable, necessitating the use of data-driven techniques. A common feature of complex systems is that they exhibit multi-modality in their dynamics, displaying distinct behaviors at different points in time and space. Here, we start from the assumption that the system dynamics can be decomposed and effectively compressed into a few discrete behaviors, and develop algorithmic tools to this end. We introduce Dynamical System Segmentation (DSS), an algorithm that generates low-dimensional symbolic abstractions of the underlying multi-modal dynamics from data. With DSS we synthesize coarse-grained descriptions of dynamical systems that can be used towards system identification and control simultaneously. We validate the utility of DSS in the analysis and control of many simulated and experimental systems of interest, such as dynamic hoppers, lower-limb exoskeletons, and robotic assistive devices.

Acknowledgements

This work would not have been possible without the help of my many great colleagues including but not limited to Ana Pervan, Kathleen Fitzsimons, Aleksandra Kalinowska, and Ian Abraham. This work was supported by the National Science Foundation under grant CBET-1637764. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

List of Figures

CHAPTER 1

Introduction

The deceptively overwhelming complexity of dynamical processes in nature often hides an underlying simplicity. While there certainly exist chaotic systems whose lack of internal structure challenges analysis, very diverse dynamics can also arise from the interplay and composition of a relatively small set of primitive behaviors. Consider human walking gaits as an example. Walking results from the collective efforts of billions of neurons firing, resulting in the coupled activations and interactions of hundreds of muscles. However, the fact that these muscles must interact coherently constrains the near-infinite behaviors that the degrees of freedom of our bodies can achieve down to just a few (at least in prototypical healthy human gait). These few behaviors are then cyclically composed in order to generate walking gaits. Although the mechanisms underlying the rise of low-dimensional multi-modality across systems are likely varied, it is present across many natural and engineered systems.

Arriving at such a coarse-graining of the behavior of a complex system can be challenging due to the intrinsic nonlinearity, nonsmoothness, and nondeterminism that is often present in the dynamics. It is the intrinsic microscopic complexity of the system prevents detailed mathematical analysis. However, the shortcomings of first-principles approaches present an opportunity for the development of data-driven techniques to aid in filling the gap.

The problem of constructing effective coarse-grainings of dynamical systems from observations of the dynamics bears close resemblance to the problem of compression in information theory. In the information-theoretic sense, a dynamical system is simply a process that generates sequences of symbols over time. In this setting, compression is simply a question of how to efficiently represent symbolic sequences with fewer symbols. The particulars of the chosen compressed representation are determined by a choice of encoding procedure [**40**]. Taking inspiration from information theory, we are interested in developing an equivalent algorithmic coarse-graining procedure for dynamical systems.

Particularly, the goal of this work is to develop such an algorithmic method to effectively coarse-grain and compress the dynamics of a multi-modal dynamical system. To serve this purpose, the primary contribution of this work is an algorithm, Dynamical System Segmentation (DSS): an unsupervised, nonparametric system identification technique specializing in the synthesis of low-dimensional symbolic abstractions of dynamical systems. In addition to introducing DSS, we motivate the many possible uses of such a compression procedure for dynamical systems, and demonstrate its application across diverse domains, from quantitative motion analysis and hybrid systems control.

The structure of this manuscript is as follows: in Chapter 2, we introduce the Koopman operator both as an operator-theoretic description of dynamical systems, and as a data-driven tool for system identification. Then in Chapter 3, we introduce and describe DSS. In Chapter 4, we motivate DSS as a tool for both qualitative and quantitative motion analysis. In Chapter 5, we demonstrate that the models generated by DSS are effective in predicting the trajectories of nonlinear dynamical systems and towards synthesizing

model predictive control. Finally, in Chapter 6 we highlight some limitations of the current approach and point towards future work directions.

CHAPTER 2

Koopman Operators

To achieve our goal of compressing complex multi-modal system dynamics into few-state abstractions, we require a data-driven representation of a dynamical system that is amenable to the analysis we are proposing. While many different representation choices may be applicable, we focus on the Koopman operator for its particular suitability to robotics and control [**4**].

Koopman operator theory dates back to the work of Bernard Koopman and John von Neumann, and their search for connections between quantum mechanics and statistical mechanics rooted in operator theory [**25, 33**]. Of the many important results derived throughout this process, the Koopman operator formalism remained overlooked until recent years. The Koopman operator is an operator-theoretic formulation of dynamical systems that describes nonlinear dynamical systems in a linear representation without the need for approximation [**25**]. While this may seem too good to be true, there is an inherent trade-off that takes place as a result of this choice of representation: the linearized dynamical system now evolves through an infinite-dimensional function space. It is then as a result of this intractability that the Koopman operator received little attention for decades. However, recent computational developments in the numerical estimation of operators has enabled approximation of the Koopman operator in a variety of frameworks [**49, 31, 44, 8**].

In this chapter, we introduce and develop the basics of Koopman operator theory, as well as the necessary algorithmic tools we need for applying Koopman operators to the numerical identification of complex dynamics.

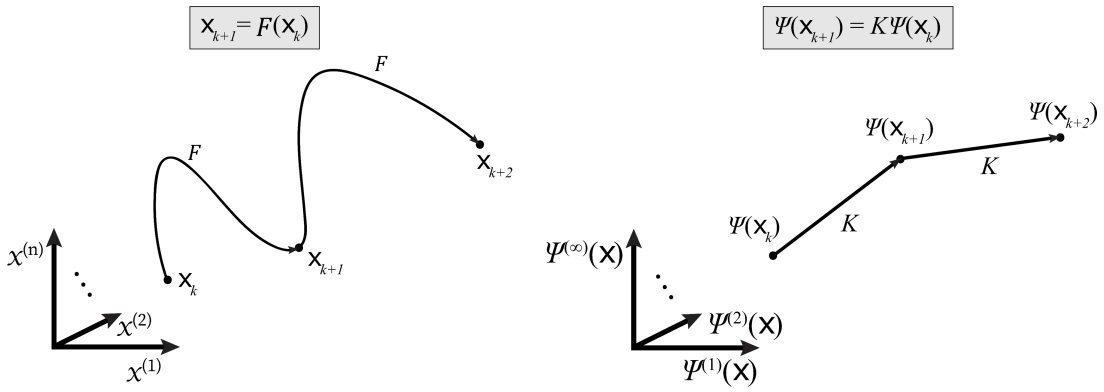
2.1. Theory

We begin by defining the domain of dynamical systems of interest to our analysis, which we restrict to measure-preserving discrete-time dynamical systems that can be described by

$$\vec{x}_{k+1} = F(\vec{x}_k) = \vec{x}_k + \int_{t_k}^{t_k+\Delta t} f(\vec{x}(\tau))d\tau, \tag{2.1}$$

where the map $F : X \to X$ describes the evolution of states $\vec{x}_k$ over time-indices $k \in \{\mathbb{Z}^+ \cup 0\}$ in a finite-dimensional metric space X (*e.g.*, a manifold or some region of Euclidean space). We note on the right-hand side of the equation that continuous-time dynamical systems are captured by this formalism through the evolution of the corresponding flow, f, over some interval Δt [**4**].

While one can focus on describing the orbits, trajectories, and attractors of the states of such a dynamical system, one can alternatively consider the behavior of an *observable* of the system. We can think of observables as real-valued functions of the state $\psi : X \to \mathbb{R}$, which are drawn from some function space $\psi \in \mathcal{F}$ that is often endowed with some properties that facilitate mathematical analysis [**29**]. For the purposes of this work, we consider $\mathcal{F}$ to be a subset of an infinite-dimensional Hilbert space of Lebesgue square-integrable functions that is closed under composition.



(a) Nonlinear state-space representation (b) Linear function-space representation

Figure 2.1. **Schematic of the relationship between a nonlinear dynamical system and its corresponding Koopman operator representation.** Given a nonlinear system in an n-dimensional state space as shown in (a), the Koopman operator is a linear representation of the system in an infinite-dimensional function space, shown in (b). The function-space coordinates, Ψ, are functions of the original system state that are referred to as observables. The observables lift the original system coordinates onto an infinite-dimensional function space. In this function space, the Koopman operator, K, is a linear map that captures the underlying system dynamics.

The evolution of all observables of our system dynamics is then given by the *Koopman operator*. The Koopman operator $K : \mathcal{F} \to \mathcal{F}$ associated with the system $F : X \to X$ can be described by its action on system observables:

$$K\psi = \psi \circ F, \ \forall \psi \in \mathcal{F}. \tag{2.2}$$

It is important to note that the Koopman operator formally provides a *global* picture of the nonlinear dynamics of a system in a linear representation [**29**]. Indeed, we can easily verify its linearity: $K(c_1\psi_1 + c_2\psi_2) = c_1 K\psi_1 + c_2 K\psi_2, \forall c_1, c_2 \in \mathbb{R}, \forall \psi_1, \psi_2 \in \mathcal{F}$. Additionally, we can equivalently describe the evolution of vector-valued observables $\Psi : X \to \mathbb{R}^m$ through this equation. In Figure 2.1, we provide a sketch of the relationship between a Koopman

operator representation of a dynamical system and its typical representation. Now that we have introduced the Koopman operator, as well as some of its theoretical properties, we can illustrate numerical approximation schemes for computing a Koopman operator in finite-dimensions.

2.2. Numerics

In this chapter we introduce Extended Dynamic Mode Decomposition (EDMD) [**49**] as an algorithm for synthesizing finite-dimensional approximations of the Koopman operator. Then, from this approximation as a starting point, we describe extensions of the Koopman formalism for applications in hybrid systems (*i.e.*, a particular type of multi-modality in dynamical systems), as well as control.

2.2.1. EDMD Approximation

We begin by framing Koopman operator synthesis as a learning problem. It is important to note that the Koopman operator itself is not a machine learning tool; it is an alternative operator-theoretic representation of dynamical systems. However, numerically synthesizing an approximation of this representation in finite dimensions from data is indeed a machine learning problem. In order to implement Koopman operators in computational applications, we generate finite-dimensional approximations of it in the form of a matrix operator. While certain systems can be represented by an exact, closed-form, finite-dimensional Koopman operator, this is generally not the case [**22**]. Dynamic Mode Decomposition (DMD) [**38**] and Extended Dynamic Mode Decomposition (EDMD) [**49**]

are some of many methods developed for approximating Koopman operators in finite dimensions and are commonly applied in fluid dynamics as analytical tools [**31**].

EDMD synthesizes a data-driven Koopman operator by constructing a linear map that tries to span a finite-dimensional subspace of a given function space specified by a finite set of basis functions. Given basis functions $\Psi(\vec{x}) = [\psi_1(\vec{x}), ..., \psi_N(\vec{x})]^T$, $s.t.$ $\{\psi_i(\vec{x}) : X \to \mathbb{R},\ \forall i\}$ and a dataset of sequential observations $\vec{X} = [\vec{x}_1, ..., \vec{x}_M]$, we can generate an approximate Koopman operator, K, to describe the evolution of the dynamical system lifted into observable space

$$\Psi(\vec{x}_{k+1}) = K\Psi(\vec{x}_k) + \epsilon, \tag{2.3}$$

such that $\epsilon \sim \mathcal{N}(\vec{0}, \Sigma)$, where $\mathcal{N}$ is a multivariate Gaussian distribution with mean $\vec{0}$, and variance tensor Σ. The corresponding approximate Koopman operator is given by the solution to the optimization

$$K^* = \underset{K}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{M-1} ||\Psi(\vec{x}_{k+1}) - K\Psi(\vec{x}_k)||^2. \tag{2.4}$$

The unconstrained problem has a closed-form solution; however, recent work has focused on gradient-descent-based solutions to this problem under spectral constraints for the operator [**28**]. The solution to the unconstrained problem is given by

$$K^* = AG^{\dagger}, \tag{2.5}$$

where † denotes the Moore-Penrose pseudoinverse and the individual matrix components are

$$\begin{aligned} G &= G[M] = \frac{1}{M}\sum_{k=1}^{M-1}\Psi(\vec{x}_k)\Psi(\vec{x}_k)^T \\ A &= A[M] = \frac{1}{M}\sum_{k=1}^{M-1}\Psi(\vec{x}_{k+1})\Psi(\vec{x}_k)^T. \end{aligned} \tag{2.6}$$

An additional property of interest that we note based on Eq. 2.6, is that we can generate these matrices incrementally in real-time as we measure our states online [**2, 21**]. The incremental updates to these matrices can be computed via

$$\begin{aligned} G[M+1] &= G[M] + \frac{1}{M+1}(\Psi(\vec{x}_M)\Psi(\vec{x}_M)^T - G[M]) \\ A[M+1] &= A[M] + \frac{1}{M+1}(\Psi(\vec{x}_{M+1})\Psi(\vec{x}_M)^T - A[M]), \end{aligned} \tag{2.7}$$

where we can calculate the updated Koopman operator with

$$K[M+1] = A[M+1](G[M+1])^{\dagger}. \tag{2.8}$$

This formulation does not scale in complexity with the number of measurements since the pseudoinverse computation complexity scales with respect to the number of basis functions considered. The difference equation in Eq. 2.7 describes an implicit cumulative average of all measurements. However, it is possible to perform incremental updates to the Koopman operator using other kinds of difference equations, such as moving average filters, exponential moving average filters, or Kalman filters [**19, 16**].

2.2.2. Koopman Operator for Control

In order to apply the Koopman operator in control systems, we must formulate the operator to account for real-time inputs [**7**]. By considering control inputs as a part of the state vector, we can partition the basis function vector $\Psi(\vec{x}, \vec{u})$ into $\Psi(\vec{x}, \vec{u}) = [\Psi_x(\vec{x})^T, \Psi_u(\vec{x}, \vec{u})^T]^T$, where $\{\Psi_x(\vec{x}) : X \to \mathbb{R}^{N_x}\}$ and $\{\Psi_u(\vec{x}, \vec{u}) : X \times U \to \mathbb{R}^{N_u}\}$ such that $N = N_x + N_u$. Here, U represents the space of control inputs, u. As a result, the corresponding data-driven Koopman operator can be split into submatrices

$$K = \begin{bmatrix} K_x & K_u \\ K_{ux} & K_{uu} \end{bmatrix}. \tag{2.9}$$

Using these submatrices we can formulate the time-evolution of state observables while accounting for the influence of control [**2**]

$$\Psi_x(\vec{x}_{k+1}) = K_x \Psi_x(\vec{x}_k) + K_u \Psi_u(\vec{x}_k, \vec{u}_k). \tag{2.10}$$

Additionally, when considering the evolution of observables that depend on input, the following equation also holds

$$\Psi_u(\vec{x}_{k+1}, \vec{u}_{k+1}) = K_{ux} \Psi_x(\vec{x}_k) + K_{uu} \Psi_u(\vec{x}_k, \vec{u}_k). \tag{2.11}$$

It is worth noting a couple special cases of Eq. 2.10. For the case in which $\Psi_u(\vec{x}, \vec{u})$ is linear in $\vec{u}$ the equation becomes

$$\Psi_x(\vec{x}_{k+1}) = K_x \Psi_x(\vec{x}_k) + \hat{K}_u \vec{u}_k, \tag{2.12}$$

where $\hat{K}_u = K_u \frac{\partial \Psi_u(\vec{x}_k, \vec{u}_k)}{\partial \vec{u}}$. Then for the case in which $\Psi_u(\vec{x}, \vec{u})$ is linear in $\vec{u}$ and does not depend on the state the equation is

$$\Psi_x(\vec{x}_{k+1}) = K_x \Psi_x(\vec{x}_k) + K_u \vec{u}_k. \tag{2.13}$$

This formulation in Eq. 2.13 of the Koopman operator for control systems enables us to easily incorporate it into classical control schemes such as optimal control, or other model-predictive frameworks [**26**]. Alternative data-driven formulations of the Koopman operator for control have been described in the work of [**22**].

As an illustrative example, we consider the case of linear-quadratic (LQ) optimal control. In optimal control, we seek to specify control laws that best drive systems towards a desired goal state with respect to an objective. For a discrete-time LQ control problem with goal state $\vec{x}_d$, we can define an objective function of the following form over the iterate k

$$J = \sum_{k=0}^{\infty} (\Psi_x(\vec{x}_k) - \Psi_x(\vec{x}_d))^T Q (\Psi_x(\vec{x}_k) - \Psi_x(\vec{x}_d)) + \vec{u}_k^T R \vec{u}_k, \tag{2.14}$$

where Q and R are positive semi-definite square matrices of appropriate dimensions that specify weights on system states and control effort, respectively. Deriving an optimal solution to this LQ problem leads to a linear feedback law

$$\vec{u}_{LQ_K} = -F_{LQ_K}(\Psi_x(\vec{x}) - \Psi_x(\vec{x}_d)) \tag{2.15}$$

such that the optimal feedback gain $\vec{F}_{LQ_K}$ is

$$F_{LQ_K} = (R + K_u^T P K_u)^{-1} K_u^T P K_x, \tag{2.16}$$

where P is the solution to the discrete-time algebraic Ricatti equation. Through this process, we can solve optimal control problems with data-driven Koopman models [**7, 22**].

2.2.3. Koopman Operator for Hybrid Dynamical Systems

Recent work presented in [**1, 4**] introduced an extension of the numerical Koopman framework to hybrid dynamical systems. Hybrid dynamical systems experience discontinuities in their dynamics where depending on the system state the system can exhibit different dynamics. Hybrid systems are an example of complex multi-modal systems where the differential equations describing the behavior of the system explicitly include discretized behaviors over the state-space. While the complex systems we consider in this work also exhibit multi-modality, hybrid systems represent a small subset of the set of such systems.

Hybrid dynamics are most commonly experienced in robotic systems as the result of impacts and intermittent contacts in legged locomotive systems. These contacts lead to discontinuities and non-smoothness in the state-space trajectories of the system, and hence must be modeled piece-wise to account for such jumps in the dynamics. An indicator function determines which hybrid mode will evolve system at its current state. We can formulate these dynamics as

$$F(\vec{x}) = \begin{cases} F_1(\vec{x}), & \text{if } \Phi(\vec{x}) = 1 \\ \vdots & \vdots \\ F_i(\vec{x}), & \text{if } \Phi(\vec{x}) = i \\ \vdots & \vdots \\ F_B(\vec{x}), & \text{otherwise} \end{cases} \tag{2.17}$$

where each $F_i(\vec{x})$ represents the dynamics of states $\vec{x} \in X$ at hybrid mode $\Phi(\vec{x}) = i$. Additionally, at each hybrid mode boundary there is a discontinuous map between modes. Assuming that $\Phi(\vec{x})$ is known *a priori*, it is possible to compute a Koopman operator K_i to represent each hybrid mode. We can specify a hybrid Koopman operator using

$$\overline{K}, \overline{\Psi}(\vec{x}) = \begin{cases} K_1, \Psi_1(\vec{x}), & \text{if } \Phi(\vec{x}) = 1 \\ \vdots & \vdots \\ K_i, \Psi_i(\vec{x}), & \text{if } \Phi(\vec{x}) = i \\ \vdots & \vdots \\ K_B, \Psi_B(\vec{x}), & \text{otherwise} \end{cases} \tag{2.18}$$

where we characterize the discontinuous jumps between hybrid modes by collecting data at the mode boundaries and using $\Psi_i^*(\vec{x}_k^+) = \Psi_i^*(\vec{x}_k^-)K_i^*$. Given that these jumps in state often take place over very small time scales, it may be difficult to gather enough data to provide a robust Koopman operator for that mapping. Active excitation methods might be necessary in order to properly characterize transitions. Alternative treatments of the Koopman operator have looked into modeling and analyzing switched [**35**], hybrid [**18**], and distributed systems [**20**].

2.2.4. Comparison to Alternative Methods

To illustrate the predictive capacity of the EDMD approximation of the Koopman operator, we make use of it to predict the unforced nonlinear dynamics of a double pendulum from a variety of initial conditions. Moreover, we compare the Koopman operator

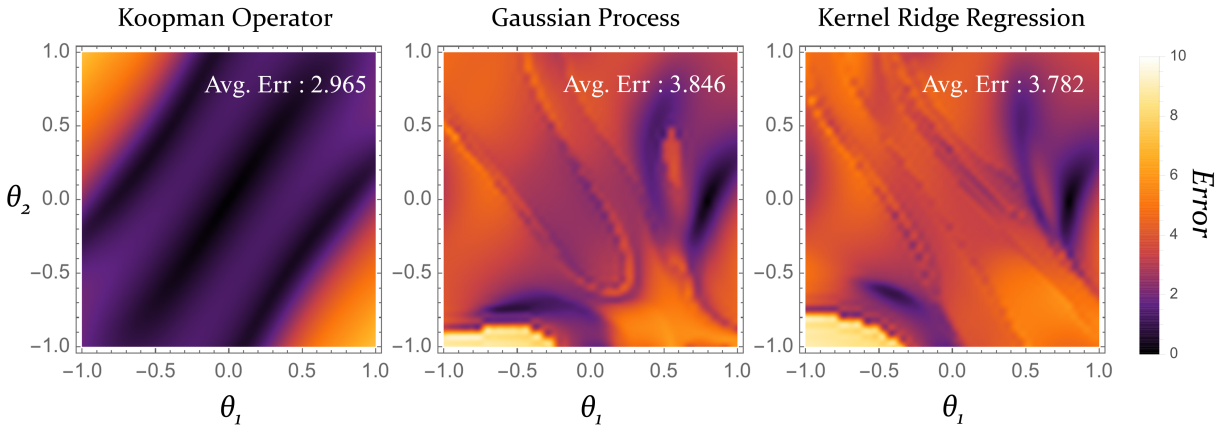


Figure 2.2. **Performance comparison between EDMD Koopman operators, Gaussian process regression, and kernel ridge regression.** Each model was trained on the same dataset collected from a $4s$ trajectory of the free dynamics of a double pendulum system with initial condition $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = (0.8, 0, 0, 0)$. Then, we calculated the integrated mean squared error (MSE) over a $3s$ predictive horizon from each data-driven model over the entire $\{(\theta_1, \theta_2) : [-1, 1] \times [-1, 1]\}$ domain. Standard implementations of each model were used so as to compare performance under typical usage.

to alternative machine learning techniques, namely Gaussian processes and kernel ridge regression.

To compare between these methods, we make use of standard, easily available implementations rather than state-of-the-art. To this end, we generate three independent data-driven models of the dynamics of a double pendulum system—one with Koopman operators, one with Gaussian processes, and one with kernel ridge regression. We collected a training dataset consisting of a single $4s$ trajectory taken from the double pendulum free dynamics sampled at $100Hz$ with an initial condition of $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = (0.8, 0, 0, 0)$. We use the training dataset to instantiate a Koopman operator, kernel ridge regression model, and a Gaussian process model. The Koopman operator model is calculated using a set of

second-order polynomial basis functions of the double pendulum system states, while both the kernel ridge regression model and the Gaussian process use the radial basis function kernel (which is the default choice in many common software packages). The regularization and variance parameters for the Gaussian process and kernel ridge regression models were selected by the Scikit-Learn software package [**34**] via Bayesian hyperparameter optimization [**39**]. Each model is then used to predict double pendulum trajectories for a time horizon of $3s$ from each (θ_1, θ_2) initial condition over the $\{(\theta_1, \theta_2) : [-1, 1] \times [-1, 1]\}$ domain, with zero initial velocity. Then, we calculate the integrated mean squared error (MSE) between each model's prediction and the true dynamics over the entire predictive horizon for all initial conditions.

Figure 2.2 depicts the results of the comparison. The Koopman operator model prediction has the lowest average MSE of the three models. We observe that both the Gaussian process model and kernel ridge regression model prediction errors are lowest at the initial conditions of the training trajectory $(\theta_1, \theta_2) = (0.8, 0)$ as a result of regularization, while the Koopman operator model generalizes more easily over the domain.

Throughout this chapter we have motivated the Koopman operator and its numerical approximation as a convenient tool for prediction of dynamical systems. The example above and comparison alternative models also depicts the Koopman operator an effective linear representation of the underlying nonlinear dynamics. Equipped with Koopman operators as a tool, we may now use them towards developing techniques tailored for analyzing complex multi-modal dynamical systems.

CHAPTER 3

Dynamical System Segmentation

Now that we have described Koopman operators as an effective tool for understanding and predicting dynamical systems, we return to our problem of analyzing complex multi-modal dynamical systems. While finite-dimensional Koopman operators can be useful in describing nonlinear dynamics, they often have limited predictive capacity outside of the state-space domain of observed trajectories. In this sense, this is a limitation of finite-dimensional approximations of the Koopman operator relative to the theoretical globally predictive Koopman operator. While in systems with simple dynamics the locality of the operator may not pose substantial hurdles, in complex systems this can lead to substantial predictive errors.

An example problem domain where substantial predictive errors with a single Koopman operator would arise is in the prediction of hybrid dynamical systems. Here, the most natural Koopman representation of the system would be comprised of multiple individual operators, as was described in Chapter 2.2.3. However, in that chapter we made the assumption that we knew the number of system modes *a priori*, and that we also knew the function describing the transitions between such modes. In general, such fine information about the system is often impossible to acquire from first-principles outside of toy example problems. Thus, we are interested in developing data-driven tools for representing multi-modal systems from observations of their behavior.

To this end, we developed and applied Dynamical System Segmentation (DSS) in [**5, 37, 23**]. DSS is a nonparametric, unsupervised learning algorithm that discovers and segments system behaviors from observations of state-space trajectories. The algorithm synthesizes a set of distinct, yet interrelated, system behaviors that capture the multi-modal dynamics of the system without any *a priori* knowledge of the system. We refer to the set of models synthesized by DSS as an alphabet comprised of symbols—each a Koopman operator—describing the coarse-grained dynamics of the system. DSS is comprised of three primary subroutines: (i) computation of sequential Koopman operators, (ii) unsupervised density-based clustering over the space of operators, (iii) and self-supervised learning of a projection onto the system state-space.

Given a dataset $\vec{X} = [\vec{x}_1, ..., \vec{x}_M]$ consisting of state trajectories from a dynamical system (assumed to be of the same form as in Eq. 2.1), and a set of vector-valued observables $\{\Psi(\vec{x}) : X \to \mathbb{R}^N$, we can evaluate the basis functions onto the dataset X in order to generate a transformed dataset $\Psi_X = [\Psi(\vec{x}_1), ..., \Psi(\vec{x}_M)]^T$. We then split the transformed dataset Ψ_X into a set of W (potentially overlapping) rectangular windows, and calculate a Koopman operator for each, thereby generating a set of symbols $\mathbb{K} = \{K_1, ..., K_W\}$.

We are interested in creating a minimal alphabet of Koopman operators with which to span all observed system behaviors. Unsupervised learning methods such as clustering algorithms that specialize in the identification of classes within datasets are well-suited for this task. By considering each $\mathbb{R}^{N \times N}$ Koopman operator as a point in $\mathbb{R}^{N^2}$ space, we can construct a feature array and partition the set $\mathbb{K}$ into subsets using a clustering algorithm. In particular, we use Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), which is a nonparametric clustering algorithm that performs well in

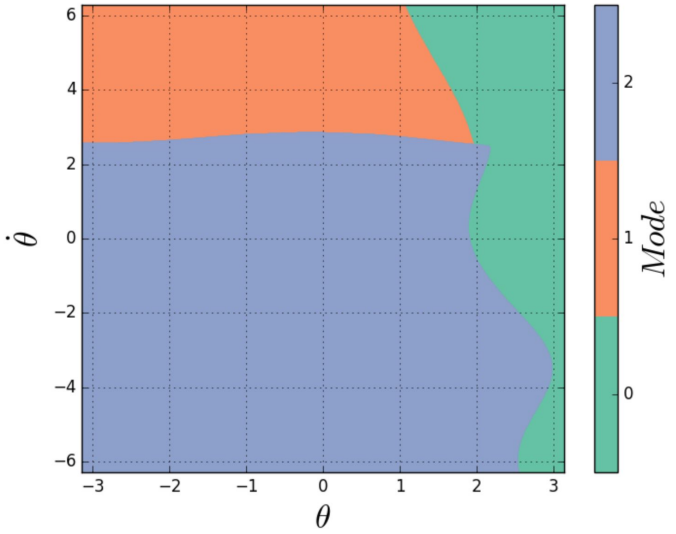


Figure 3.1. **Example phase-space partitions generated by DSS SVM in a cart-pendulum system.** To illustrate the partitions generated for a real dynamical system, we apply DSS to sample trajectories from a cart-pendulum system whose state-space coordinates are $(\theta, \dot{\theta}, x_c, \dot{x}_c)$, see Chapter 4.2 for a more detailed description of the system. DSS generates a 3-mode abstraction that partitions the phase-space of the system. Here, we show the $(x_c, \dot{x}_c) = (1, -1)$ cross-section of the complete system phase-space.

high-dimensional settings subject to noise [**9**]. The algorithm groups the operators into B classes $\{C_1, ..., C_B\}$ using only the threshold of points required to form a cluster as a parameter. We compose a set $\overline{\mathbb{K}} = \{\overline{K}_1, ..., \overline{K}_B\}$ of class exemplars by taking a weighted-average of all $K_i \in C_j,\ \forall j \in \{1, ..., B\}$, according to the class-membership probability $p(K_i | K_i \in C_j)$. The class-membership probability function is provided by the HDBSCAN software package [**30**].

Although we have created a minimal alphabet $\overline{\mathbb{K}}$ of system behaviors, it is of interest to project these behaviors onto the state-space manifold from this abstract operator space. We label all points in the transformed dataset Ψ_X with a label $l \in \{0, ..., B\}$ according to the class label of the Koopman operator each point corresponds to. Then, we train a

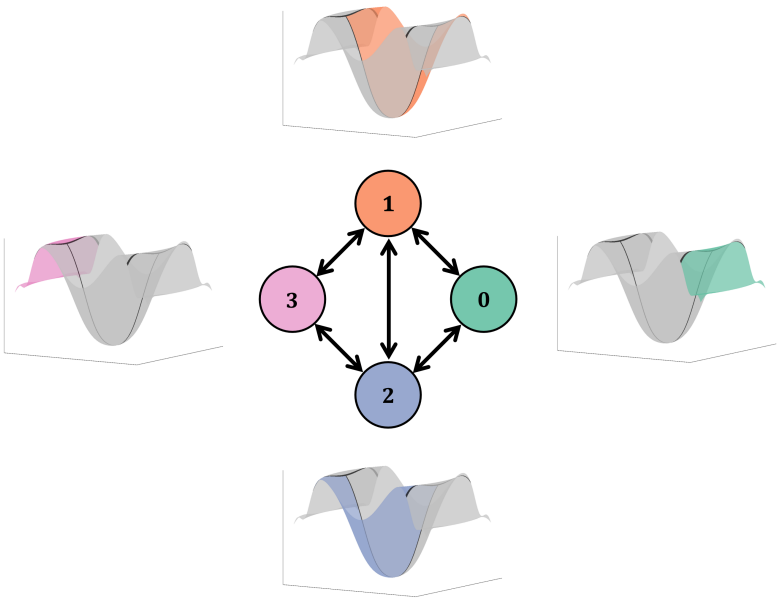


Figure 3.2. **Schematic of the output of Dynamical System Segmentation (DSS).** DSS is a nonparametric system identification algorithm that synthesizes low-dimensional representations of dynamical systems. DSS takes in data from in the form of a multi-dimensional time-series and segments it into overlapping contiguous subsets. The algorithm computes local estimates of the underlying system dynamics over these subsets and represents them as finite-dimensional Koopman operators. These operators are then compared by a clustering algorithm (HDBSCAN) to extract a set of models that best represent the distinct behaviors exhibited by the underlying dynamics. Finally, DSS constructs a mapping from the space of system behaviors onto the underlying state-space using a support vector machine (SVM). The output of DSS is then a graphical model where each node describes the dynamics of the system over some domain of the underlying state-space manifold.

support vector machine (SVM) classifier, $\Phi(\psi(\vec{x}))$ to project the class labels onto the state-space manifold, thereby generating partitions of the state-space [**34**]. In this setting, the SVM implementation is self-supervised because the algorithm itself generates the labels for training. An example of the SVM partitions onto the state-space of a dynamical system is shown in Figure 3.1.

Algorithm 1 Dynamical System Segmentation (DSS)

Input: Sequential dataset $\vec{X} = [\vec{x}_1, ..., \vec{x}_M]$ consisting of realizations of a dynamical system defined over some state-space X, and basis functions $\{\Psi(\vec{x})|\ \Psi : X \to \mathbb{R}^N\}$

Procedure:

1: Transform the $\vec{X}$ dataset into $\Psi_X = [\Psi(\vec{x}_1), ..., \Psi(\vec{x}_M)]^T$ using selected basis
2: Split Ψ_X into a set of W (possibly disjoint) subsets, themselves comprised of sequential data
3: Calculate a Koopman operator K_i for each subset of Ψ_X using least-squares optimization to construct the set $\mathbb{S} = \{K_1, ..., K_W\}$
4: Construct a feature array $\mathbb{S}_{flat}$ by flattening all $K_i \in \mathbb{R}^{N \times N}$ in $\mathbb{K}$ into points in $\mathbb{R}^{N^2}$ and appending them together
5: Apply nonparametric clustering on $\mathbb{S}_{flat}$ and label all K_i's from one of B discerned classes $\{C_1, ..., C_B\}$
6: Construct a set $\mathbb{K} = \{\overline{K}_1, ..., \overline{K}_B\}$ of class exemplars by taking a weighted-average of all $K_i \in C_j,\ \forall j \in \{1, ..., B\}$, according to $p(K_i|K_i \in C_j)$
7: Label all points in Ψ_X with the label $l \in \{1, ..., B\}$ of the Koopman operator they were used in training
8: Train an SVM $\Phi(\Psi(\vec{x}))$ on the labeled points
9: Construct set of observed transitions $\mathbb{E}$ by tracking all sequential labels in the dataset
10: Construct a graph $\mathcal{G} = (\mathbb{K},\ \mathbb{E})$ encoding system behaviors as well as transition probabilities between each one

Return: Probabilistic graphical model $\mathcal{G}$, and trained SVM $\Phi(\Psi(\vec{x}))$

The output of DSS is best represented by a graphical model. We can define a graph $\mathcal{G} = (\overline{\mathbb{K}},\ \mathbb{E})$ where the node set $\overline{\mathbb{K}}$ contains the exemplar Koopman operators synthesized by the clustering procedure. The set of edges $\mathbb{E}$ and associated weights are determined by directly observing the sequences of class labels in the dataset, and tracking the frequencies of transitions. The transition probabilities estimated from this procedure can either be interpreted as state-space adjacency in a deterministic system (by taking $\lceil w \rceil$ for each edge weight w), or as estimates of the true transition probabilities between nodes if the system is stochastic. Figure 3.2 illustrates how the output graphical model from DSS relates to the underlying manifold of the dynamical system. Each node in the graph represents a distinct dynamical system over its respective partition of the state-space manifold. By

traversing the graph symbolically from one node to another, traversal of the state-space manifold is implied. The DSS algorithm is summarized in Algorithm 10.

DSS as an algorithm enables detailed analyses of the dynamics of complex nonlinear dynamical systems from data. The graphical model output by DSS is an informative object itself in addition to its predictive capabilities. In the following chapters, we illustrate the utility of the DSS graphical model as a function of the structure of the graph, the information encoded in the node and transition frequencies, as well as the predictive capacity of the Koopman models themselves.

CHAPTER 4

Data-Driven Symbolic Abstractions for Motion Analysis

Throughout this chapter and the next we proceed to demonstrate and motivate different uses and applications of DSS. In this chapter, we focus on applications of DSS for constructing models of motion, and then assessing the quality of said motion. First, we investigate the information directly encoded in the structure of the DSS graphical model, and apply it in context of gait segmentation. Then, we use the relative frequencies of DSS-derived symbols observed in the trajectories of agents to assess the quality of movement in the context of a dynamic task. Many of the results in this chapter were presented in [**23**] and [**5**].

4.1. Gait Segmentation

Human gait has been an active clinical and algorithmic subject of study for many years. While the clinical literature is largely settled in relation to healthy human gait, understanding the role impairments play in gait is less so. Because different impairments may lead to unique modifications to human gait, it has been challenging to develop models of impaired gait in the absence of efficient data-driven techniques to analyze movement. In this domain, it is particularly important that the algorithms developed to serve this purpose be unsupervised because there is no medical consensus with which to generate labels of impaired gaits. We suggest that progress in this domain may prove useful

towards developing assistive platforms for walking, such as exoskeletons, as well as clinical intervention strategies.

A particularly active area in this field is the study of gait partitioning and gait phase identification [**42**]. Through the use of a variety of algorithms, one can often obtain models that predict and distinguish in real-time between the clinically-specified phases of healthy human gait. Many examples of this can be found in the literature as implemented across different machine learning techniques, such as neural networks [**48, 11**], hidden Markov models [**43**], or Gaussian mixture models [**12**]. However, these models are generated from a rich suite of sensory information, such as ground contact forces, joint positions, inertial data, or muscle signals. Particularly, most of these studies rely heavily on ankle data for phase identification, which is often not present in lower-limb assistive devices, such as exoskeletons. The greater the number of sensors in the device, the greater the cost, leading to a decrease in accessibility for end-users.

Moreover, the fact that these algorithms require labeled training datasets also limits their applicability in scenarios with abnormal gaits. Work in this area often requires the practitioner to make assumptions about the number of gait phases expected, phase transition times, or phase durations, limiting these algorithms' ability to adequately partition abnormal gait. All of this amounts to a rebuke of traditional machine learning techniques in the domain of abnormal gait analysis.

Here, we propose DSS as a tool for developing data-efficient, unsupervised models of human gait, and demonstrate it by synthesizing models that match clinical descriptions of healthy gait in an experimental lower-limb exoskeleton. Particularly, we apply DSS to the gaits generated by a healthy human subject in an Ekso Bionics® exoskeleton, and

Figure 4.1. **Ekso Bionics EksoGT™lower-limb exoskeleton.** The exoskeleton was primarily used for data collection from two sets of sensors: hip and knee encoders. We generated a kinematics-based dynamical model using DSS, and then validated its state-space partitions using foot-mounted pressure sensors at the heel and toe.

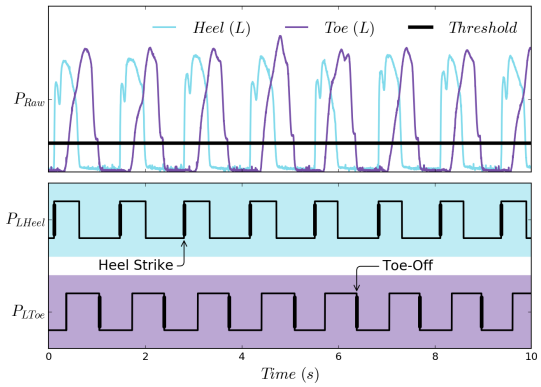
demonstrate accurate gait segmentation using data only from knee and hip joints. We validate our gait partitions by comparing it to pressure data from heel and toe contact sensors, and reliably predict heel-strike and toe-off events without use of impact sensors in the model. We do not pre-label transitions in our training data or pre-define the expected number of gait phases, which allows us to identify a range of recurring movement patterns in the gait cycle in an unsupervised manner, showing promise for meaningful partitioning of abnormal walking gaits.

We collected gait data using EksoGT™—a robotic lower-limb exoskeleton from Ekso Bionics, Richmond, CA, USA, visible in Figure 4.1. When not actively in assistance mode, the device offers freedom to move in the sagittal plane, and to a limited extent in the

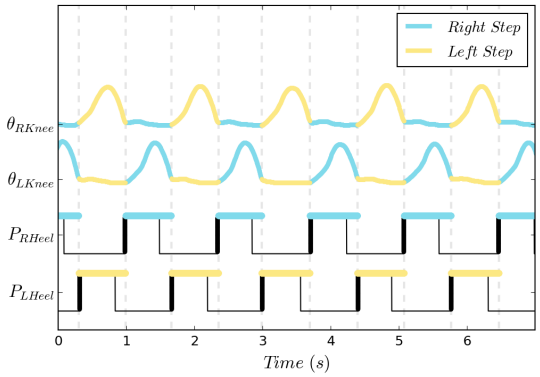
frontal plane. When active, it provides assistance solely in the sagittal plane. Both knee and hip joints can be used for assistance, where angular position and angular velocity can be measured at $500Hz$ by encoders in all four joints. The ankle joints are passive and no sensory data is available.

For the purposes of this study, two minutes of data were collected of straight-line flat-ground walking from one healthy subject with previous experience walking in the exoskeleton (an Ekso Bionics employee). No assistance or resistance was provided to the wearer through motor activity; any perceptible resistance was passive from the mechanical structure of the device. A total of sixteen variables were recorded. Twelve of them (right/left knee angles, right/left knee angular velocities, right/left hip angles, right/left hip angular velocities, right/left knee motor currents, and right/left hip motor currents) were used for analysis. The additional four variables (right/left toe sensors and right/left heel sensors) were excluded from the DSS analysis, and used solely for validation and verification of DSS-generated gait partitions.

Gait cycles are generally defined from one foot strike to the subsequent foot strike on the same side. Clinically, they are often partitioned separately for each leg based on functionally critical events for that leg [**10**]. For this study, we are interested in generating gait partitions that capture critical changes in the behavior of both legs. We did not impose leg symmetry as a constraint, but we chose states and basis functions symmetrically for right and left legs to allow the algorithm to remain equally sensitive to recurring patterns on both sides. Moreover, in the instantiated models we look for gait events representing contact with the ground, specifically heel-strike and toe-off, because these impact events indicate transitions between dynamically distinct modes that are



(a) Thresholded pressure signals provide a ground truth for heel-strike and toe-off events in gait.



(b) The learned DSS model, exclusively generated from hip and knee trajectory information, correctly determines heel-strike events.

Figure 4.2. **DSS 2-mode symbolic abstraction predicts heel-strike events.** Using a DSS-generated abstraction we are able to correctly predict heel-strike events with 100% accuracy over a 30s horizon within 11ms off of the ground truth detection.

important for generating control in a robotic assistive device. As a tertiary objective, we are interested in sub-dividing leg swings, because the majority of active assistance during walking takes place during the swing phase.

In order to validate DSS-generated gait partitions, we utilize the foot-mounted pressure sensors at the heel and toe. We collect analog signals from the pressure sensors and threshold them to obtain binary readings of whether the heel and toe are in contact with the ground. These processed sensor readings allow us to directly record heel-strikes and toe-offs, establishing a notion of ground truth for transitions between stance and swing phases, as demonstrated in Figure 4.2(a). As a result, for each gait decomposition, we can find mode transitions that correspond to these ground-contact events, and measure the offset between the prediction of the event and the ground truth from the pressure sensors. The offset measurement is dependent on DSS reliably recognizing specific mode

transitions, and thus conveys both the precision and accuracy of event detection. We report the offset, or latency, as our validation statistic for each of the obtained gait decompositions.

Using DSS, we can segment gait into a range of decompositions purely from kinematic information, where each decomposition is determined by a set of distinct transition conditions that are recurrent throughout the gait cycle. Here, we report on two distinct gait partitions obtained from the same dataset synthesized by DSS with different sets of basis functions. For each gait partition, we expand the state space through quadratic, cubic, and/or trigonometric functions of the original 12 states. Depending on the choice of basis functions, and consequently the recognized transition events, we segment gaits into 2 and 6 phases, where in each case the phase transitions correspond to easily interpretable gait events.

We begin with the simpler and more easily verifiable gait segmentation: a 2-phase segmentation. In the 2-phase segmentation, we solely identify transitions that correspond to heel-strikes with each leg. This way, the gait cycle gets simply split into right and left steps, as shown in Figure 4.2(b). We verify the accuracy of the DSS heel-strike event detection against pressure sensor signal, and observe no misclassifications with an average latency of $11ms \pm 8.5ms$. We note that this latency falls well within the range of human reaction times, so a robotic device with access to such a segmentation at this time scale could provide assistance based on it.

Finally, we report a 6-phase gait partition, which we visualize it in Figure 4.3. This gait partition allows us to identify both heel-strikes and toe-offs, splitting the gait into right/left swing and stance. In addition, it divides swing into two phases: initial and

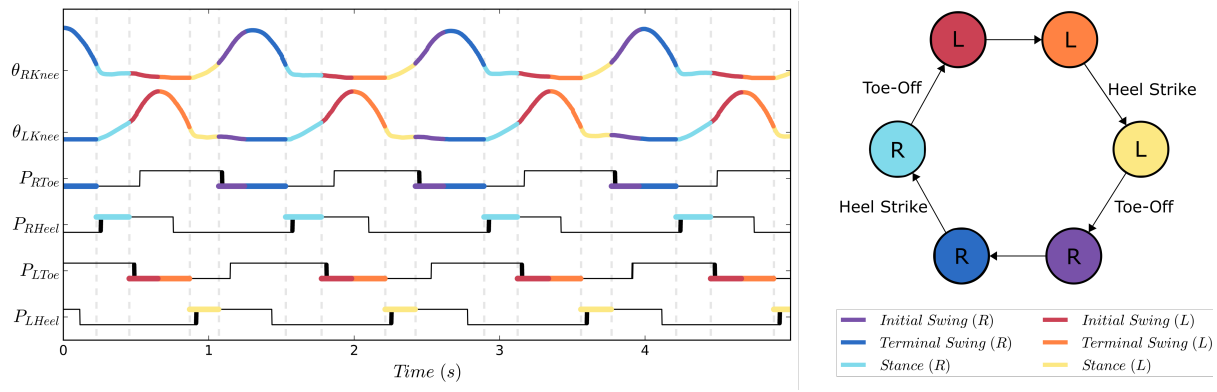


Figure 4.3. **DSS 6-mode decomposition of human gait.** By varying DSS parameters we are able to recover a complete clinical model of human gait from data. Notably this is done in the absence of any sensors corresponding to the ankle joint or foot pressure information. Here, we can resolve events such as heel-strikes, toe-offs, and swings for both legs purely from hip and knee kinematic data. We achieve an event classification accuracy of 100% within $36ms$ of each event on average when tested against ground truth measurements from foot-mounted pressure sensors over a $30s$ horizon.

terminal swing. Transition from initial into terminal swing corresponds to the clinically recognized foot clearance event—when the swing leg passes the stance leg—and near maximal knee flexion. This segmentation could also be of great use for the development of assistance platforms, as well as clinical interventions, because it allows one to detect the start of swing. For example, in the setting of using robotic rehabilitation to aid a subject in independently initiating steps, we might want them to complete pre-swing prior to receiving robotic assistance and then wait for a toe-off event to apply control. As with the previous segmentation, we verify the toe-offs and heel-strikes against pressure sensors, detecting no misclassifications and an average offset of $36ms \pm 9.6ms$—also well within the range of recorded human reaction times.

The gait partitions presented here are certainly not exhaustive, and are only meant to illustrate the capabilities of the algorithm. Alternative gait segmentations can be obtained, depending on what gait phase transitions are important to detect in a particular application. Thus, by applying DSS onto walking data we have demonstrated how the graphical model generated by DSS can encode a clinically-correct representation of healthy human gait in a completely unsupervised manner (as shown in Figure 4.3).

4.2. Motion Quality Assessment

In the previous section of the chapter, we demonstrated that the graphical models generated by DSS can encode information about the state trajectories they are derived from. While it is difficult to arrive at closed-form hybrid dynamical descriptions capturing the complexities of human gait, we still had a notion of ground truth from clinical models of gait to compare against our DSS graph. In general we do not expect this to be the case, so we are interested in applying and validating DSS in complex systems that do not possess the sharp mode-boundary delineations that hybrid dynamical systems have.

In this section, we apply DSS to the analysis of a joint human-machine system without obvious mode boundaries, and use it towards assessing the quality of movements in the context of a task. To this end, we take an information-theoretic viewpoint of human movement where we presuppose that motions are information-carrying signals comprised of symbols drawn from an alphabet of motion primitives. While such an assumption may seem unwarranted, certain theories of neural motor control support it.

In [**32**], the authors propose that human motions are the result of the composition of a finite set of premotor signals emanating from the spinal cord. As a consequence,

the neurological feasibility of motion decomposition supports the existence of movement primitives, also referred to as *movemes* [**46**]. In this setting, movemes are fundamental units of motion, and derive their name from their linguistic analogue: phonemes. Thus, all smooth human motions may be comprised of symbolic sequences drawn from an alphabet of movemes. Movemes motivate the application of information-theoretic tools in human motion analysis, because they provide evidence of finite structure in otherwise continuous motion signals. Moreover, the neurological existence of movemes indicates that under some choice of representation human motion can be discretized without loss of information.

In the human motion analysis literature, movemes are often mathematically described using linear causal dynamical systems [**46, 47**], or autoregressive models [**17**]. Most motor signal segmentation methods demand prior specification of the moveme alphabet either through direct template matching or manual labeling of training data, which limits their use in exploratory analyses where the structure of the alphabet may not be known *a priori*. Techniques in symbolic dynamic filtering can generate symbolic alphabets given an expected alphabet size by creating partitions of the state-space using methods such as maximum entropy partitioning [**27**]. Additionally, state-space partition techniques can be applied to nonlinear transformations of the space via methods such as wavelet transforms [**36**]. However, the symbols synthesized by these techniques correspond to quasi-static abstractions, and hence do not capture the dynamic nature of movemes.

In this setting, we propose DSS as an algorithmic method for generating moveme alphabets from data without any prior system knowledge. We posit that the relationships between symbols in the alphabet specify a language of motion, where symbols and

their relationships are given by the nodes and edges (with respective weights) in a DSS graph. Just as in spoken languages, symbols (*i.e.*, letters) have a natural frequency distribution that gives us information about the kinds of words that are typically formed in that language. Here, we suggest that the symbols extracted by DSS and their relative frequencies—though coarse-grained—encode substantial information about the underlying movements. Particularly, we will analyze the amount of information that movements encode in the context of an underlying task.

To quantify the task-specific information content of movements, we must first define a relevant information metric. In the context of information theory, information is an ordered sequence of symbols drawn from an alphabet generated by a source [**40**]. We define our notion of task information with respect to a given source, such that the source generates sequences of symbols comprising realizations of the task. Any physical system or agent attempting a given task is what we term a task-specific information source. Throughout this study, we quantify task information from the relative symbol frequencies distribution from a task-specific information source. Thus, we define *task embodiment* as a measure of task information encoded in an agent's trajectories by calculating the relative entropy of their relative symbol frequency distribution, with respect to a reference symbol distribution that "optimally" embodies the task.

Given trajectory demonstrations from an agent that optimally embody a task, we can construct a DSS model of said agent and generate a corresponding graph, $\mathbb{G}_{opt}$. We refer to the relative frequency distribution over the symbols, $\overline{\mathbb{K}}$, that we observe in the optimal agent as p. However, we note that we can use the trained SVM classifier $\Phi_{opt}(\psi(\vec{x}))$ to identify the behaviors of the optimal agent in other "non-optimal" agents. Moreover,

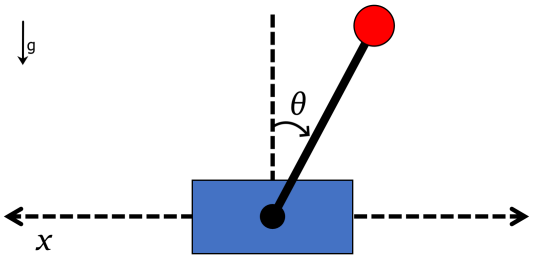


Figure 4.4. **Cart-pendulum system inversion task used in human subjects study (n=53).** In this work, participants use a robotic interface (NACT-3D) to interact with a cart-pendulum system where the goal is to invert the pendulum to its upright position and hold it in that configuration as long as possible. This is a dynamic task requiring fine motor control, as well as brusque movement generation.

by tracking the relative frequencies of such symbols in the non-optimal agent, we can empirically calculate a discrete distribution q. In this context, we now mathematically define task embodiment as the Kullback-Leibler divergence (D_{KL}) [**6**] between the symbol distribution of the optimal and non-optimal agents:

$$D_{KL}(p||q) = -\sum_{i=1}^{B} p(\overline{K}_i) log\big(\frac{q(\overline{K}_i)}{p(\overline{K}_i)}\big). \tag{4.1}$$

To test the efficacy with which DSS can coarse-grain and compress the movement patterns of complex dynamical systems, we applied the DSS-enabled task-embodiment formalism to assess the motions of multiple agents attempting a challenging dynamic task. Specifically, the proposed assessment of task embodiment was applied to data collected from human subjects performing a cart-pendulum inversion task[1].

[1]The authors utilized de-identified data from a study approved by the Northwestern Institutional Review Board.

The dynamics of the cart-pendulum (pictured in Figure 4.4) are given by the following nonlinear equations of motion

$$\dot{\vec{x}} = f(\vec{x}, u) = \begin{bmatrix} \dot{\theta} \\ \frac{g}{l}\sin(\theta) + u\cos(\theta) - \frac{b}{ml^2}\dot{\theta} \\ \dot{x}_c \\ u \end{bmatrix} \tag{4.2}$$

where the system state is given by $\vec{x} = [\theta, \dot{\theta}, x_c, \dot{x}_c]^T$, and the scalar control input u represents accelerations along the axis of movement of the cart. Constants g, l, m, and b represent gravitational acceleration, pendulum length, mass, and viscous damping factor, respectively. The participants interacted with the cart-pendulum dynamics through an admittance-controlled haptic robot (NACT-3D, similar to the system described in [**41**] and [**14**]) that served as an interface for the users to provide input accelerations to the cart. Through the NACT-3D as an interface, we were able to record the system trajectories as a function of user input at sampling frequency of $60Hz$.

In this experimental protocol, all subjects ($n = 53$) were instructed to attempt to invert the virtual cart-pendulum with the goal of spending as much time as possible in the upright unstable equilibrium during the course of a thirty second trial. Subjects repeated this task for 30 trials in each of two sessions. Forty subjects completed this task while receiving filter-based robotic assistance in one session and without assistance in the following session. The order in which the subjects received assistance was counterbalanced to account for learning effects. An additional thirteen subjects were placed in a control group which completed both sessions without assistance.

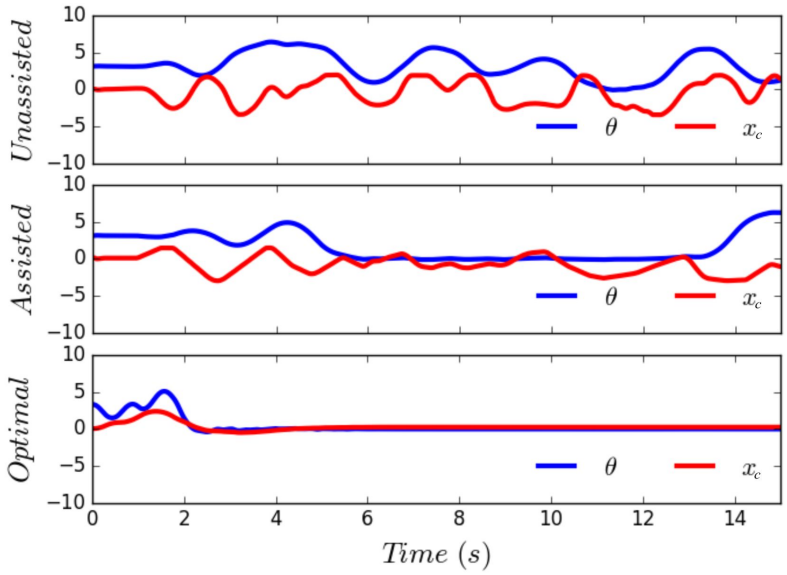


Figure 4.5. **Sample cart-pendulum inversion task executions.** Sample trajectories from different types of task executions. Here, we note qualitative differences between unassisted and assisted trajectories from subject 16, which are then both distinct from an execution generated by an optimal controller.

The filter-based assistance algorithm that was used was first proposed in [**45**] for a system with pure noise inputs, and was later adapted for user input in [**15**]. The assistance physically filters the user's inputs—accelerations in this case—such that their actions are always in the direction of an optimal control policy calculated in real time. Figure 4.5 depicts the effect of assistance on the state trajectories of a representative subject, and includes a trajectory generated by the optimal controller for comparison. In the assisted trajectory, the subject reaches the goal state of $\theta = 0$ (*i.e.*, the unstable equilibrium of the pendulum), and is able to balance the pendulum starting around $t = 5s$. At this point, the assistance restricts the user's input motion $x_c(t)$ such that the inverted state is maintained until $t = 13s$. However, the same subject is unable to maintain the inverted configuration without assistance.

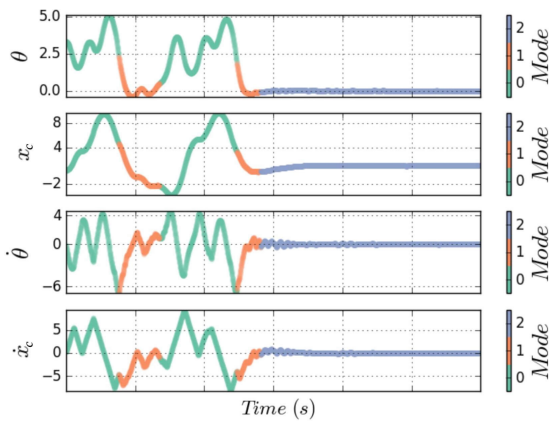
In [**24**], the filter-based assistance algorithm was applied to the virtual cart-pendulum inversion task on the NACT-3D that is described here. In that context, the authors were qualifying the effects of the algorithm on user performance and long-term learning. In this work, the authors found in a variety of metrics that the assistance algorithm significantly improved user performance and learning. Taking the fact that robotic assistance unilaterally resulted in performance improvements for participants, for the purposes of the analysis in this chapter we limit our scope to analyzing differences in the movements of unassisted vs. assisted trajectories with our task embodiment metric in Eq. 4.1.

To generate a reference alphabet of optimal behaviors for the task, we synthesize a dataset representative of an optimal user by using an optimal controller. Data from the expert is segmented by applying DSS in order to generate a graphical model $\mathbb{G}_{opt}$, and a set of exemplar behaviors to track. The set of basis functions utilized to achieve this were

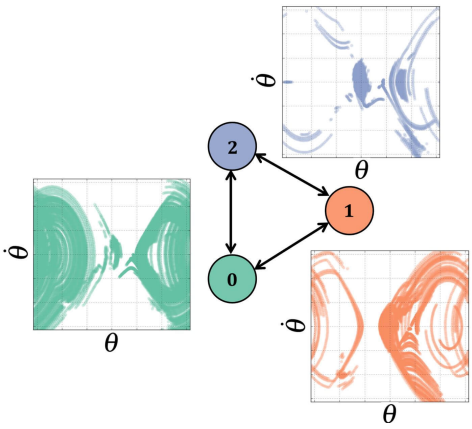
$$\Psi(\vec{x}) = [\theta,\ \dot{\theta},\ x_c,\ \dot{x}_c,\ u,\ u\ cos(\theta),\ u\ cos(\dot{\theta}),\ |u_{sat}|cos^2\big(\frac{u\pi}{|u_{sat}|}\big),\ \dot{x}_c^2,\ 1], \tag{4.3}$$

where $|u_{sat}|$ is the optimal controller's saturation limit on the control effort. The basis functions were selected from the set of linear combinations of second order polynomial and sinusoidal functions. Since clustering occurs in $\mathbb{R}^{N^2}$ space, where N is the number of basis functions, we chose a low-dimensional set ($N = 10$) of representative basis functions in Eq. 4.3 from the larger set of linear combinations of polynomial and sinusoidal functions. This dimensionality reduction can be achieved via multiple methods, such as principal component analysis [**6**].

In Figure 4.6, we depict the segmentation generated from the exemplar trials. We qualitatively describe the identified modes 0, 1 and 2 as energy pumping and swing-up, energy



(a) Time-domain segmentation of a selected optimal control solution of the pendulum inversion task. Mode 0 corresponds to energy pumping and swing-up, mode 1 corresponds to energy removal and slow-down, and mode 2 corresponds to stabilization.



(b) Graphical model resulting from the segmentation of a dataset of 30 optimal control solutions to the cart-pendulum inversion task. The set of segmented behaviors are shown projected onto the system's phase portrait over the domain $\{(\theta, \dot{\theta}) : (-\pi, \pi) \times (-2\pi, 2\pi)\}$.

Figure 4.6. **Data-driven identification of exemplar behaviors.** Through the use of DSS we recover a graphical model describing the segmentation of optimal executions of the cart-pendulum inversion task.

removal and slow-down, and stabilization, respectively. Intuitively, these modes represent a coarse set of strategies that an expert user should exhibit in succeeding at the task. The distribution of behaviors found in the optimal user was $p = [0.2437,\ 0.1275,\ 0.6288]$.

The human subjects data was analyzed by using the trained SVM $\Phi_{opt}(\Psi(\vec{x}))$ to detect the optimal behaviors in the movements of each subject throughout all of their trials with and without the presence of assistance. By tracking the relative frequencies of behaviors $\overline{\mathbb{K}}$ we can generate a distribution q with which to compare to $\mathbb{G}_{opt}$'s state distribution p. We can compare the distributions using task embodiment quantified by $D_{KL}(p||q)$, where a lower D_{KL} indicates greater embodiment of the task. This same procedure is applied to the two sessions the control group subjects.

We analyzed the human subjects dataset, and found that task embodiment is a reliable predictor of physical assistance and thus task performance. All subjects better embodied the task in their assisted trials, whereas there was no observed difference in the control group. In addition to comparing the groups using task embodiment, we also evaluated a standard metric for assessing task performance, the integrated MSE. Specifically, we calculated the integrated MSE with respect to the goal state of $(\theta, \dot{\theta}) = (0, 0)$. Integrated MSE is a reasonable performance metric for this task since success is defined as the ability to reach a single system configuration. However, we find that it predicts assistance at a lower significance level, and lower effect size than task embodiment.

A paired two-sample t-test on the task embodiment of each subject with and without assistance showed that the subjects' sessions with assistance ($\mu = 0.0756,\ \sigma = 0.0436$) significantly outperformed the sessions without assistance ($\mu = 0.2084,\ \sigma = 0.0560$), with $p =$ 2.8633e-16, $t(39) = 13.4876$, and an effect size of $d = 2.1326$. In contrast, there was no significant difference between the first session ($\mu = 0.2039,\ \sigma = 0.0406$) and the second session ($\mu = 0.1943,\ \sigma = 0.0400$) of the control group when a paired two-sample t-test was performed $p = 0.5546,\ t(12) = -0.6051$. These results indicate that task embodiment reliably captures assistance and lack thereof.

We also performed a paired two-sample t-test on the MSE of each subject with and without assistance, and found that the session with assistance ($\mu = 124.66,\ \sigma =$ 119.96) significantly outperformed the session without assistance ($\mu = 428.88,\ \sigma =$ 307.46), but with a lower significance and effect size than task embodiment, with $p =$ 1.2353e-7, $t(39) = 6.4526$ and an effect size of $d = 1.0202$. Again, we applied the paired two-sample t-test to the control group and found that the first session ($\mu = 352.83,\ \sigma =$

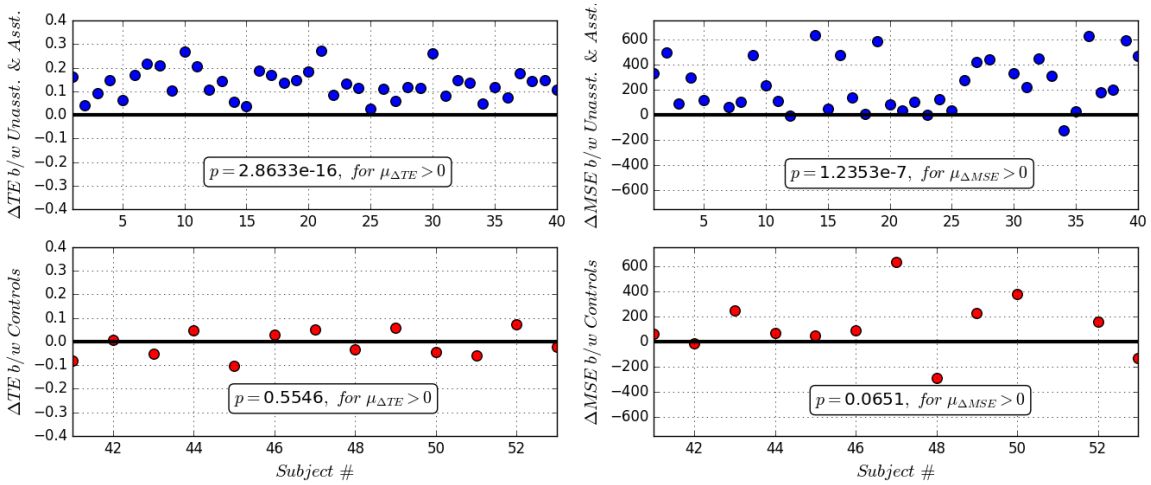


Figure 4.7. **Summary of human-subject experiment results.** Subjects in the experimental group who received assistance (blue) were compared to their own unassisted trials. The control group subjects (red) were compared from their initial session to their final session. The pair of plots to the left show the difference in task embodiment between the sessions of the experimental and control groups. The plots to the right show the difference between the same groups using the integrated MSE instead. Both task embodiment and MSE are good predictors of assistance, validating the DSS-defined performance measure, task embodiment, as a motion quality assessment tool.

217.67) did not significantly outperform the second ($\mu = 546.31$, $\sigma = 446.10$), had $p = 0.0651$, $t(12) = 2.0320$. These results indicate that MSE can also predict the presence of assistance, but not as reliably as task embodiment. The task embodiment measure has both a significance level several orders of magnitude greater than that of integrated MSE, and showed an effect size that was twice as large as integrated MSE. This demonstrates that task embodiment captures the broad difference between the assisted and unassisted trials. These results are summarized in Figure 4.7, where we see that the change in task embodiment (ΔTE) from assisted to unassisted trials is always positive.

By applying DSS onto the movements of a joint human-cart-pendulum system we were able to synthesize a coarse-grained representation of the dynamical system. Moreover, we

were able to compress the essence of what it means to succeed at the pendulum inversion task down to the 3 discrete elements of the node distribution, as demonstrated by the analysis above. Throughout this chapter, we have illustrated the effectiveness of DSS as a tool for motion analysis that is inspired by the formalisms of hybrid dynamical systems and information theory. In addition to analyzing the information encoded in the graphical model of DSS, in the following chapter we will demonstrate that the Koopman operator models themselves (graphically represented by the nodes of a DSS model) have sufficient predictive capacity intrinsically to be of use in model predictive control.

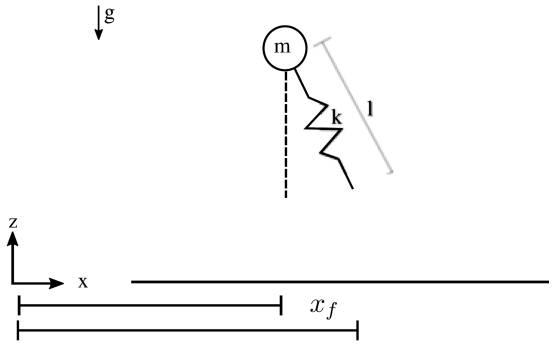
CHAPTER 5

Data-Driven Symbolic Abstractions for Control

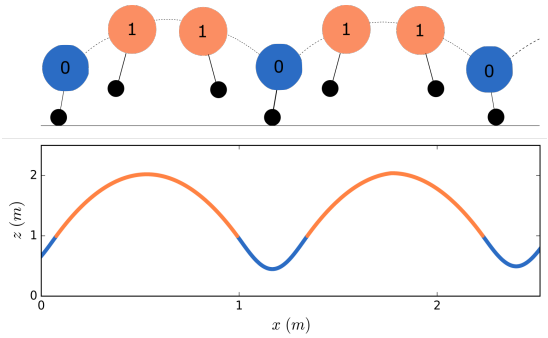
Hybrid systems can be difficult to model and control because of inherent discontinuities in the system dynamics. These discontinuities often cannot be represented by a single global model and must instead be represented piecewise. When these dynamics and their corresponding mode transition function are known ahead of time, nonlinear model predictive control (MPC) can be readily applied. However, for many systems of interest, such as lower-limb exoskeletons, the necessary information is not readily available. Hence, data-driven techniques like DSS are needed to predict the behavior of these multi-model systems and their transition events. Here, we illustrate the predictive capabilities of DSS and its application in the model predictive control of unknown hybrid dynamical systems.

To demonstrate the effectiveness of DSS model predictions, we consider an a simple yet highly nonlinear dynamic walking model: the spring-loaded inverted pendulum (SLIP). Without any *a priori* information about the dynamics, transition function, or number of hybrid modes, we use a DSS model to characterize the dynamics and then provide model predictive control to the system.

The SLIP system dynamics are split into two hybrid modes: stance and flight. The transition between these hybrid modes is described by an indicator function, often referred to as a guard equation, which specifies the set of dynamics evolving the system at a given coordinate in state-space. The system states are $\vec{x} = [x, z, \dot{x}, \dot{z}, x_f]^T$, where (x, z) indicates the position of the mass in the configuration specified in Figure 5.1(a), and x_f



(a) Configuration of spring-loaded inverted pendulum (SLIP) model.



(b) Identified behaviors of SLIP dynamic hopper: stance or flight.

Figure 5.1. **SLIP dynamic walker and corresponding DSS abstraction.** Here, we generate a DSS representation of the nonlinear SLIP dynamics as well as its guard equation purely from data.

is the position of the SLIP model's foot. The model's control inputs are $\vec{u} = [u_s, u_f]^T$. The dynamics of the SLIP model are

$$f_{stance} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ (k(l_0 - l(\vec{x})) + u_s)\frac{x - x_f}{ml(\vec{x})} \\ (k(l_0 - l(\vec{x})) + u_s)\frac{z}{ml(\vec{x})} - g \\ 0 \end{bmatrix}, \quad f_{flight} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ 0 \\ -g \\ \dot{x} + u_f \end{bmatrix}, \tag{5.1}$$

where l_0 is the resting length of the spring, k its stiffness, and $l(\vec{x}) = \sqrt{(x - x_f)^2 + z^2}$ is its variable length [**24**]. Then, the indicator function specifying transitions from one dynamical mode to another is $\Phi_{SLIP}(\vec{x}) = \text{sign}(1 - \frac{l_0}{l(\vec{x})}) \in \{-1, 1\}$. The complete hybrid

dynamics are

$$f_{SLIP}(\vec{x},\vec{u}) = \begin{cases} f_{stance}(\vec{x},\vec{u}), & \text{if } \Phi_{SLIP}(\vec{x}) = -1 \\ f_{flight}(\vec{x},\vec{u}), & \text{otherwise} \end{cases}. \tag{5.2}$$

To instantiate the DSS model, we use the dynamics in Eq. 5.1 and the controller in [**3**] to generate a stable trajectory hopping in place for $30s$ at $60Hz$. Despite this remarkably small amount of data, DSS synthesizes a symbolic model that closely matches the analytical dynamics, as shown in Figure 5.1(b). The generated model has two dynamic modes corresponding to flight and stance—just as in the analytical equations. Moreover, the learned guard equation that determines the discontinuous transition between hybrid modes accurately captures the behavior of the true function. We tested the accuracy of the hybrid mode classification in two $30s$-long trajectories: a constant velocity trajectory, and a variable velocity trajectory with direction changes. In the constant velocity scenario, the learned DSS transition function perfectly predicted the hybrid dynamic mode of the system with 100% accuracy. Additionally, in the variable velocity scenario DSS retained its effective mode predictions at an accuracy of 99.5%.

Equipped with the learned model, we now shift to using it towards model predictive control. To this end, we define an objective function with which to express our desired system behavior:

$$J(\vec{x}(t),\vec{u}(t),\vec{x}_d(t)) = \frac{1}{2}\int_{t_o}^{t_f} \|\vec{x}(t)-\vec{x}_d(t)\|_Q^2 + \|\vec{u}(t)\|_R^2 \, dt, \tag{5.3}$$

with $Q \succeq 0$ and $R \succeq 0$ being cost modifiers for the state error and control effort, respectively, and $\vec{x}_d(t)$ the desired trajectory to track. By specifying the initial condition,

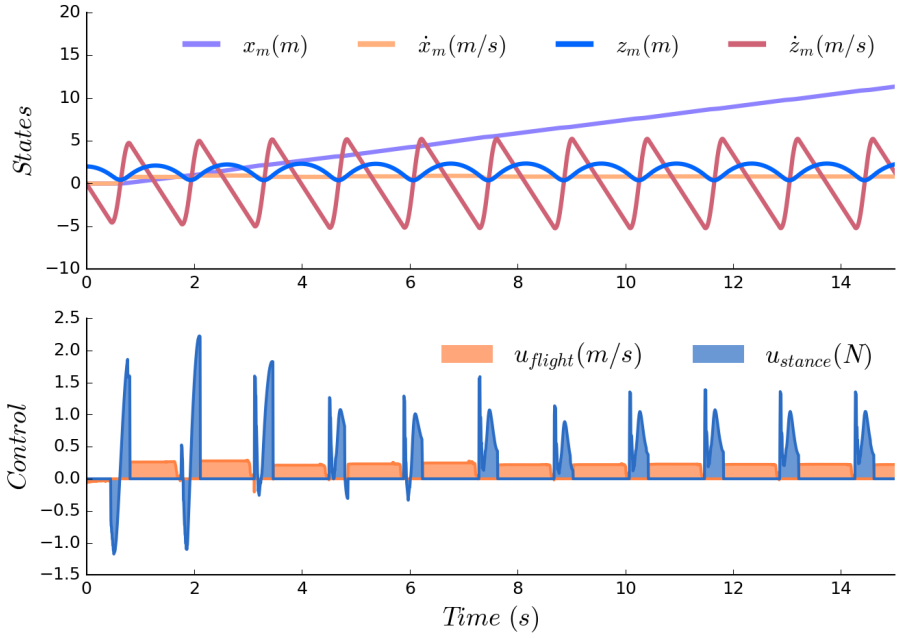


Figure 5.2. **Using DSS model to generate optimal model predictive control for SLIP walker.** Here, we make use of the DSS model of the nonlinear hybrid SLIP dynamics to enable nonlinear model predictive control of the system using an algorithm known as sequential action control (SAC). We demonstrate the performance of the model by successfully tracking a forward velocity of $0.4m/s$.

desired trajectory, objective function, system dynamics, and their corresponding spatial derivatives, nonlinear MPCs generate control inputs that minimize the given objective.

Using the DSS-derived model to inform the nonlinear MPC in [**3**], we run control the system to follow a desired trajectory of $\vec{x}_d(t) = [0, 0.4, 1.6, 0, 0]$, with matrix R as the 2×2 identity matrix, and Q with diagonal $Q_{diag} = [0, 50, 100, 0, 0]$ and 0 elsewhere. Based on this reference trajectory, the controller tries to maintain the SLIP center of mass at a height of $1.6m$ and a forward velocity of $0.4m/s$ throughout the duration of the simulation. For these control experiments, we start the SLIP system at an initial condition with height $z = 2m$ and zero forward velocity ($\dot{x}_m = 0m/s$). As shown in Figure 5.2, the

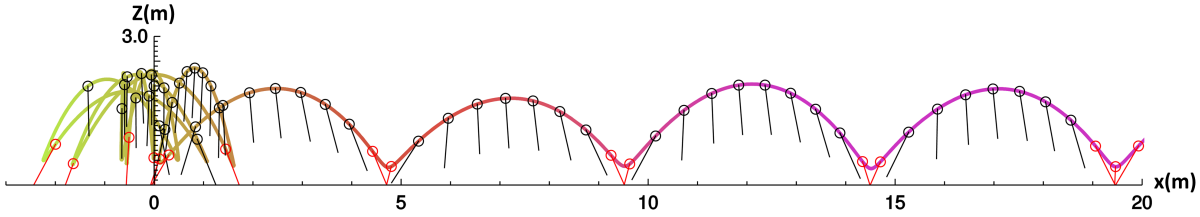


Figure 5.3. **Animated trajectory of SLIP hopper.** Here, we illustrate the trajectory of the SLIP hopper. First, we initialize the hopper to bounce in place with a stabilizing controller using its true dynamics to generate data to instantiate the DSS model, and then we switch to the control generated by the MPC using the DSS model to track the forward trajectory.

controller is able to keep the SLIP upright and moving forward at a velocity of $0.37m/s$, close to the desired $0.4m/s$, while having knowledge solely of the data-driven dynamics determined by DSS. In Figure 5.3, we illustrate an example of such a controlled SLIP trajectory.

In this chapter, we have illustrated the predictive capacity of the models generated by DSS beyond the information encoded in the structure of the symbolic model, and applied it to a control task. From analysis to application, DSS provides low-dimensional abstractions of complex dynamical systems that are both informative and useful towards the control of these systems.

CHAPTER 6

Future Work and Conclusions

Multi-modality is a salient feature of many complex systems of interest to the field of robotics and beyond. However, there do not exist many methods that explicitly analyze and take advantage of this structure. Here, we have introduced DSS to serve this need. By starting from the assumption that systems exhibit low-dimensional behavioral structure, DSS constructs a symbolic graphical abstraction that captures distinct system behaviors. We explored and demonstrated the different properties and applications of the models generated by DSS. We studied the topology of the DSS graphical models and what structure they qualitatively capture in the underlying motions, the relative frequencies of behaviors and how they can be used towards assessing motion quantitatively, and finally the model predictions themselves and their application in control.

While we hope to have motivated the broad utility of DSS across fields, there are particular questions that remain to be addressed. First, as with all learning models that depend on the use of basis functions, the methodology for selecting these functions is often *ad hoc* or heuristic. Although this issue is not unique to DSS, it is certainly a limitation of the approach and circumventing this issue would be greatly beneficial. Second, the use of the Euclidean metric in DSS when comparing Koopman operators is inadequate because the space of linear models is invariant under the action of the GL(n) group [**13**]. Hence, it would be beneficial to develop a different metric that more accurately captures the distance between dynamical systems qualitatively instead of an

element-wise distance. Finally, perhaps the more profound question is to ask what are the causes of the observed multi-modality in complex systems. Given an understanding of the mechanisms underlying the behavior of complex dynamical systems, we would be able to develop a physically-motivated algorithms that we can ground in our knowledge of the class of systems under study.

Despite lacking analytical physical grounding, we have demonstrated with DSS that complex dynamics can indeed be compressed into low-dimensional symbolic abstractions that capture much of the original behavior of the system. In future work, we will take a more mechanistic approach and develop algorithmic tools to match these physical insights. By imbuing our algorithms with our fundamental knowledge of physics, we may be able to develop analysis and control techniques for systems in which alternatives do not exist, such as spontaneously self-organizing swarms and active matter systems.

References

[1] I. Abraham and T. D. Murphey. Active learning of dynamics for data-driven control using koopman operators. *IEEE Transactions on Robotics*, 35(5):1071–1083, Oct 2019.

[2] I. Abraham, G. D. L. Torre, and T. D. Murphey. Model-based control using Koopman operators. *Robotics: Science and Systems*, 2017.

[3] A. Ansari and T. D. Murphey. Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems. *IEEE Trans. Rob.*, 32, 2017.

[4] T. A. Berrueta, I. Abraham, and T. Murphey. *Experimental Applications of the Koopman Operator in Active Learning for Control*, pages 421–450. Springer International Publishing, 2020.

[5] T. A. Berrueta, A. Pervan, K. Fitzsimons, and T. D. Murphey. Dynamical system segmentation for information measures in motion. *IEEE Robotics and Automation Letters*, 4(1):169–176, 2019.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[7] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS ONE*, 11(2):1–19, 2016.

[8] M. Budišić, R. Mohr, and I. Mezić. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.

[9] R. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, pages 160–172. Springer, 2013.

[10] H. G. Chambers and D. H. Sutherland. A practical guide to gait analysis. *Journal of the American Academy of Orthopaedic Surgeons (JAAOS)*, 10(3):222–231, 2002.

[11] C. Chen, D. Liu, X. Wang, C. Wang, and X. Wu. An adaptive gait learning strategy for lower limb exoskeleton robot. In *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 172–177, 2017.

[12] J. Chu, K. Song, S. Han, S. H. Lee, J. Y. Kang, D. Hwang, J. F. Suh, K. Choi, and I. Youn. Gait phase detection from sciatic nerve recordings in functional electrical stimulation systems for foot drop correction. *Physiological Measurement*, 2013.

[13] A. C. Costa, T. Ahamed, and G. J. Stephens. Adaptive, locally linear models of complex dynamics. *Proceedings of the National Academy of Sciences*, 116(5):1501–1510, 2019.

[14] M. D. Ellis, Y. Lan, J. Yao, and J. P. A. Dewald. Robotic quantification of upper extremity loss of independent joint control or flexion synergy in individuals with hemiparetic stroke: a review of paradigms addressing the effects of shoulder abduction loading. *J. NeuroEngineering and Rehabilitation*, 13(1):95, 2016.

[15] K. Fitzsimons, E. Tzorakoleftherakis, and T. D. Murphey. Optimal human-in-the-loop interfaces based on Maxwell's demon. In *American Control Conference (ACC)*, pages 4397–4402, 2016.

[16] R. G. Gallager. *Stochastic Processes: Theory for Applications*. Cambridge University Press, 2013.

[17] C. Gonzalez, D. Svenkeson, D. J. Kim, M. J. McKeown, and M. Oishi. Detection of manual tracking submovements in parkinson's disease through hybrid optimization. *IFAC Proceedings*, 48(27):291 – 297, 2015.

[18] N. Govindarajan, H. Arbabi, L. van Blargian, T. Matchen, E. Tegling, and I. Mezić. An operator-theoretic viewpoint to non-smooth dynamical systems: Koopman analysis of a hybrid pendulum. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6477–6484, 2016.

[19] J. D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[20] B. Heersink, M.A. Warren, and H. Hoffmann. Dynamic mode decomposition for interconnected control systems. *arXiv*, 2017.

[21] M. S. Hemati, M. O. Williams, and C. W. Rowley. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids*, 26(11):111701, 2014.

[22] E. Kaiser, J. N. Kutz, and S. L. Brunton. Data-driven discovery of Koopman eigenfunctions for control. In *arXiv*, 2017.

[23] A. Kalinowska, T. A. Berrueta, A. Zoss, and T. D. Murphey. Data-driven gait segmentation for walking assistance in a lower-limb assistive device. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1390–1396. IEEE, 2019.

[24] A. Kalinowska, K. Fitzsimons, J. P. A. Dewald, and T. D. Murphey. Online user assessment for minimal intervention during task-based robotic assistance. In *Robotics: Science and Systems*, 2018.

[25] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proc. National Academy of Sciences*, 17(5):315–318, 1931.

[26] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149 – 160, 2018.

[27] G. Mallapragada, I. Chattopadhyay, and A. Ray. Automated behaviour recognition in mobile robots using symbolic dynamic filtering. *J. Syst. Control Eng.*, 222:409–424, 2008.

[28] G. Mamakoukas, I. Abraham, and T. D. Murphey. Learning data-driven stable koopman operators, 2020.

[29] A. Mauroy, Y. Susuki, and I. Mezić. *Introduction to the Koopman Operator in Dynamical Systems and Control Theory*, pages 3–33. Springer International Publishing, 2020.

[30] L. McInnes, J. Healy, and S. Astels. HDBSCAN: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), 2017.

[31] I. Mezić. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Review of Fluid Mechanics*, 45(1):357–378, 2013.

[32] F. A. Mussa–Ivaldi and E. Bizzi. Motor learning through the combination of primitives. *Philos. Trans. R. Soc. B*, 355(1404):1755–1769, 2000.

[33] J. v. Neumann. Proof of the quasi-ergodic hypothesis. *Proceedings of the National Academy of Sciences*, 18(1):70–82, 1932.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] S. Peitz and S. Klus. Koopman operator-based model reduction for switched-system control of PDEs. In *arXiv*, 2017.

[36] C. Rao, A. Ray, S. Sarkar, and M. Yasar. Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns. *Signal, Image and Video Processing*, 3(2):101–114, 2009.

[37] W. Savoie, T. A. Berrueta, Z. Jackson, A. Pervan, R. Warkentin, S. Li, T. D. Murphey, K. Wiesenfeld, and D. I. Goldman. A robot made of robots: Emergent transport and control of a smarticle ensemble. *Science Robotics*, 4(34), 2019.

[38] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[39] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[40] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[41] A. H. A. Stienen, J. G. McPherson, A. C. Schouten, and J. P. A. Dewald. The ACT-4D: a novel rehabilitation robot for the quantification of upper limb motor impairments following brain injury. In *IEEE Int. Conf. on Rehabilitation Robotics*, pages 1–6, 2011.

[42] J. Taborri, E. Palermo, S. Rossi, and P. Cappa. Gait partitioning methods: A systematic review. *Sensors*, 16(1):66, 2016.

[43] J. Taborri, S. Rossi, E. Palermo, F. Patanè, and P. Cappa. A novel HMM distributed classifier for the detection of gait phases by means of a wearable inertial sensor network. *Sensors*, 14(9):16212–16234, 2014.

[44] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: Theory and applications. *J. Computational Dynamics*, 1:391, 2014.

[45] E. Tzorakoleftherakis and T. D. Murphey. Controllers as filters: Noise-driven swing-up control based on Maxwell's demon. In *IEEE Conf. on Decision and Control (CDC)*, pages 4368–4374, 2015.

[46] D. D. Vecchio, R. M. Murray, and P. Perona. Primitives for human motion: A dynamical approach. *IFAC Proceedings*, 35(1):25 – 30, 2002.

[47] D. D. Vecchio, R. M. Murray, and P. Perona. Decomposition of human motion into dynamics-based primitives with application to drawing tasks. *Automatica*, 39(12):2085 – 2098, 2003.

[48] H. T. T. Vu, F. Gomez, P. Cherelle, D. Lefeber, A. Nowé, and B. Vanderborght. ED-FNN: A new deep learning algorithm to detect percentage of the gait cycle for powered prostheses. *Sensors*, 18(7):2389, 2018.

[49] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *J. Nonlinear Science*, 25:1307–1346, 2015.