Data-Driven Optimal Targeting Control of Chaotic Dynamical Systems with Application to the Forced Duffing Equation

Samuel Otto

Princeton University, MAE 546 Optimal Control and Estimation, spring 2017

Abstract

Chaotic dynamical systems are common in nature and engineering problems. In many such systems, it is necessary to quickly stabilize an unstable periodic orbit in the chaotic motion with access only to small control inputs. In this report, we develop a modeling and control framework for solving this problem using only data collected from the dynamical system. The techniques are applied to stabilizing an unstable periodic orbit of the forced Duffing equation. A novel technique for modeling the dynamics is presented in which a collection of nonlinear models is fit to data using the Expectation Maximization (EM) algorithm. The modeled dynamics are used to design a Linear Quadratic Regulator (LQR) to stabilize the fixed point in a small neighborhood using the technique of Ott, Grebogi, and Yorke (OGY). Two methods are investigated for guiding far-away trajectories towards the desired orbit in a process known as targeting. Targeting is shown to reduce the average time to control by nearly two orders of magnitude over OGY alone. The first method utilizes an existing trajectory from the training data as a targeting path for guiding neighboring points towards a small neighborhood of the desired orbit. The second approach solves the optimal control problem globally using reinforcement learning. Though the second approach is expensive, it shows the greatest improvement in time to control. Methods are discussed for reducing the cost of this technique for real time optimal control.

Table of Contents

1. Introduction
2. The Forced Duffing Equation
   2.1 Overview
   2.2 Exact Linearized Dynamics at a Fixed Point
   2.3 Setting Up the Problem
   2.4 Generation and Use of Data
3. Data-Driven Modeling Technique
   3.1 Motivation and Requirements
   3.2 Local Nonlinear Modeling Approach
   3.3 Learning Maximum Likelihood Model Parameters
   3.4 The Expectation Maximization (EM) Algorithm
   3.5 Weighted Nonlinear Kernel Regression
   3.6 Weighted Gaussian Mixture Model
   3.7 Fitting the Categorical Distribution

3.8 Sparsity and K-Means
3.9 Application to Forced Duffing Equation's Poincare Map
4. Optimal OGY Control of an UPO
4.1 Designing the Optimal OGY Controller
4.2 Estimating the Time to Control
5. Simple Targeting Control
6. Optimal Targeting Control using Fitted Value Iteration
7. Conclusion

1. Introduction

Chaos is observed throughout nature and occurs frequently in engineering applications. Exceedingly simple systems possessing weak nonlinearities can exhibit sensitivity to initial conditions and seemingly random behavior. Chaotic dynamical systems are characterized by sensitivity to initial conditions and topological mixing of trajectories in an attracting region of phase space. An attractor is a minimal closed subset which is forward invariant under the dynamics and has an open neighborhood consisting of points which stay in the neighborhood and approach the attractor in forward time. An attractor is minimal in the sense that it has no proper subset with these properties. Chaotic attractors are topologically mixing -- a notion which corresponds with intuition in that any open subset of the attractor will eventually intersect any other open subset. Equivalently, the attractor possesses dense orbits which approach all points arbitrarily closely infinitely many times. Physically, such systems exhibit a wide range of behavior over their evolution. Though their dynamics are usually (but not necessarily) deterministic, the trajectory depends strongly on the initial condition. Therefore, a true trajectory diverges from neighboring trajectories and any simulation at an exponential rate over short times. Due to the topological mixing property, a chaotic attractor cannot possess a stable orbit, thereby leading to seemingly random evolution in time. A strange attractor possesses a transverse homoclinic orbit and hence a fractal invariant set with countable unstable periodic orbits. The apparent randomness may manifest itself in a number of ways depending on how quickly the mixing takes place and how much time the system spends near a particular unstable periodic orbit. Evolution may seem totally irregular or almost periodic over long time intervals. Chaotic systems may undergo intermittent intervals of varying length where different and distinct behaviors are observed – like lobe switching in the famous Lorenz system. This notion of randomness may be made precise if the chaotic attractor has a well-defined natural measure corresponding to the fraction of the time almost every trajectory spends in a given Borel set. The Sinai-Ruelle-Bowen theorem guarantees the existence of such a measure if the attractor meets an additional technical requirement by possessing a hyperbolic structure. The fact that a natural measure exists means that the attractor cannot be broken into disjoint invariant pieces of positive natural measure. Therefore, the evolution of almost every trajectory is ergodic and time averages of continuous functions over such trajectories correspond to spatial averages with respect to the natural measure (Guckenheimer & Holmes, 1983).

Chaos is widely considered to be an undesirable property in engineering systems due to fundamental unpredictability and sensitivity. Usually, we want to design our systems so that they have sufficient damping or control to remain in a non-chaotic regime. However, this standard approach may not be feasible if we have a limited power budget and/or access to a prohibitively small number of inputs compared to the active degrees of freedom in the system. Atrial fibrillation (Aejaz, 2004), (Creanga,

Nadejde, & Gasner, 2011) and epilepsy (Babloyantz & Destexhe, 1986), (Iasemidis & Sackellares, 1996) present medical instances of chaotic dynamics which implanted devices might be used to control. The amount of power consumed by such an implanted controller must be limited in order to retain operability over long periods of time. Furthermore, the number of accessible degrees of freedom relative to the number of active degrees of freedom in the organ is strictly limited by organ complexity and a need to minimize the invasiveness of any medical procedures used to implant such devices. Chaos is also known to arise in class B laser systems which include ruby, CO2, and most diode lasers (Arecchi & Meucci, 2008). In these cases, small control fluctuations are desired so that the maximum power can be supplied to the laser. It would be exceedingly expensive for controlling fluctuations to be on the same order of magnitude as the power used to drive the laser. Similar situations may also occur in multibody problems where only small amounts of thrust can be applied. An example might be found in asteroid mining where an asteroid laden with valuable minerals is in an unstable multi-body orbit. We will likely want to stabilize an unstable periodic orbit embedded in this motion using the minimum possible amount of energy or fuel. Nonlinear structural stiffness is also known to give rise to chaotic vibrations in lightweight structures. For spaceflight applications, an active device used to control these vibrations should also be lightweight and utilize the minimum possible power. In all of these examples, we wish to stabilize a chaotic dynamical system using small inputs.

Edward Ott, Celso Grebogi, and James Yorke (Ott, Grebogi, & Yorke, 1990) were the first to ask whether it is possible to control chaotic dynamical systems with arbitrarily small input. As they show in their seminal 1990 paper, it is the very ergodicity inherent to chaotic attractors which enables embedded unstable trajectories to be stabilized in finite time using arbitrarily small control perturbations to a system parameter. Their idea was simple – wait until a trajectory enters a sufficiently small neighborhood of the desired orbit in phase space before activating the control system. This method and its variants broadly referred to as OGY controllers open up the possibility of stabilizing both periodic and non-periodic trajectories existing in a small neighborhood of the chaotic attractor (Boccaletti, Grebogi, Lai, Mancini, & Maza, 2000). Another “attractive” application is synchronizing chaotic dynamical systems for covert communication using low power. Furthermore, using OGY control, it is possible to stabilize unstable chaotic orbits in repellers. In other words, we may incite lasting chaos in systems that only exhibit it temporarily or not at all, but still have an embedded fractal set. This has applications in mixing and chemical processes.

While the OGY method is theoretically attractive, it suffers from long waiting times before control can be achieved. The time to control can be estimated by a Poisson process using the natural measure of the small region where control can be achieve with small input. This neighborhood shrinks with the allowable control size and leads to long expected waiting times if the region is visited infrequently. In practical applications for controlling chaos, we desire a method of shortening the time to control while still using only small control actions. This entails applying control throughout the trajectory to guide it towards the desired orbit in a process called targeting. A targeting controller can be designed by considering “root” trajectories of the unperturbed system which naturally pass close to the desired orbit. A tree can be built from unperturbed trajectories passing closely to the root trajectory and so on. When a trajectory passes closely enough to this tree, control is activated and the point is guided down the branches and to the desired orbit (Boccaletti et al., 2000). A targeting controller using this method with 40 points on a simple tree consisting of two roots was shown to be capable of reducing the time to control an unstable periodic orbit of the forced Duffing equation by two orders of magnitude (Otto, 2017b). In

this report, we will investigate the problem of targeting using two optimal control methods. In the first, an optimal LQ controller is designed to guide points along a root trajectory to the desired orbit. In the second, we use Bellman's equations and fitted value iteration to design an optimal targeting controller on the entire attractor.

In order to design chaos controllers, we must have an accurate model of the system's dynamics. If the dynamics equations are unknown or are expensive to simulate, we must build a model with sufficient accuracy to enable control with small actions. In practice, we usually have access to observational data in the form of state snapshots coming from experiment or simulation with known control perturbations to some set of parameters. Due to the discrete nature of the data snapshots and modern digital control implementations, we consider continuous dynamical systems represented by a stroboscopic Poincare map capturing the phase space at regular time intervals. The Poincare map is a natural technique used to study the rich geometric structure of dynamical systems in phase space including the important interactions between stable and unstable manifolds associated with orbits embedded in strange attractors. Discrete systems are also capable of exhibiting more complex dynamics than continuous systems (Guckenheimer & Holmes, 1983). To be as general as possible, we will consider modeling and controlling discrete time chaotic dynamical systems. The high degree of nonlinearity and filamented distributions of data points observed in chaotic Poincare maps present unique challenges for data-driven modeling. A novel technique is developed where nonlinear kernel-based regression models are fit locally to data clusters in phase space using the Expectation Maximization (EM) algorithm. A Bayesian framework is used to select the model that is most likely to accurately reflect the dynamics at a given point (Otto, 2017a). The model fit to the chaotic Poincare map using this technique and perturbed training data alone is used to locate an unstable periodic orbit and design the targeting controllers.

In this report, the forced Duffing equation with additional forcing terms for control will be used as an example problem demonstrating chaotic dynamics. A novel data-driven modeling approach for chaotic dynamical systems is developed and evaluated on the forced Duffing equation's Poincare map. The model fit using the technique is used to identify an unstable periodic orbit of the forced Duffing equation. The model is then used to design optimal targeting controllers to stabilize this orbit using small control actions. Two types of targeting controllers are considered. The first is a simple approach which effectively expands the basin of attraction for the desired orbit using trajectories from the training data which came close to the desired orbit. The second approach constructs an optimal controller over the entire attractor using Bellman's equations and fitted value iteration.

2. The Forced Duffing Equation

2.1 Overview

This study will focus on the development of data-driven modeling and discrete time optimal targeting control techniques with numerical examples and experiments conducted on the Poincare map of the forced Duffing equation. The Duffing equation is a model of a simple forced oscillator with cubic nonlinear stiffness.

$$\ddot{\xi} + \delta\dot{\xi} + \alpha\xi + \beta\xi^3 = \gamma\cos(\omega t) + p(t) \tag{1}$$

In the above equation, δ is the damping, α is the linear stiffness, β is the nonlinear stiffening or softening, γ is the dominant forcing amplitude, and ω is the dominant forcing frequency. $p(t)$ is a small additional perturbative forcing term that we will use for control. The following parameter values have been selected $\delta = 0.25$, $\alpha = -1$, $\beta = 1$, $\gamma = 0.3$, and $\omega = 1$ since the time $T = 2\pi/\omega$ Poincare map of the Duffing equation is known to exhibit a transverse homoclinic crossing as these values giving rise to a strange attractor (Guckenheimer & Holmes, 1983). Physically, the Duffing equation at these parameter values represents a damped, forced oscillator in a double-welled potential. It is a model for the physical problem where a flexible steel beam is held at one end and allowed to be attracted at the other end to two separated magnets. The beam is excited with sinusoidal forcing as shown in the following figure. The Duffing oscillator is a simple physical apparatus known to give rise to chaotic oscillatory behavior in the beam's deflection.

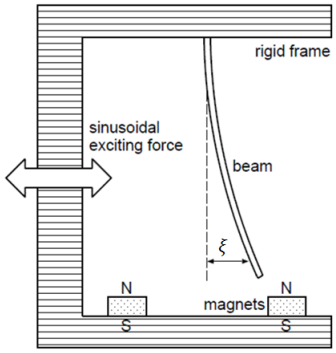

Figure 1: Physical apparatus whose dynamics are modeled by the forced Duffing equation. This setup is known to give rise to chaotic oscillations of the beam in experiments. Figure taken from (Kanamaru, 2008)

2.2 Exact Linearized Dynamics at a Fixed Point

An exact fixed point of the Poincare map corresponding to an unstable period-T orbit of the continuous time system was located at $\boldsymbol{x}_{FP}^{(exact)} = \left[\xi(nT), \dot{\xi}(nT)\right]^T = [0.1290, 0.1937]^T$ using a shooting method and Matlab's fsolve function. The continuous time dynamics were linearized about this periodic orbit $\bar{\xi}(t)$.

$$\frac{d}{dt}\begin{bmatrix}\xi \\ \dot{\xi}\end{bmatrix} = F(t)\begin{bmatrix}\xi \\ \dot{\xi}\end{bmatrix}, \quad F(t) = \begin{bmatrix}0 & 1 \\ -\alpha - \beta\bar{\xi}(t)^2 & -\delta\end{bmatrix} \tag{2}$$

The fundamental matrix $\Phi(t)$ was found by numerically integrating the initial conditions $[1,0]^T$ and $[0,1]$ using the linearized continuous time dynamics from $t = 0$ to $t = T$.

$$\frac{d}{dt}\Phi(t) = F(t)\Phi(t), \quad \Phi(0) = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix} \tag{3}$$

The Poincare map was linearized about $\boldsymbol{x}_{FP}^{(exact)}$ using the fundamental matrix and letting $\Delta\boldsymbol{x}_n = \boldsymbol{x}(nT) - \boldsymbol{x}_{FP}^{(exact)}$ as follows

$$\Delta \boldsymbol{x}_{n+1} = \Phi(T)\Delta \boldsymbol{x}_n \tag{4}$$

This allows us to determine the stability type of the identified fixed point. We find that $\boldsymbol{x}_{FP}^{(exact)}$ is a saddle point with stable eigenvalue $\lambda_s = -0.026$ and unstable eigenvalue $\lambda_u = -8.076$ corresponding to stable and unstable tangent subspaces $E^s = span[0.7878, -0.6159]^T$ and $E^u = span[0.7692, 0.6390]^T$ respectively. The un-forced Poincare map was iterated 50000 times from the initial condition $[0,0]^T$ and plotted in the following figure. The exact fixed point as well as the stable and unstable tangent spaces are shown in the figure.

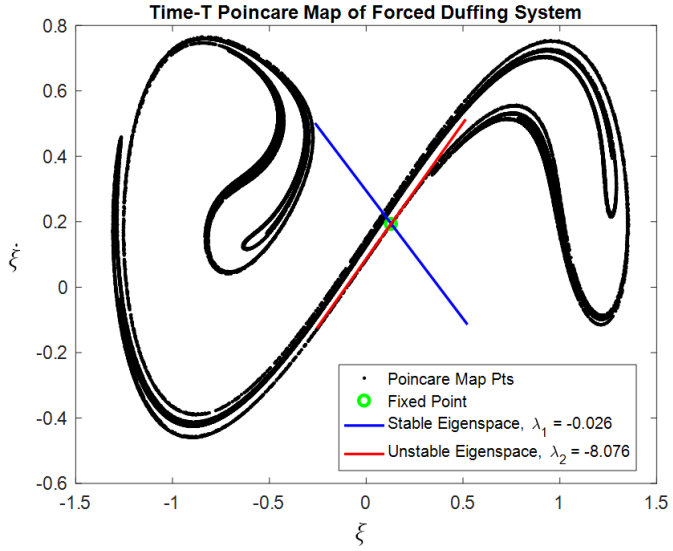

Figure 2: Poincare map of the forced Duffing equation with no additional forcing perturbations. The exact fixed point as well as the stable and unstable tangent spaces are illustrated.

2.3 Setting up the Problem

We are primarily interested in controlling the discrete time Poincare map by making small changes to some set of parameters once every map iteration. The objective will be to stabilize the unstable periodic orbit corresponding the fixed Point of the Poincare map using small control perturbations. We will follow the technique of optimal multidimensional OGY control (Epureanu & Dowell, 2000) and introduce time-dependent parametric forcing of the form

$$p(t) = u(1)\underbrace{1}_{b_1(t)} + u(2)\underbrace{\cos(\omega t)}_{b_2(t)} + u(3)\underbrace{\sin(\omega t)}_{b_3(t)} + u(4)\underbrace{\cos(2\omega t)}_{b_4(t)} + u(5)\underbrace{\sin(2\omega t)}_{b_5(t)} \tag{5}$$

The size of the control input is measured using the energy of each Fourier basis mode. These energies define a Gram matrix G_W and weighted norm for measuring the "size" of the control input $\boldsymbol{u}_n \in \mathbb{R}^5$ at the nth time step. This norm is to be kept below a given threshold ϵ_u limiting the energy used for control over each time interval $t \in [nT, (n+1)T]$.

$$\|\boldsymbol{u}\|_W^2 = \int_0^T |p(t)|^2 dt = \sum_{i=1}^{5}\sum_{j=1}^{5} u(i)\left[\int_0^T b_i(t)b_j(t)dt\right]u(j) = \boldsymbol{u}^T G_W \boldsymbol{u} \leq \epsilon_u^2 \tag{6}$$

The effect of using different numbers of Fourier basis modes in the parameterization was investigated in (Otto, 2017b). No benefit was found using more that 7 such modes. We have selected 5 modes as a middle ground between optimizing the time-dependent forcing and keeping the model fitting problem inexpensive with limited training data. This formulation results in controlled discrete time dynamics captured by the time T Poincare map

$$\boxed{\boldsymbol{x}_{n+1} = \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n), \quad \boldsymbol{x}_n \in \mathbb{R}^2, \quad \boldsymbol{u}_n \in \mathbb{R}^5} \tag{7}$$

2.4 Generation and Use of Data

The problem is now in the common and highly general form to which the methods developed herein can be applied. Using the simulated forced duffing equation, three data sets were generated and formed the sole basis for constructing models of the Poincare map and designing targeting controllers. The data sets were intended to reflect the data that we might collect in a physical experiment. The first data set consists of 5000 iterations of the Poincare map starting from $[0,0]^T$ using no additional forcing $p(t) \equiv 0$. It will be referred to as the unforced training set. The second data set consists of 5000 iterations of the Poincare map starting from $[0,0]^T$ using uniformly distributed random forcing perturbations at each time step $u_n(j) \sim unif([-0.01,0.01]), j = 1,2,\dots,5$. It will be referred to as the training set. The last data set was reserved for testing the accuracy of the learned model and consisted of 1000 iterations of the Poincare map starting from a random initial condition in $[-1.5,1.5] \times [-0.6,0.8]$. uniformly distributed random forcing perturbations were used at each time step $u_n(j) \sim unif([-0.01,0.01]), j = 1,2,\dots,5$. It will be referred to as the testing set. It is emphasized that we have limited ourselves to using the data only and never the Duffing equation itself to design all controllers presented herein. The learned dynamics model is evaluated on the test set. Of course the controllers, once designed, are tested using the simulated forced Duffing equation.

3. Data-Driven Modeling Technique

3.1 Motivation and Requirements

In real world modeling and control problems, we may have access to accurate simulations or experimental data coming from the system itself. It is usually prohibitively expensive to use the simulation or experimental apparatus itself to predict and analyze the dynamics. This problem is especially evident when we must make many predictions quickly in order to solve nonlinear estimation and control problems. In the design of optimal controllers and estimators like the extended Kalman filter, we also require quick access to linearizations of the dynamics around many points in phase space. Even in the case of the forced Duffing equation – a simple ordinary differential equation with 5 control inputs, such computations are prohibitive. This is because each time we want to predict the Poincare map 1 step into the future, we must integrate the Duffing equation over a new time interval of length T. In order to linearize, we must integrate the Jacobian matrix over a nominal trajectory to arrive at the fundamental

matrix as we did when we analyzed the exact fixed point in section 2.2. To solve even the simplest Linear Quadric (LQ) optimal control problem, we must do this at least once for each point on the nominal trajectory. If the nominal trajectory is unknown, then the nonlinear optimization process may require us to evaluate such linearizations thousands of times as we minimize some cost function by gradient descent. If we desire to implement an extended Kalman filter, then we will need to be able to evaluate linearizations of the dynamics on the fly. This means that we must perform these calculations at least as fast as the system evolves.

If it is not practical to perform full simulations or experiments to predict the dynamics and construct linearizations, we must build simple and accurate models of the system from the available data. This will be our starting point. We assume that data can be collected from the system consisting of its state and the control action applied. Of course we need not observe the entire state. If we have access to a limited number of observables, the dynamics contributing to the observable variables may be recovered using a time-delay or Takens embedding (Takens, 1981). All methods discussed herein are extensible to the case of time-delayed coordinates. For the model we build to be useful, we must be able to evaluate and linearize it much more quickly than the system itself. The model must also be sufficiently accurate to accomplish the intended goal. In our case, this means accurately locating fixed points, periodic orbits, and designing optimal controllers which perform only small control actions. The requirement that control actions be small adds a layer of difficulty since the model we build must be accurate to the same scale as the control effect. In other words, if our control actions can only produce deviations in the trajectory of magnitude δ then our model must be able to make accurate predictions to within $\epsilon < \delta$. This task is further complicated by the high degree of nonlinearity in chaotic Poincare maps as well as the geometry typical chaotic data sets in phase space. The data generally tends towards the attractor itself, taking on its fractal structure. The attractor can be thought of as consisting of the closure of the unstable manifolds attached to each embedded orbit (Guckenheimer & Holmes, 1983). In a strange attractor, these manifolds are piled up on themselves, causing the distribution of points to be continuously stretched and re-injected like folding taffy. As seen in the Poincare map of the forced Duffing equation, these interactions result in a highly filamented fractal structure of the data point distribution in phase space. Typical nonlinear regression-based approaches are not ideal when applied globally to these problems. They do not seem to have the representational capacity required to capture the nonlinearity in the dynamics from data which are not sufficiently spread out. When the capacity is increased through addition of more nonlinear terms, they tend to over fit the data and generalize poorly to new points. Additionally, high-capacity nonlinear models are expensive to evaluate and begin to rival the cost of simulating the entire system.

3.2 Local Nonlinear Modeling Approach

We must balance the need to capture highly nonlinear dynamics over irregular regions of phase space with the requirement that the model can be evaluated and linearized quickly. This naturally leads us towards an approach where many lower capacity models are used to approximate the dynamics in different regions covering the attractor. To predict the dynamics of a new point, we need only evaluate and/or linearize a single relatively simple model from the collection. If the attractor spans a large region or we wish to increase the accuracy of our model, we need not increase the complexity of the individual models. Rather, we add more such simple models to give a refined covering of the attractor.

Two related problems are fundamental to the construction and use of such a collection of simple models. The first problem is how to choose which of the many models is best applied to a new point. The second problem is how to fit such a collection of simple models to the data. A naïve approach is to select points from the data at random or perform k-means clustering prior to fitting a model to each cluster. Model selection is based on the nearest neighboring model. This approach is not ideal since there is no guarantee that the data points will cluster in regions where there is high nonlinearity in the dynamics requiring a higher concentration of models.

We need an approach which incorporates information about the dynamics with information about the distribution of the data in phase space. This is accomplished by introducing a Bayesian probabilistic framework whose parameters are fit according to maximum likelihood using the Expectation Maximization (EM) algorithm. Suppose that each simple model $i = 1,2,\dots,N$ of the dynamics at the given state $\boldsymbol{x}_n$ takes the form

$$\boldsymbol{X}_{n+1}^i = \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n) + \boldsymbol{V}^i, \quad \boldsymbol{V}^i \sim p_{\boldsymbol{V}^i}(\boldsymbol{v}^i) = \mathcal{N}(\boldsymbol{v}^i, \boldsymbol{0}, R^i) \tag{8}$$

Where the model consists of deterministic dynamics given by $\hat{\boldsymbol{f}}^i$ and additive Gaussian noise $\boldsymbol{V}^i$. The noise term $\boldsymbol{V}^i$ accounts for noise, exogenous disturbances, and modeling errors. We are treating the updated state as a Gaussian random vector $\boldsymbol{X}_{n+1}^i$ centered at the ith deterministic model estimate. We use a latent random variable $Z \in \{1,2,\dots,N\}$ to indicate the model. Before we know anything about the current state $\boldsymbol{x}_n$, the prior probabilities of the models are assumed to follow a categorical or "multinoulli" distribution

$$P(Z = i) = \phi^i, \quad \sum_{i=1}^{N} \phi^i = 1 \tag{9}$$

In order to classify the current state $\boldsymbol{x}_n$ among the N models, it is most natural to consider first a region of validity for each model. The conditional probability density of states $\boldsymbol{x}$ whose dynamics are described by the ith model is assumed to have a Gaussian distribution with centroid $\boldsymbol{\mu}_x^i$ and covariance Σ^i.

$$p_{\boldsymbol{X}|Z=i}(\boldsymbol{x}|Z = i) = \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_x^i, \Sigma^i) \tag{10}$$

The centroids $\boldsymbol{\mu}_x^i$ will be referred to as the "model centroids" since they represent the centroids of the collections of points where each model is valid. Bayes rule is used to infer the conditional probabilities of each model at given state $\boldsymbol{x}$.

$$\boxed{P(Z = i|\boldsymbol{X} = \boldsymbol{x}) = \frac{p_{\boldsymbol{X}|Z=i}(\boldsymbol{x}|Z = i)P(Z = i)}{\sum_{k=1}^{N} p_{\boldsymbol{X}|Z=k}(\boldsymbol{x}|Z = k)P(Z = k)}} \tag{11}$$

The probabilities of each model at given state $\boldsymbol{x}$ give us a variety of options for estimating the dynamics. Numerical experiments (Otto, 2017a) have shown that simply choosing the most likely model gives more

accurate estimates of the dynamics than weighted combinations, at least in the case of the forced Duffing equation.

3.3 Learning Maximum Likelihood Model Parameters

The task remains to learn all of the model parameters including distributions involving the latent variable Z. In order to use a compact notation, we will call all the parameters associated with the ith model $M^i = \{\hat{\boldsymbol{f}}^i, \boldsymbol{\mu}_x^i, R^i, \Sigma^i, \phi^i\}$ and the parameters of all the models $\boldsymbol{M} = \{M^1, M^2, \dots, M^N\}$. We will maximize the likelihood of the model parameters given the training data $\{(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n)\}_{n=1}^m$ using the Expectation Maximization (EM) algorithm. The log likelihood of $\boldsymbol{M}$ is given by the negative empirical cross entropy between the training data distribution and the model distribution using parameters $\boldsymbol{M}$. This is the average number of bits needed to identify examples drawn from the true data distribution if the code is designed for the model distribution with parameters $\boldsymbol{M}$. We get the minimum average number of bits when the true data distribution $p_{data}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n)$ is used in a code achieving the entropy $H\big(p_{data}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n)\big)$. If the encoding of experimental data drawn from the true distribution is done based on model distribution $p_{model}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n; \boldsymbol{M})$, the average number of bits required is the cross entropy. This will be a sub-optimal code if the distributions differ; thereby achieving a sub-optimal average number of bits and having higher cross entropy than if the distributions agreed. Minimizing the cross entropy minimizes the difference between the model distribution $p_{model}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n; \boldsymbol{M})$ and the true distribution in the limit of infinite training data.

$$\ell(\boldsymbol{M}) = -H(p_{data}, p_{model}) = \underset{(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n) \sim p_{data}}{\mathbb{E}} \log[p_{model}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n)] \tag{12}$$

$$\ell(\boldsymbol{M}) = \lim_{m \to \infty} \ell_m(\boldsymbol{M}) = \lim_{m \to \infty} \frac{1}{m} \sum_{n=1}^{m} \log\left[p_{model}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n; \boldsymbol{M})\right] \quad A.E. \tag{13}$$

Using our Bayesian framework, we can calculate the model probability density at training points $(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n)$ using parameter values $\boldsymbol{M}$. Using the law of total probability, we sum over the disjoint events corresponding to choices of the latent random variable $Z = 1,2, \dots, N$

$$p_{model}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n; \boldsymbol{M}) = \sum_{i=1}^{N} p_{\boldsymbol{X}_n, \boldsymbol{X}_{n+1}, \boldsymbol{U}_n | Z=i}\big(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n | Z = i; M^i\big) P\big(Z = i; M^i\big) \tag{14}$$

Applying the chain rule for conditional probability, the joint probability of the state, control, and updated state given the ith model is given by the following. From now on, we will occasionally dropt the M^i from expressions where it is clear by context that the ith model's parameters are being used i.e. when conditioning on Z.

$$\begin{aligned} &p_{\boldsymbol{X}_n, \boldsymbol{X}_{n+1}, \boldsymbol{U}_n | Z=i}(\boldsymbol{x}_n, \boldsymbol{x}_{n+1}, \boldsymbol{u}_n | Z = i) \\ &= p_{\boldsymbol{X}_{n+1} | \boldsymbol{X}_n = \boldsymbol{x}_n, \boldsymbol{U}_n = \boldsymbol{u}_n, Z=i}(\boldsymbol{x}_{n+1} | \boldsymbol{X}_n = \boldsymbol{x}_n, \boldsymbol{U}_n = \boldsymbol{u}_n, Z = i) p_{\boldsymbol{X}_n, \boldsymbol{U}_n | Z=i}(\boldsymbol{x}_n, \boldsymbol{u}_n | Z = i) \end{aligned} \tag{15}$$

So far, we have included the joint distribution $p_{\boldsymbol{X}_n,\boldsymbol{U}_n|Z=i}(\boldsymbol{x}_n,\boldsymbol{u}_n|Z=i)$ over the state and input. Keeping the joint distribution is important if the control applied during the experiment did not have a uniform distribution or was dependent on the state. Such a situation will arise if some kind of control was attempted during the training phase. If we gathered the training data using a preliminary controller, we could use the explicit control law to model this joint probability and fit a collection of nonlinear models taking the dependence of control on state into account. This gives us a way to iteratively design controllers, where we collect data using the current controller and append it to the data used to fit the model. The model is updated and used to design a new controller. Accounting for the dependence of control $\boldsymbol{u}_n$ on state $\boldsymbol{x}_n$ in the growing training set, we improve the model – enabling it to resolve the controlled dynamics more accurately each time. The model and controller will simultaneously converge by repeatedly iterating this process. In this report, we ignore the more general case and assume that the distribution of control inputs in the training data was an unchanging uniform distribution described in section 2.4.

$$p_{\boldsymbol{X}_n,\boldsymbol{U}_n|Z=i}(\boldsymbol{x}_n,\boldsymbol{u}_n|Z=i) = \underbrace{p_{\boldsymbol{U}_n|\boldsymbol{X}_n=\boldsymbol{x}_n,Z=i}(\boldsymbol{u}_n|\boldsymbol{X}_n=\boldsymbol{x}_n,Z=i)}_{const.}\, p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i) \tag{16}$$

Therefore, putting (13)-(16) together, the empirical log likelihood is given by

$$\ell_m(\boldsymbol{M}) = \frac{1}{m}\sum_{n=1}^{m} \log\left[\sum_{i=1}^{N} p_{\boldsymbol{X}_{n+1}|\boldsymbol{X}_n=\boldsymbol{x}_n,Z=i}(\boldsymbol{x}_{n+1}|\boldsymbol{X}_n=\boldsymbol{x}_n,Z=i,\boldsymbol{u}_n)p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i)P(Z=i)\right] + m*\log\left[const.\right] \tag{17}$$

Ignoring the constant and substituting in our model for the dynamics, we have the log likelihood

$$\boxed{\ell_m(\boldsymbol{M}) = \frac{1}{m}\sum_{n=1}^{m} \log\left[\sum_{i=1}^{N} p_{\boldsymbol{V}^i}\left(\boldsymbol{x}_{n+1}-\hat{\boldsymbol{f}}^i(\boldsymbol{x}_n,\boldsymbol{u}_n)\right) p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i)P(Z=i)\right]} \tag{18}$$

If we kept the dependence of $\boldsymbol{u}_n$ on $\boldsymbol{x}_n$ then we get the following instead

$$\ell_m(\boldsymbol{M}) = \frac{1}{m}\sum_{n=1}^{m} \log\left[\sum_{i=1}^{N} p_{\boldsymbol{V}^i}\left(\boldsymbol{x}_{n+1}-\hat{\boldsymbol{f}}^i(\boldsymbol{x}_n,\boldsymbol{u}_n)\right) p_{\boldsymbol{X}_n,\boldsymbol{U}_n|Z=i}(\boldsymbol{x}_n,\boldsymbol{u}_n|Z=i)P(Z=i)\right] \tag{19}$$

where the joint distribution over the state and control is determined from (16) by the (possibly stochastic) control law $p_{\boldsymbol{U}_n|\boldsymbol{X}_n=\boldsymbol{x}_n,Z=i}(\boldsymbol{u}_n|\boldsymbol{X}_n=\boldsymbol{x}_n,Z=i)$ used during training. If the iterative approach is to be taken, it makes sense to choose the model and corresponding control law at $\boldsymbol{x}_n$ stochastically based on the posterior model probabilities given by (11). Using this process to generate new training data allows us to improve all of the models at each point, rather than just the most likely one. It is also possible to engage in exploratory control during training, where the distribution from which control actions are sampled changes based on our confidence in the model. This is a topic for future research. The iterative model fitting and controller design process will be explored in later work. We do not pursue it further in this report.

3.4 The Expectation Maximization (EM) Algorithm

The EM algorithm is an iterative method for fitting maximum likelihood parameters in Bayesian models with latent variables. In eqn. (18) we introduce a to-be-determined distribution over the models at each training point $Q_n(Z)$

$$\ell_m(\boldsymbol{M}) = \frac{1}{m}\sum_{n=1}^{m} \log\left[\sum_{i=1}^{N} Q_n(Z=i)\frac{p_{\boldsymbol{V}^i}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n)\right) p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i)P(Z=i)}{Q_n(Z=i)}\right] \tag{20}$$

The inner sum is now in the form of an expectation with respect to the distribution we introduced

$$\ell_m(\boldsymbol{M}) = \frac{1}{m}\sum_{n=1}^{m} \log\left(\mathop{\mathbb{E}}_{Z\sim Q_n}\left[\frac{p_{\boldsymbol{V}^Z}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^Z(\boldsymbol{x}_n, \boldsymbol{u}_n)\right) p_{\boldsymbol{X}_n|Z}(\boldsymbol{x}_n|Z)P(Z)}{Q_n(Z)}\right]\right) \tag{21}$$

The logarithm is a concave function. Therefore, we can lower bound the log likelihood using Jensen's inequality

$$\ell_m(\boldsymbol{M}) \geq \tilde{\ell}_m(\boldsymbol{M}) = \frac{1}{m}\sum_{n=1}^{m} \mathop{\mathbb{E}}_{Z\sim Q_n}\left[\log\left(\frac{p_{\boldsymbol{V}^Z}\left(\boldsymbol{x}_{n+1} - \boldsymbol{f}^Z(\boldsymbol{x}_n, \boldsymbol{u}_n)\right) p_{\boldsymbol{X}_n|Z}(\boldsymbol{x}_n|Z)P(Z)}{Q_n(Z)}\right)\right] \tag{22}$$

$$\tilde{\ell}_m(\boldsymbol{M}) = \frac{1}{m}\sum_{n=1}^{m}\sum_{i=1}^{N} Q_n(Z=i)\log\left[\frac{p_{\boldsymbol{V}^i}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n)\right) p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i)P(Z=i)}{Q_n(Z=i)}\right] \tag{23}$$

Two observations are in order regarding eqn. (23). Now that the logarithm has been brought inside the sum, we may split up the log of the product. This will allow us to decouple our optimizations over the parameters for each distribution at each step. We also observe that the inequality $\ell_m(\boldsymbol{M}) \geq \tilde{\ell}_m(\boldsymbol{M})$ becomes equality when the inside term does not depend on the model choice Z. In this case, we are taking the expectation of a constant, which is just the constant itself. We may freely exchange the log and the expectation of a constant. The EM algorithm is the iterative process in which a lower bound with expectation based on previous model parameters is maximized to give new parameters. We make a clever choice for the distribution $Q_n(Z)$ so that $\ell_m(\boldsymbol{M}) = \tilde{\ell}_m(\boldsymbol{M})$ precisely when the model parameters have converged. Letting $\bar{\boldsymbol{M}}$ be the current set of model parameters, we choose to use the following distribution in the expectation

$$\boxed{\bar{Q}_n^i \triangleq Q_n(Z=i;\bar{\boldsymbol{M}}) = \frac{p_{\boldsymbol{V}^i}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n);\bar{\boldsymbol{M}}\right) p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i;\bar{\boldsymbol{M}})P(Z=i;\bar{\boldsymbol{M}})}{\sum_{k=1}^{N} p_{\boldsymbol{V}^k}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^k(\boldsymbol{x}_n, \boldsymbol{u}_n);\bar{\boldsymbol{M}}\right) p_{\boldsymbol{X}_n|Z=k}(\boldsymbol{x}_n|Z=k;\bar{\boldsymbol{M}})P(Z=k;\bar{\boldsymbol{M}})}} \tag{24}$$

We see that if we use eqn. (24) for $Q_n(Z)$ in eqn. (22) then the expectation and log commute if $\boldsymbol{M} = \bar{\boldsymbol{M}}$ yielding equality $\ell_m(\boldsymbol{M}) = \tilde{\ell}_m(\boldsymbol{M})$ when the model parameters have converged. Computing $\bar{Q}_n^i$ using eqn. (24) is the Expectation or E-step in the EM algorithm. Once the form of the expectation is determined, we proceed with the Maximization or M-step by maximizing the lower bound $\tilde{\ell}(\boldsymbol{M})$ as follows

$$\bar{\boldsymbol{M}} \leftarrow \underset{\boldsymbol{M}}{\operatorname{argmax}} \sum_{n=1}^{m} \sum_{i=1}^{N} \bar{Q}_n^i \log \left[\frac{p_{\boldsymbol{V}^i}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n)\right) p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i)P(Z=i)}{\bar{Q}_n^i} \right] \tag{25}$$

Successively maximizing the lower bounds $\tilde{\ell}(\boldsymbol{M})$ gives a monotone increase in the log likelihood function as illustrated in the following figure. The E-step and M-step are iterated until convergence $\boldsymbol{M}^* = \bar{\boldsymbol{M}}$ where equality of the lower bound and the true log likelihood is achieved $\ell_m(\boldsymbol{M}^*) = \tilde{\ell}_m(\boldsymbol{M}^*)$.

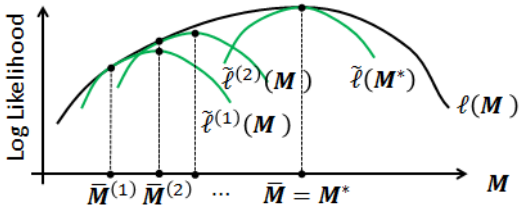

Figure 3: Monotone increase in log likelihood achieved by iterative maximization of lower bounds using the EM algorithm

Expanding the log of the product in eqn. (25) we see that the maximization step decouples into three parts whose solutions are each well-known.

$$\bar{\boldsymbol{M}} \leftarrow \underset{\boldsymbol{M}}{\operatorname{argmax}} [(I) + (II) + (III)] = \underset{\boldsymbol{M}}{\operatorname{argmin}} (J_I + J_{II} + J_{III}) \tag{26}$$

Where the first part consist of solving a weighted regression problem with cost function depending only on $\{(\hat{\boldsymbol{f}}^1, R^1), (\hat{\boldsymbol{f}}^2, R^2), \ldots, (\hat{\boldsymbol{f}}^N, R^N)\}$

$$J_I = -\sum_{n=1}^{m} \sum_{i=1}^{N} \bar{Q}_n^i \log p_{\boldsymbol{V}^i}\left(\boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n)\right) = -\sum_{n=1}^{m} \sum_{i=1}^{N} \bar{Q}_n^i \log \mathcal{N}\left(\boldsymbol{x}_{n+1}, \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n), R^i\right) \tag{27}$$

The second part is to fit a weighted mixture of Gaussians with cost function depending only on $\{(\boldsymbol{\mu}_x^1, \Sigma^1), (\boldsymbol{\mu}_x^2, \Sigma^2), \ldots, (\boldsymbol{\mu}_x^N, \Sigma^N)\}$

$$J_{II} = -\sum_{n=1}^{m} \sum_{i=1}^{N} \bar{Q}_n^i \log p_{\boldsymbol{X}_n|Z=i}(\boldsymbol{x}_n|Z=i) = -\sum_{n=1}^{m} \sum_{i=1}^{N} \bar{Q}_n^i \log \mathcal{N}\left(\boldsymbol{x}_n, \boldsymbol{\mu}_x^i, \Sigma^i\right) \tag{28}$$

And the last part is to fit the categorical distribution with cost function and constraint whose parameters are $\{\phi^1, \phi^2, \dots, \phi^N\}$

$$J_{III} = -\sum_{n=1}^{m}\sum_{i=1}^{N} \bar{Q}_n^i \log\, P(Z=i) = -\sum_{n=1}^{m}\sum_{i=1}^{N} \bar{Q}_n^i \log\, \phi^i, \quad 1 = \sum_{i=1}^{N} \phi^i \tag{29}$$

Each part of the maximization step has an analytical solution. These solutions are discussed in the following sections. Though the maximization step decouples, the resulting learned model parameters are coupled through the distribution $\bar{Q}_n^i$ found in the E-step.

3.5 Weighted Nonlinear Kernel Regression

We consider the minimization of J_I and show that it is equivalent to nonlinear least squares. Observe first the eqn. (27) decouples further into separate minimizations to find each $(\hat{\boldsymbol{f}}^1, R^1), (\hat{\boldsymbol{f}}^2, R^2), \dots, (\hat{\boldsymbol{f}}^N, R^N)$

$$J_I = \sum_{i=1}^{N} J_I^i(\hat{\boldsymbol{f}}^i, R^i), \quad J_I^i(\hat{\boldsymbol{f}}^i, R^i) = -\sum_{n=1}^{m} \bar{Q}_n^i \log\, \mathcal{N}\left(\boldsymbol{x}_{n+1}, \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n), R^i\right) \tag{30}$$

Substituting the Gaussian densities and letting $\boldsymbol{v}_n^i = \boldsymbol{x}_{n+1} - \hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n)$ gives the independent optimization problems below

$$(\hat{\boldsymbol{f}}^i, R^i) = \underset{(\hat{\boldsymbol{f}}^i, R^i)}{\operatorname{argmin}} J_I^i(\hat{\boldsymbol{f}}^i, R^i) = \underset{(\boldsymbol{f}^i, R^i)}{\operatorname{argmin}} \sum_{n=1}^{m} \bar{Q}_n^i \left[\log|R^i| + (\boldsymbol{v}_n^i)^T (R^i)^{-1} (\boldsymbol{v}_n^i)\right] \tag{31}$$

We may first solve for R^i by setting the derivative of $J_I^i(\hat{\boldsymbol{f}}^i, R^i)$ with respect to R^i equal to zero. Doing so yields the following solution after some manipulation

$$\boxed{R^i = \left[\sum_{n=1}^{m} \bar{Q}_n^i\right]^{-1} \sum_{n=1}^{m} \bar{Q}_n^i (\boldsymbol{v}_n^i)(\boldsymbol{v}_n^i)^T} \tag{32}$$

Before solving the weighted nonlinear kernel regression problem, we consider the case where each $\hat{\boldsymbol{f}}^i$ is a finite linear combination of nonlinear features. Each feature $\theta_1, \theta_2, \dots, \theta_L: \mathbb{R}^{n+q} \to \mathbb{R}$ is a real valued function of the state and control. It will be convenient to work with the feature vector

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}(\boldsymbol{x}_n, \boldsymbol{u}_n) \triangleq [\theta_1(\boldsymbol{x}_n, \boldsymbol{u}_n) \quad \theta_2(\boldsymbol{x}_n, \boldsymbol{u}_n) \quad \cdots \quad \theta_L(\boldsymbol{x}_n, \boldsymbol{u}_n)]^T \in \mathbb{R}^L \tag{33}$$

We define the models using these features

$$\hat{\boldsymbol{f}}^i(\boldsymbol{x}_n, \boldsymbol{u}_n) = A^i \boldsymbol{\theta}(\boldsymbol{x}_n, \boldsymbol{u}_n), \quad A^i \in \mathbb{R}^{n \times L} \tag{34}$$

Differentiating the cost $J_l^i(\hat{\boldsymbol{f}}^i, R^i) = J_l^i(A^i, R^i)$ with respect to A^i and setting it equal to zero gives us an equation depending on R^i. Substituting eqn. (32) and solving for A^i gives

$$\boxed{A^i = \left[\sum_{n=1}^{m} \bar{Q}_n^i \boldsymbol{x}_{n+1}(\boldsymbol{\theta}_n)^T\right]\left[\sum_{n=1}^{m} \bar{Q}_n^i \boldsymbol{\theta}_n(\boldsymbol{\theta}_n)^T\right]^{-1}} \tag{35}$$

We now connect this to the classical nonlinear weighted least squares problem. This will allow us to formulate nonlinear kernel regression in its most natural context. If we create the following data matrices

$$X' = [\boldsymbol{x}_2 \cdots \boldsymbol{x}_{m+1}] \in \mathbb{R}^{n\times m}, \quad \Theta = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_m] \in \mathbb{R}^{L\times m}, \quad W^i = diag\left[\bar{Q}_1^i, \dots, \bar{Q}_m^i\right] \in \mathbb{R}^{m\times m} \tag{36}$$

the solutions given by eqn. (32) and eqn. (35) can be written compactly

$$A^i = \left(X'W^i\Theta^T\right)\left(\Theta W^i\Theta^T\right)^{-1}, \quad R^i = \frac{1}{tr(W^i)}\left(X' - A^i\Theta\right)W^i\left(X' - A^i\Theta\right)^T \tag{37}$$

More importantly, we observe that eqn. (37) is precisely the solution of the weighted least squares regression problem

$$A^i = \underset{A\in\mathbb{R}^{n\times L}}{\operatorname{argmin}} \left\|(X' - A\Theta)\sqrt{W^i}\right\|_F^2 \tag{38}$$

The above formulation of the problem allows us to introduce possibly infinite dimensional nonlinear feature maps in a natural way. We will find that the solution depends only on inner products of features in the intrinsic feature Hilbert space $\mathcal{H}$ enabling us to use a kernel. Define the feature map and combination operator into Hilbert space $\mathcal{H}$ respectively as follows

$$\begin{array}{cc} \boldsymbol{\theta}\colon \mathbb{R}^{n+q} \to \mathcal{H} & \boldsymbol{\Theta}\colon \mathbb{R}^m \to \mathcal{H} \\ (\boldsymbol{x}, \boldsymbol{u}) \mapsto \boldsymbol{\theta}(\boldsymbol{x}, \boldsymbol{u}) & \boldsymbol{v} \mapsto v_1\boldsymbol{\theta}(\boldsymbol{x}_1, \boldsymbol{u}_1) + \cdots + v_m\boldsymbol{\theta}(\boldsymbol{x}_m, \boldsymbol{u}_m) \end{array} \tag{39}$$

For compactness, we will use the same notation as above with $\boldsymbol{\theta}_n \triangleq \boldsymbol{\theta}(\boldsymbol{x}_n, \boldsymbol{u}_n) \in \mathcal{H}$. We will start by considering only the lth component $\hat{f}_l^i(\boldsymbol{x}, \boldsymbol{u})$ of the nonlinear function $\hat{\boldsymbol{f}}^i(\boldsymbol{x}, \boldsymbol{u})$. We will find a (possibly infinite) combination $\boldsymbol{\psi}_l^i \in \mathcal{H}$ of features in $\mathcal{H}$ to represent the function

$$\hat{f}_l^i(\boldsymbol{x}, \boldsymbol{u}) = \langle\boldsymbol{\theta}(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{\psi}_l^i\rangle_{\mathcal{H}} \tag{40}$$

For ease of notation, we form the following vector and write the adjoint operator $\boldsymbol{\Theta}^*\colon \mathcal{H} \to \mathbb{R}^m$ explicitly

$$\boldsymbol{x}'(l) \triangleq \begin{bmatrix} x_1(l) \\ \vdots \\ x_m(l) \end{bmatrix} \in \mathbb{R}^m \qquad \boldsymbol{\Theta}^*\boldsymbol{\psi}_l = \begin{bmatrix} \langle \boldsymbol{\theta}_1, \boldsymbol{\psi}_l \rangle_{\mathcal{H}} \\ \vdots \\ \langle \boldsymbol{\theta}_m, \boldsymbol{\psi}_l \rangle_{\mathcal{H}} \end{bmatrix} \in \mathbb{R}^m \tag{41}$$

Using eqn. (38) as a template, we pose the weighted least squares regression problem for $\hat{f}_l^i$ in the Hilbert space $\mathcal{H}$. In the following equation, we have introduced an L^2 regularization penalty or "ridge" on the size of the combination $\boldsymbol{\psi}_l$. This is done in order to avoid overfitting the training data when there are an excess of nonlinear features. Ridge also ensures that the solution is unique when $\lambda > 0$.

$$\boldsymbol{\psi}_l^i = \underset{\boldsymbol{\psi}_l \in \mathcal{H}}{\operatorname{argmin}} \left\| \sqrt{W^i} (\boldsymbol{x}'(l) - \boldsymbol{\Theta}^* \boldsymbol{\psi}_l) \right\|_2^2 + \lambda \langle \boldsymbol{\psi}_l, \boldsymbol{\psi}_l \rangle_{\mathcal{H}} \tag{42}$$

When $\lambda > 0$ we can prove that $\boldsymbol{\psi}_l = \boldsymbol{\Theta}\boldsymbol{a}_l = a_l(1)\boldsymbol{\theta}_1 + \cdots + a_l(m)\boldsymbol{\theta}_m \in \mathcal{R}(\boldsymbol{\Theta})$. Suppose that $\boldsymbol{\psi}_l = \widehat{\boldsymbol{\psi}}_l + \widetilde{\boldsymbol{\psi}}_l$ where $\widehat{\boldsymbol{\psi}}_l \in \mathcal{R}(\boldsymbol{\Theta})$ and $\widetilde{\boldsymbol{\psi}}_l \in \mathcal{R}(\boldsymbol{\Theta})^\perp$. Then $\boldsymbol{\Theta}^*\boldsymbol{\psi}_l = \boldsymbol{\Theta}^*\widehat{\boldsymbol{\psi}}_l + \boldsymbol{\Theta}^*\widetilde{\boldsymbol{\psi}}_l = \boldsymbol{\Theta}^*\widehat{\boldsymbol{\psi}}_l$ since $\mathcal{N}(\boldsymbol{\Theta}^*) = \mathcal{R}(\boldsymbol{\Theta})^\perp$, but $\langle \boldsymbol{\psi}_l, \boldsymbol{\psi}_l \rangle_{\mathcal{H}} = \langle \widehat{\boldsymbol{\psi}}_l, \widehat{\boldsymbol{\psi}}_l \rangle_{\mathcal{H}} + \langle \widetilde{\boldsymbol{\psi}}_l, \widetilde{\boldsymbol{\psi}}_l \rangle_{\mathcal{H}} \geq \langle \widehat{\boldsymbol{\psi}}_l, \widehat{\boldsymbol{\psi}}_l \rangle_{\mathcal{H}}$. Therefore,

$$\therefore \quad \boldsymbol{\psi}_l = \widehat{\boldsymbol{\psi}}_l \quad \text{and} \quad \exists \boldsymbol{a}_l \in \mathbb{R}^m \quad \text{s.t.} \quad \boldsymbol{\psi}_l = \boldsymbol{\Theta}\boldsymbol{a}_l \in \mathcal{R}(\boldsymbol{\Theta}) \tag{43}$$

If $\lambda = 0$, there may be infinitely many $\boldsymbol{\psi}_l \in \mathcal{H}$ solving eqn. (42). In this case, we look for the smallest $\boldsymbol{\psi}_l \in \mathcal{R}(\boldsymbol{\Theta})$ in the span of the training data. This is analogous to the right pseudoinverse of $\boldsymbol{\Theta}^*$ in $\mathcal{H}$. Representing the solution as a linear combination of the training data in $\mathcal{H}$ is akin to Occam's razor and is called the Learning Subspace Property (LSP) in kernel-based machine learning. Using LSP, the regression problem becomes finite-dimensional

$$\boldsymbol{a}_l^i = \underset{\boldsymbol{a}_l \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \sqrt{W^i} (\boldsymbol{x}'(l) - \boldsymbol{\Theta}^* \boldsymbol{\Theta} \boldsymbol{a}_l) \right\|_2^2 + \lambda \sum_{r=1}^{m} \sum_{s=1}^{m} a_l(r) \langle \boldsymbol{\theta}(\boldsymbol{x}_r, \boldsymbol{u}_r), \boldsymbol{\theta}(\boldsymbol{x}_s, \boldsymbol{u}_s) \rangle_{\mathcal{H}} a_l(s) \tag{44}$$

Let the matrix

$$K \triangleq \boldsymbol{\Theta}^*\boldsymbol{\Theta} = [\langle \boldsymbol{\theta}(\boldsymbol{x}_r, \boldsymbol{u}_r), \boldsymbol{\theta}(\boldsymbol{x}_s, \boldsymbol{u}_s) \rangle_{\mathcal{H}}]_{1 \leq r,s \leq m} \in \mathbb{R}^{m \times m} \tag{45}$$

Doing some simple manipulation on eqn. (44) we may solve the problem

$$\boldsymbol{a}_l^i = \underset{\boldsymbol{a}_l \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \sqrt{W^i} (\boldsymbol{x}'(l) - K\boldsymbol{a}_l) \right\|_2^2 + \lambda \boldsymbol{a}_l^T K \boldsymbol{a}_l = \underset{\boldsymbol{a}_l \in \mathbb{R}^m}{\operatorname{argmin}} \left\| \begin{bmatrix} \sqrt{W^i}\boldsymbol{x}'(l) \\ \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \sqrt{W^i}K \\ \sqrt{\lambda K} \end{bmatrix} \boldsymbol{a}_l \right\|_2^2 \tag{46}$$

Which is in the form of a standard $2m$-dimensional least squares problem whose solution is

$$\boldsymbol{a}_l^i = \left(W^i K + \lambda I_m \right)^{-1} W^i \boldsymbol{x}'(l) \tag{47}$$

Giving the lth component of the ith nonlinear model

$$\hat{f}_l^i(\boldsymbol{x},\boldsymbol{u}) = \langle \boldsymbol{\theta}(\boldsymbol{x},\boldsymbol{u}), \boldsymbol{\Theta}\boldsymbol{a}_l^i \rangle_{\mathcal{H}} = \left(\boldsymbol{\Theta}^*\boldsymbol{\theta}(\boldsymbol{x},\boldsymbol{u})\right)^T \left(W^i K + \lambda I_m\right)^{-1} W^i \boldsymbol{x}'(l) \tag{48}$$

Notice that the entire solution to the regression problem is expressed in terms of inner products of elements in $\mathcal{H}$. We introduce the kernel function

$$\boldsymbol{\kappa}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{bmatrix}, \begin{bmatrix}\boldsymbol{x}'\\ \boldsymbol{u}'\end{bmatrix}\right) \triangleq \langle \boldsymbol{\theta}(\boldsymbol{x},\boldsymbol{u}), \boldsymbol{\theta}(\boldsymbol{x}',\boldsymbol{u}') \rangle_{\mathcal{H}} \in \mathbb{R} \quad \text{and} \quad \boldsymbol{\kappa}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{bmatrix}\right) \triangleq \boldsymbol{\Theta}^*\boldsymbol{\theta}(\boldsymbol{x},\boldsymbol{u}) \in \mathbb{R}^m \tag{49}$$

Combining the $l = 1,2,\dots,n$ components of the nonlinear function

$$\boxed{\hat{\boldsymbol{f}}^i(\boldsymbol{x},\boldsymbol{u}) = X' W^i \left(K W^i + \lambda I_m\right)^{-1} \boldsymbol{\kappa}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{bmatrix}\right)}, \quad K = \left[\boldsymbol{\kappa}\left(\begin{bmatrix}\boldsymbol{x}_r\\ \boldsymbol{u}_r\end{bmatrix}, \begin{bmatrix}\boldsymbol{x}_s\\ \boldsymbol{u}_s\end{bmatrix}\right)\right]_{1\le r,s\le m} \in \mathbb{R}^{m\times m} \tag{50}$$

This approximation is easily differentiated if we know the derivative of the kernel with respect to one of its slots $D_{\left[\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{smallmatrix}\right]}\boldsymbol{\kappa}\left(\begin{bmatrix}\boldsymbol{x}_0\\ \boldsymbol{u}_0\end{bmatrix}, \begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{bmatrix}\right)$ allowing us to find $D_{\left[\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{smallmatrix}\right]}\boldsymbol{\kappa}\left(\begin{bmatrix}\boldsymbol{x}_0\\ \boldsymbol{u}_0\end{bmatrix}\right) \in \mathbb{R}^{m\times(n+q)}$

$$\boxed{D_{\left[\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{smallmatrix}\right]}\hat{\boldsymbol{f}}^i(\boldsymbol{x}_0,\boldsymbol{u}_0) = X' W^i \left(K W^i + \lambda I_m\right)^{-1} D_{\left[\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{smallmatrix}\right]}\boldsymbol{\kappa}\left(\begin{bmatrix}\boldsymbol{x}_0\\ \boldsymbol{u}_0\end{bmatrix}\right) = \begin{bmatrix}\Phi(\boldsymbol{x}_0,\boldsymbol{u}_0) & \Gamma(\boldsymbol{x}_0,\boldsymbol{u}_0)\end{bmatrix}} \in \mathbb{R}^{n\times(n+q)} \tag{51}$$

If some of the weights on the diagonal of W^i are zero, the Schur compliment can be used to show that we may neglect the corresponding training examples. This gives identical but smaller matrices in eqns. (50) and (51). This allows us to evaluate the model and its linearization more quickly if the weights $\bar{Q}_n^i$ are sparse. Encouraging sparsity in the weights is the topic of section 3.8.

3.6 Weighted Gaussian Mixture Model

We consider the minimization of J_{II} and show that it is equivalent to fitting a weighted mixture of Gaussians. We proceed analogously to our approach for J_I. Observe first the eqn. (28) decouples further into separate minimizations to find each $(\boldsymbol{\mu}_x^1, \Sigma^1), (\boldsymbol{\mu}_x^2, \Sigma^2), \dots, (\boldsymbol{\mu}_x^N, \Sigma^N)$

$$J_{II} = \sum_{i=1}^{N} J_{II}^i\left(\boldsymbol{\mu}_x^i, \Sigma^i\right), \quad J_{II}^i\left(\boldsymbol{\mu}_x^i, \Sigma^i\right) = -\sum_{n=1}^{m} \bar{Q}_n^i \log\, \mathcal{N}\left(\boldsymbol{x}_n, \boldsymbol{\mu}_x^i, \Sigma^i\right) \tag{52}$$

Substituting the Gaussian densities gives the independent optimization problems below

$$\left(\boldsymbol{\mu}_x^i, \Sigma^i\right) = \underset{\left(\boldsymbol{\mu}_x^i, \Sigma^i\right)}{\operatorname{argmin}} J_{II}^i\left(\boldsymbol{\mu}_x^i, \Sigma^i\right) = \underset{\left(\boldsymbol{\mu}_x^i, \Sigma^i\right)}{\operatorname{argmin}} \sum_{n=1}^{m} \bar{Q}_n^i \left[\log\left|\Sigma^i\right| + \left(\boldsymbol{x}_n - \boldsymbol{\mu}_x^i\right)^T \left(\Sigma^i\right)^{-1} \left(\boldsymbol{x}_n - \boldsymbol{\mu}_x^i\right)\right] \tag{53}$$

We first solve for Σ^i by setting the derivative of $J_{II}^i(\boldsymbol{\mu}_x^i, \Sigma^i)$ with respect to Σ^i equal to zero. Doing so yields the following solution after some manipulation

$$\boxed{\Sigma^i = \left[\sum_{n=1}^{m} \bar{Q}_n^i\right]^{-1} \sum_{n=1}^{m} \bar{Q}_n^i (\boldsymbol{x}_n - \boldsymbol{\mu}_x^i)(\boldsymbol{x}_n - \boldsymbol{\mu}_x^i)^T} \tag{54}$$

Setting the derivative of $J_{II}^i(\boldsymbol{\mu}_x^i, \Sigma^i)$ with respect to $\boldsymbol{\mu}_x^i$ equal to zero gives an equation depending on Σ^i into which eq. (54) is substituted. The solution is

$$\boxed{\boldsymbol{\mu}_x^i = \left[\sum_{n=1}^{m} \bar{Q}_n^i\right]^{-1} \sum_{n=1}^{m} \bar{Q}_n^i \boldsymbol{x}_n} \tag{55}$$

It is also possible, as it was for the regression problem, to extend the Gaussian mixture model into a high-dimensional nonlinear feature Hilbert space using kernels. This allows us to fit curved non-Gaussian densities to the data in the original empirical phase space. This is an area of current research and will not be pursued here.

3.7 Fitting the Categorical Distribution

The categorical distribution is easy to fit using eqn. (29). In order to account for the constraint, the Lagrangian is formed

$$\mathcal{L}(\phi^1, \ldots, \phi^N, \alpha) = \sum_{n=1}^{m} \sum_{i=1}^{N} \bar{Q}_n^i \log \phi^i + \alpha \left[1 - \sum_{i=1}^{N} \phi^i\right] \tag{56}$$

Setting the derivatives of eqn. (56) to zero and doing some algebra gives the solution

$$\boxed{\phi^i = \frac{1}{m} \sum_{n=1}^{m} \bar{Q}_n^i} \tag{57}$$

The above is the maximum likelihood categorical distribution fit to the $\bar{Q}_n^i$-weighted training data.

3.8 Sparsity and K-Means

While our solutions to the maximization step given above are given in closed form, the regression update can be prohibitively expensive to compute. Observe that eqns. (50) and (51) require us to invert the symmetric positive definite $m \times m$ matrix $\left(KW^i + \lambda I_m\right)$ for each model $i = 1,2, \ldots, N$ for each iteration of the EM algorithm. We also observe that most $\bar{Q}_n^i$ are small due to the Gaussian weighting of points assigned to each model. If we introduce sparsity by setting some $\bar{Q}_n^i$ to zero, it can be shown that the size of the corresponding matrix to be inverted shrinks correspondingly. Considering the ith model,

assume that only $\bar{Q}^i_{n_1}, \bar{Q}^i_{n_2}, \dots, \bar{Q}^i_{n_{m_i}}$ are nonzero. Eqns. (50) and (51) reduce to identical computations involving smaller matrices and fewer kernel evaluations

$$\tilde{X}'^i = \left[\boldsymbol{x}_{n_1}, \dots, \boldsymbol{x}_{n_{m_i}}\right], \quad \widetilde{W}^i = diag\left[\bar{Q}^i_{n_1}, \dots, \bar{Q}^i_{n_{m_i}}\right], \quad \widetilde{K} = \left[K_{n_r n_s}\right]_{1\le r,s\le m_i}, \quad \text{and}$$
$$\tilde{\boldsymbol{k}}^i\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{k}\left(\left[\boldsymbol{x}_{n_1}, \boldsymbol{u}_{n_1}\right]^T, [\boldsymbol{x}, \boldsymbol{u}]^T\right) \\ \vdots \\ \boldsymbol{k}\left(\left[\boldsymbol{x}_{n_{m_i}}, \boldsymbol{u}_{n_{m_i}}\right]^T, [\boldsymbol{x}, \boldsymbol{u}]^T\right)\end{bmatrix} \in \mathbb{R}^{m_i} \tag{58}$$

In order to promote sparsity in the weights, we will take an approach similar to the idea behind k-means clustering. Rather than letting a training point $\boldsymbol{x}_n$ be used in several models, we force it to belong only to the top $N' < N$ models based on the likelihood $\bar{Q}_n(Z; \bar{\boldsymbol{M}})$. The remaining $N - N'$ weights are zeroed out. Allowing a point to belong to more than a single $N' > 1$ model forces the learned models to be accurate in overlapping regions. Numerical experiments show that we can quickly fit a collection of models using $N' = 1$ or 2 and use it to initialize a more expensive solution with $N' > 2$.

3.9 Application to Forced Duffing Equation's Poincare Map

This technique was used to solve the system identification problem for the Poincare map of the forced Duffing equation with control. In addition to system identification, the collection of local Gaussian densities fit to the data can be used to analyze the structure of its distribution in phase space. The task of inferring underlying data manifolds using these clusters will not be taken up in this report but is a topic for future research.

The EM algorithm with k-means sparsity was used to fit $N = 40$ nonlinear kernel models to the training data set. Regression was done using the Radial Basis Function (RBF) kernel

$$\boldsymbol{k}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{bmatrix}, \begin{bmatrix}\boldsymbol{x}'\\ \boldsymbol{u}'\end{bmatrix}\right) = \exp\left(-\frac{1}{2\sigma^2}\left\|\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{u}\end{bmatrix} - \begin{bmatrix}\boldsymbol{x}'\\ \boldsymbol{u}'\end{bmatrix}\right\|_2^2\right), \quad \sigma = 0.1 \tag{59}$$

The L^2 ridge regression penalty $\lambda = 5 * 10^{-4}$ was used. The solution was initialized using k-means sparsity with $N' = 2$. Once converged, the number of models was increased to $N' = 4$ and the algorithm was run until convergence. Convergence was determined when the centroids did not change by more than 10^{-6}. In order to prevent the Gaussians from becoming degenerate during training, minimum eigenvalues $\epsilon_\Sigma = 10^{-4}$ and $\epsilon_R = 10^{-4}$ were imposed on the Σ and R matrices. The model centroids and clusters shown in the following figure were obtained on the testing data set. We see that the cluster centroids lie on or near the filaments making up the training data distribution. Each model is relatively small and thereby able to be evaluated quickly. Maximum likelihood model assignment was done using eqn. (11).

$$\hat{\boldsymbol{x}}_{n+1} = \hat{\boldsymbol{f}}(\boldsymbol{x}_n, \boldsymbol{u}_n) = \hat{\boldsymbol{f}}^{i^*}(\boldsymbol{x}_n, \boldsymbol{u}_n) \quad \text{where} \quad i^* = \text{argmax}_i\, P(Z = i | \boldsymbol{X} = \boldsymbol{x}_n) \tag{60}$$

The testing data points shown in the figure are colored according to their maximum likelihood model assignment

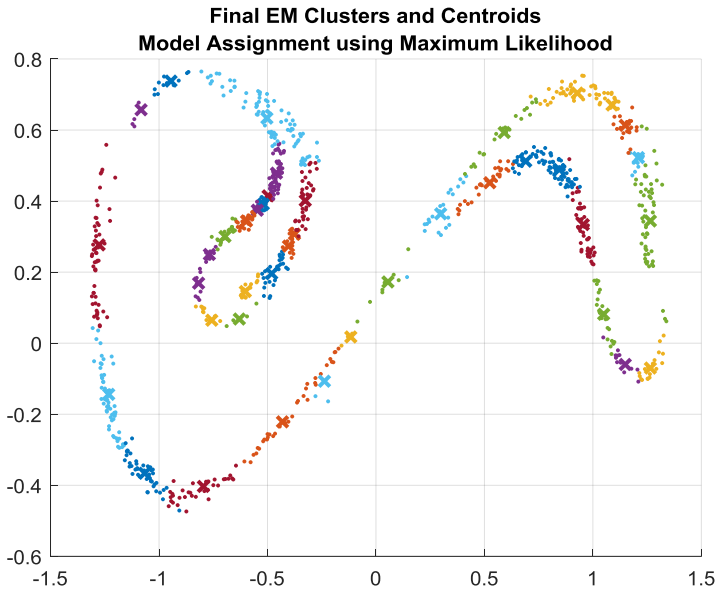

Figure 4: Point clusters and centroids identified using the EM algorithm on the testing data. Testing data points are colored according to maximum likelihood model assignment eqn. (60)

Several metrics were used to quantify the accuracy of the learned nonlinear models. The first is the mean estimate distance error on the testing data set

$$\bar{E}_D = \frac{1}{m_{test}} \sum_{n=1}^{m_{test}} \|\boldsymbol{x}_{n+1} - \hat{\boldsymbol{x}}_{n+1}\| = 0.0265 \tag{61}$$

This distance error is small comparted to the scale of the attractor. A histogram of the model estimate distance errors on the testing data set is plotted in the following figure. The median distance error was 0.0018.

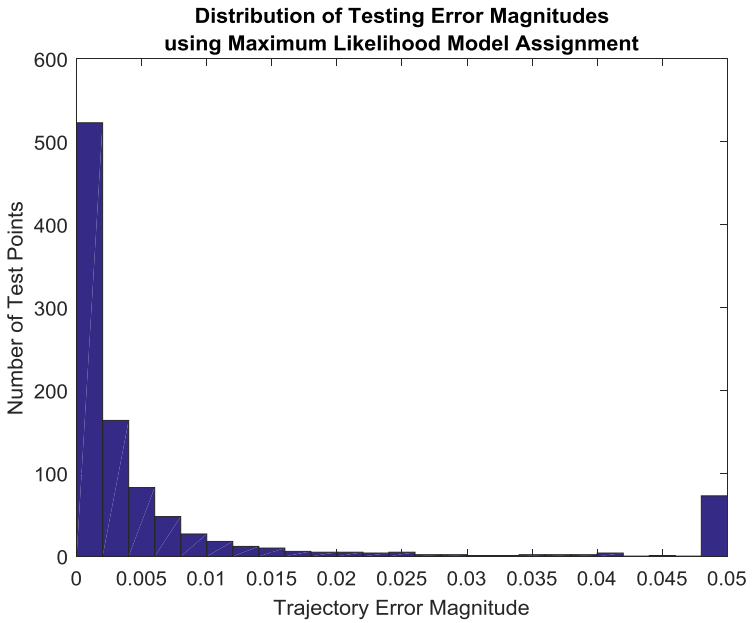

Figure 5: Histogram of model prediction distance errors on testing data set

A decaying error norm over many steps is used to assess how quickly the model diverges from the true dynamics. The decaying error norm starting at point $\boldsymbol{x}_0$ is found by simultaneously iterating the true dynamics and modeled dynamics over $M = 10$ time steps with the same random control inputs $\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_{M-1}$. The true state evolution is given by $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_M$ and the modeled state evolution is given by $\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2, \dots, \widehat{\boldsymbol{x}}_M$. We form the decaying error norm

$$E_\gamma^M(\boldsymbol{x}_0) = \sum_{n=1}^{M} \frac{\|\boldsymbol{x}_n - \widehat{\boldsymbol{x}}_n\|}{\gamma^{n-1}}, \quad \gamma = 2 \tag{62}$$

The empirical mean is taken over the data distribution by using the testing data set as initial conditions

$$\bar{E}_\gamma^M = \frac{1}{m_{test}} \sum_{n=1}^{m_{test}} E_\gamma^M(\boldsymbol{x}_n) = 0.1812 \tag{63}$$

In order to evaluate how well the model reflects the system's dynamics and long term behavior, the learned model was used to simulate an entirely new Poincare map for 2000 iterations stating from a randomly selected point $\boldsymbol{x}_0 = [0.7140, 0.5256]^T$ from the testing data set. Zero control input was used. This modeled Poincare map is plotted in the following figure. The map resembles figure 2, though it has some points which are spuriously scattered.

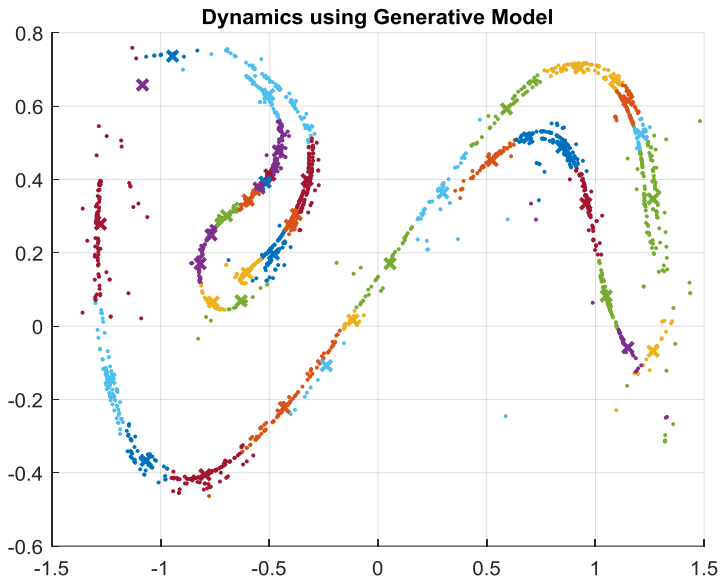

Figure 6: $m = 2000$ step Poincare map generated by successively applying the learned model with zero control input

The auto-correlations in the un-forced training data were compared to the auto-correlations of the data generate using the model. These correlations show excellent agreement over 30 time steps. The autocorrelations of the real and generated data are plotted in the following figure

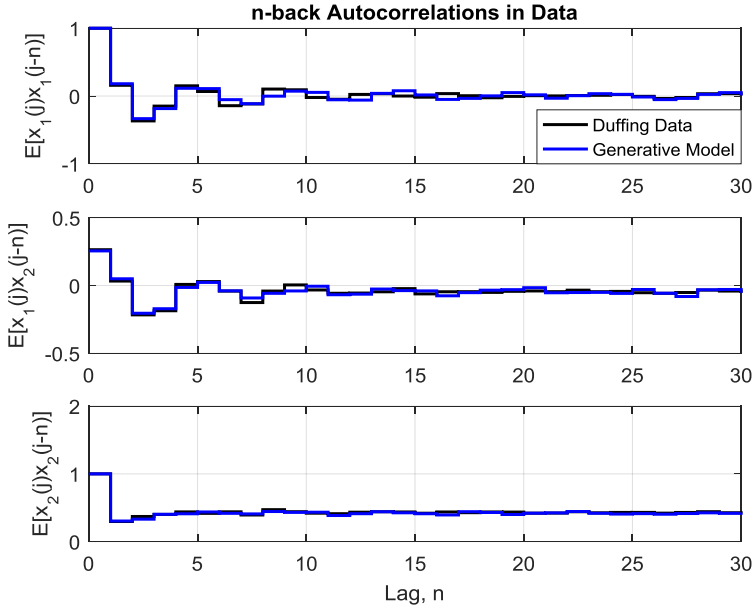

Figure 7: Autocorrelations in the discrete time series generated by the forced Duffing equation's Poincare map and the learned model

The preceding numerical tests indicate that the collection of nonlinear models learned using the EM algorithm closely approximate the dynamics of the true Poincare map. This model will be used throughout the remainder of the paper for designing the OGY and optimal targeting controllers. The model will also be used to estimate the location of the unstable periodic orbit.

4. Optimal OGY Control of an UPO

4.1 Designing the Optimal OGY Controller

The first task before the OGY method can be implemented is to locate a desired unstable orbit in the chaotic attractor. In this report, we focus on the period T orbit of the forced Duffing equation corresponding to the fixed point of the Poincare map discussed in section 2.2. We use the learned model to identify a fixed point close to the exact fixed point of the Poincare map. This was done using Matlab's fsolve function to find a zero of the following equation

$$\widehat{\boldsymbol{x}}_{FP} - \hat{\boldsymbol{f}}(\widehat{\boldsymbol{x}}_{FP}, \mathbf{0}) = \mathbf{0} \quad \text{giving the solution} \quad \widehat{\boldsymbol{x}}_{FP} = [0.1290 \quad 0.1933]^T \tag{64}$$

The fixed point located by the model is within a distance 0.0004 of the exact fixed point $\boldsymbol{x}_{FP}^{(exact)} = [0.1290 \quad 0.1937]^T$.

We proceed to design a discrete time Linear Quadratic Regulator (LQR) to stabilize this fixed point when the trajectory enters a sufficiently small neighborhood of $\widehat{\boldsymbol{x}}_{FP}$ to be controlled with small action $\|\boldsymbol{u}\|_W \leq \epsilon_u$. The modeled dynamics are linearized about $\widehat{\boldsymbol{x}}_{FP}$ using eq. (51) giving

$$\widehat{\Phi}_{FP} = D_{\boldsymbol{x}}\hat{\boldsymbol{f}}(\widehat{\boldsymbol{x}}_{FP}, \mathbf{0})$$
$$= \begin{bmatrix} -3.8137 & -4.9974 \\ -3.1477 & -4.1328 \end{bmatrix} = \begin{bmatrix} 0.7953 & 0.7709 \\ -0.6063 & 0.6369 \end{bmatrix} \begin{bmatrix} -0.0039 & 0 \\ 0 & -7.9426 \end{bmatrix} \begin{bmatrix} 0.7953 & 0.7709 \\ -0.6063 & 0.6369 \end{bmatrix}^{-1} \tag{65}$$

$$\hat{\Gamma}_{FP} = D_{\boldsymbol{u}}\hat{\boldsymbol{f}}(\widehat{\boldsymbol{x}}_{FP}, \mathbf{0}) = \begin{bmatrix} 0.3005 & -8.4941 & 2.9177 & -9.1783 & -1.1372 \\ 1.3244 & -6.7844 & 2.6736 & -8.2029 & -1.2446 \end{bmatrix} \tag{66}$$

The linearization may be compared to the exact linearization at $\widehat{\boldsymbol{x}}_{FP}$ given by

$$\Phi_{\widehat{\boldsymbol{x}}_{FP}} = \begin{bmatrix} -3.9272 & -4.9881 \\ -3.2389 & -4.1668 \end{bmatrix} \tag{65b}$$

$$\Gamma_{\widehat{\boldsymbol{x}}_{FP}} = \begin{bmatrix} 0.0919 & -8.6577 & 3.1047 & -9.4750 & -1.2691 \\ 1.0066 & -6.8181 & 2.8911 & -8.6106 & -1.3643 \end{bmatrix} \tag{66b}$$

Near the fixed point, we aim to minimize the following quadratic cost function with modeled linear dynamics in $\Delta\boldsymbol{x}_n = \boldsymbol{x}_n - \widehat{\boldsymbol{x}}_{FP}$. Without loss of generality, Q_{FP} and R_{FP} are assumed to be symmetric positive definite matrices.

$$\boxed{J^{FP} = \frac{1}{2}\sum_{n=1}^{\infty}\left[\Delta \boldsymbol{x}_n{}^T Q_{FP}\Delta \boldsymbol{x}_n + \boldsymbol{u}_n^T R_{FP}\boldsymbol{u}_n\right], \quad s.t. \quad \Delta \boldsymbol{x}_{n+1} = \hat{\Phi}_{FP}\Delta \boldsymbol{x}_n + \hat{\Gamma}_{FP}\boldsymbol{u}_n} \tag{67}$$

To solve this problem, we introduce the optimal value function and formulate Bellman's equations

$$\begin{aligned} V^*(\Delta \boldsymbol{x}_k) &= \min_{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots} \frac{1}{2}\sum_{n=k}^{\infty}\left(\Delta \boldsymbol{x}_n{}^T Q_{FP}\Delta \boldsymbol{x}_n + \boldsymbol{u}_n^T R_{FP}\boldsymbol{u}_n\right) \\ &= \min_{\boldsymbol{u}_k} \frac{1}{2}\left(\Delta \boldsymbol{x}_k{}^T Q_{FP}\Delta \boldsymbol{x}_k + \boldsymbol{u}_k^T R_{FP}\boldsymbol{u}_k\right) + V^*(\Delta \boldsymbol{x}_{k+1}) \end{aligned} \tag{68}$$

We assume that the optimal value function near the fixed point is a quadratic form

$$V^*(\Delta \boldsymbol{x}) = \tfrac{1}{2}\Delta \boldsymbol{x}^T P \Delta \boldsymbol{x}, \quad \text{for some symmetric positive definite} \quad P \in \mathbb{R}^{n\times n} \tag{69}$$

Making this substitution in the value function gives

$$V^*(\Delta \boldsymbol{x}_k) = \min_{\boldsymbol{u}_k} \frac{1}{2}\left(\Delta \boldsymbol{x}_k{}^T Q_{FP}\Delta \boldsymbol{x}_k + \boldsymbol{u}_k^T R_{FP}\boldsymbol{u}_k + \Delta \boldsymbol{x}_{k+1}{}^T P \Delta \boldsymbol{x}_{k+1}\right) \tag{70}$$

Substituting the linearized dynamics for $\Delta \boldsymbol{x}_{k+1}$ we are able to carry out the minimization explicitly

$$V^*(\Delta \boldsymbol{x}_k) = \min_{\boldsymbol{u}_k} \frac{1}{2}\left[\Delta \boldsymbol{x}_k{}^T\left(Q_{FP} + \hat{\Phi}_{FP}^T P \hat{\Phi}_{FP}\right)\Delta \boldsymbol{x}_k + 2\Delta \boldsymbol{x}_k{}^T \hat{\Phi}_{FP}^T P \hat{\Gamma}_{FP}\boldsymbol{u}_k + \boldsymbol{u}_k^T\left(R_{FP} + \hat{\Gamma}_{FP}^T P \hat{\Gamma}_{FP}\right)\boldsymbol{u}_k\right] \tag{71}$$

The above is a simple quadratic minimization with positive definite quadratic term. Setting the derivative with respect to $\boldsymbol{u}_k$ equal to zero allows us to solve for the optimal control and value function.

$$\boxed{\boldsymbol{u}_k^* = -\left(R_{FP} + \hat{\Gamma}_{FP}^T P \hat{\Gamma}_{FP}\right)^{-1}\hat{\Gamma}_{FP}^T P \hat{\Phi}_{FP}\Delta \boldsymbol{x}_k \triangleq -C_{FP}\Delta \boldsymbol{x}_k} \tag{72}$$

Substituting the form of the optimal control into the value function gives the algebraic Riccati equation which must be solved to find the value function near the fixed point. After some algebra, we have the value function

$$V^*(\Delta \boldsymbol{x}_k) = \frac{1}{2}\Delta \boldsymbol{x}_k{}^T\left[Q_{FP} + \hat{\Phi}_{FP}^T P \hat{\Phi}_{FP} - \hat{\Phi}_{FP}^T P \hat{\Gamma}_{FP}\left(R_{FP} + \hat{\Gamma}_{FP}^T P \hat{\Gamma}_{FP}\right)^{-1}\hat{\Gamma}_{FP}^T P \hat{\Phi}_{FP}\right]\Delta \boldsymbol{x}_k \tag{73}$$

Using the fact that $V^*(\Delta \boldsymbol{x}_k) = \frac{1}{2}\Delta \boldsymbol{x}_k{}^T P \Delta \boldsymbol{x}_k$, we have the algebraic Riccati equation

$$\boxed{P = Q_{FP} + \hat{\Phi}_{FP}^T P \hat{\Phi}_{FP} - \hat{\Phi}_{FP}^T P \hat{\Gamma}_{FP}\left(R_{FP} + \hat{\Gamma}_{FP}^T P \hat{\Gamma}_{FP}\right)^{-1}\hat{\Gamma}_{FP}^T P \hat{\Phi}_{FP}} \tag{74}$$

For this problem, the following design choices were made

$$Q_{FP} = diag[0.1 \quad 0.1], \quad R_{FP} = G_W = diag[2\pi \quad \pi \quad \pi \quad \pi \quad \pi] \tag{75}$$

Yielding the following LQR gain matrix to be used in the neighborhood of the fixed point

$$C_{FP} = \begin{bmatrix} -0.0091 & -0.0121 \\ 0.1830 & 0.2399 \\ -0.0666 & -0.0874 \\ 0.2072 & 0.2718 \\ 0.0282 & 0.0370 \end{bmatrix} \tag{76}$$

The controlled system in the neighborhood of the fixed point takes the form

$$\Delta \boldsymbol{x}_{n+1} = \left(\Phi_{\hat{\boldsymbol{x}}_{FP}} - \Gamma_{\hat{\boldsymbol{x}}_{FP}} C_{FP}\right) \Delta \boldsymbol{x}_n \approx \left(\hat{\Phi}_{FP} - \hat{\Gamma}_{FP} C_{FP}\right) \Delta \boldsymbol{x}_n \tag{77}$$

The actual and modeled systems with LQR feedback control have the following eigenvalues

$$eig\left(\Phi_{\hat{\boldsymbol{x}}_{FP}} - \Gamma_{\hat{\boldsymbol{x}}_{FP}} C_{FP}\right) = \{-0.1347, 0.1224\}, \quad eig\left(\hat{\Phi}_{FP} - \hat{\Gamma}_{FP} C_{FP}\right) = \{-0.1050, -0.0040\} \tag{78}$$

which are in somewhat poor agreement, but are both highly contractive and stable.

At each point $\boldsymbol{x}_n$, the OGY controller calculates a hypothetical control action using the above gain matrix $\tilde{\boldsymbol{u}}_n = -C_{FP}(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{FP})$. If the control action is sufficiently small, the control is used. Otherwise, the control is set to zero

$$\boxed{\boldsymbol{u}_n^{(OGY)} = \begin{cases} \tilde{\boldsymbol{u}}_n, & \tilde{\boldsymbol{u}}_n^T G_W \tilde{\boldsymbol{u}}_n \leq \epsilon_u^2 \\ \boldsymbol{0}, & otherwise \end{cases}} \tag{79}$$

4.2 Estimating the Time to Control

The average time to control for a chaotic system using the OGY method may be estimated using the natural measure of the controllable region. Let $\mathcal{M}$ be the Borel sigma algebra containing subsets of the phase space $\mathbb{R}^n$. Then the Sinai-Ruelle-Bowen theorem (Guckenheimer & Holmes, 1983) lets us define the natural measure of a set $C \in \mathcal{M}$ as the fraction of the time a trajectory starting in the basin of attraction of the chaotic attractor spends in C. If $\boldsymbol{x}_0 \in U$, the basin of attraction for chaotic attractor A, then define

$$\mu(C; \boldsymbol{x}_0) = \lim_{m \to \infty} \frac{1}{m} \sum_{n=1}^{m} \mathbf{1}_C\left(\boldsymbol{f}^n(\boldsymbol{x}_0, \boldsymbol{0})\right), \quad \mathbf{1}_C(\boldsymbol{x}) \triangleq \begin{cases} 1 & \boldsymbol{x} \in C \\ 0 & \boldsymbol{x} \notin C \end{cases}, \quad \boldsymbol{f}^n(\boldsymbol{x}_0, \boldsymbol{0}) \triangleq \boldsymbol{f}(\boldsymbol{f}^{n-1}(\boldsymbol{x}_0, \boldsymbol{0}), \boldsymbol{0}) \tag{80}$$

If $\mu(C;\boldsymbol{x}_0)=\mu(C)$ is constant for almost every $\boldsymbol{x}_0$ with respect to the Lebesgue measure in the basin U of the chaotic attractor then we call $\mu(C)$ the "natural measure" of C associated with the attractor. By the theorem, this will be true for hyperbolic attractors A, where $A=\bigcap_{n\geq 1}\boldsymbol{f}^n(U,\mathbf{0})$ and $\boldsymbol{f}^n(U,\mathbf{0})\subset U\ \forall n\geq 1$. The measure of a set C is easily approximated numerically by starting from any initial condition in U and counting the number of time the resulting trajectory lands in C. It is easily shown that the Poincare map with zero control preserves the natural measure $\mu\big(\boldsymbol{f}^{-1}(C,\mathbf{0})\big)=\mu(C)$. We understand $\boldsymbol{f}^{-1}(C,\mathbf{0})$ to be the preimage of C under the Poincare map with zero control.

Let R_c be the region of phase space around the desired orbit where trajectories are effectively captured by the OGY control scheme. If we consider and arbitrary trajectory starting at unknown $\boldsymbol{x}_0$ in the basin of attraction of the attractor, at each step the probability of capture is $\mu(R_c)$. We also note that the ergodic theorem for Markov processes on continuous state spaces tells us that any continuous initial distribution μ_0 over the attractor's basin will converge to the unique limiting distribution given by the natural measure $\lim_{n\to\infty}\mu_0\big(\boldsymbol{f}^{-n}(C,\mathbf{0})\big)=\mu(C)$ under successive applications of the Poincare map. Therefore, even if the system starts from a nearly certain state, its probability of entering the region R_c at each step will tend to $\mu(R_c)$ as $n\to\infty$. The rate of convergence to the limiting distribution is exponential (Koralov & Sinai, Yakov, 2007). Therefore, the probability to be captured in exactly τ steps is geometrically distributed

$$P_c(\tau)=\big(1-\mu(R_c)\big)^{\tau-1}\mu(R_c) \tag{81}$$

The expected value of the time to capture $\langle\tau\rangle=\mathbb{E}_{\tau\sim P_c}[\tau]$ is found

$$\langle\tau\rangle=\sum_{\tau=1}^{\infty}\tau\big(1-\mu(R_c)\big)^{\tau-1}\mu(R_c)=-\mu(R_c)\sum_{\tau=1}^{\infty}\frac{d}{d\mu(R_c)}\big(1-\mu(R_c)\big)^{\tau} \tag{82}$$

Since the power series is absolutely and uniformly convergent, we can exchange the order of summation and differentiation

$$\langle\tau\rangle=-\mu(R_c)\frac{d}{d\mu(R_c)}\left[\frac{1}{\mu(R_c)}-1\right]\quad\Longrightarrow\quad\boxed{\langle\tau\rangle=\frac{1}{\mu(R_c)}} \tag{83}$$

This means that the mean time to control is inversely proportional to the natural measure of the OGY controllable region. As discussed in (Otto, 2017b), the natural measure of the region scales with the size of R_c according to the local information fractal dimensions along the eigenspaces of the Poincare map at $\hat{\boldsymbol{x}}_{FP}$. The controllable region R_c in the case of LQR will be the ellipsoid defined by

$$(\boldsymbol{x}-\hat{\boldsymbol{x}}_{FP})^T C_{FP}^T G_W C_{FP}(\boldsymbol{x}-\hat{\boldsymbol{x}}_{FP})\leq\epsilon_u^2,\quad\text{where}\quad C_{FP}^T G_W C_{FP}\quad\text{has real pos. eigs.}\quad\{\rho_j\}_{j=1}^{n} \tag{84}$$

The eigenvectors of $C_{FP}^T G_W C_{FP}$ are the principal axes of the ellipsoid with semi-principal axis lengths $\epsilon_u/\sqrt{\rho_j}, j=1,2,\dots,n$.

For the case of forced Duffing equation, the average time to control has been determined experimentally. A range of maximum control action sizes ϵ_u were chosen. For each choice of ϵ_u, 10 trials were conducted starting from random initial conditions in $[-1.5,1.5] \times [-0.6,0.8]$. The time to control was determined for each trial when the control system was able to keep the trajectory under control inside R_c for 10 consecutive iterations. The times to control are plotted in the following figure

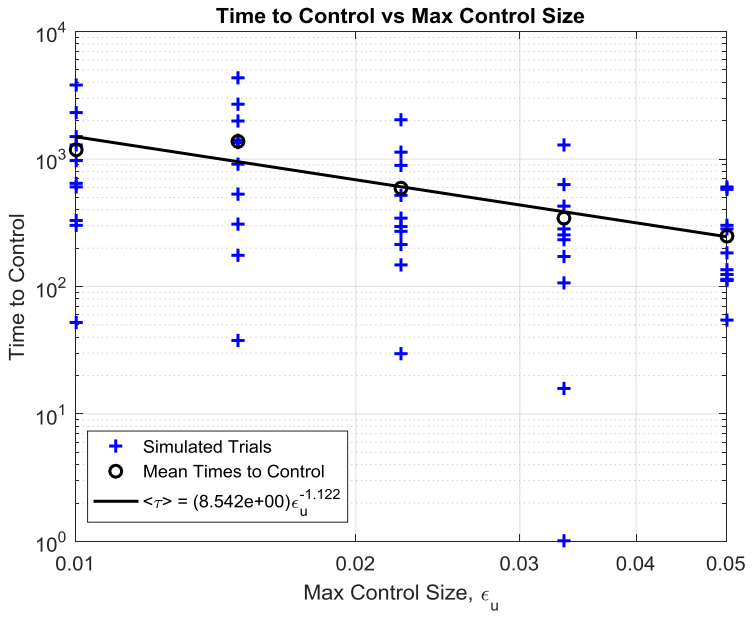

Figure 8: Experimental times to control using LQR for OGY stabilization of the fixed point. No targeting control was used.

The experimentally determined time to control scales with the maximum control size according to

$$\boxed{\langle \tau \rangle^{(OGY-LQR)} = 8.542\ (\epsilon_u)^{-1.122} \quad \Longrightarrow \quad \mu\left(R_c^{(OGY-LQR)}\right) = 0.117\ (\epsilon_u)^{1.122}} \tag{85}$$

We see that it may take hundreds or thousands of iterations to achieve control using the OGY method alone with small control. Targeting will be used to reduce the time to control using small control actions.

5. Simple Targeting Control

Taking up the problem of targeting control, it is natural to consider trajectories from the training data which pass close to the desired orbit. We may take as a nominal path, an un-forced trajectory passing close to the fixed point and design a neighboring optimal controller. When a point comes sufficiently close to any of the points along the nominal targeting path, the neighboring optimal controller is activated and guides the trajectory along it towards the desired orbit. If the time scale associated with mixing in the attractor is small, the nominal path will spread very quickly in backwards time. Furthermore, the nominal target path will be distributed roughly according to the natural measure. Therefore, there will be more targeting path points located in areas of high natural measure where trajectories are likely to spend more

time. This improves the chances of capturing trajectories starting from random initial conditions. It is also possible to create a tree of targeting paths by considering trajectories in the training data which come close to the points already part of the path. Once a point has been guided along the target path to the neighborhood of the fixed point, the optimal OGY controller designed using LQR is activated.

The simple approach using a single targeting path consisting of $M = 50$ points $\{\bar{\boldsymbol{x}}_n\}_{n=1}^{M}$ from the unforced training data set is taken here. The point from the unforced training set requiring the least control using the OGY-LQR method designed in section 4.1 was selected as the base point for the targeting path. Of course the control magnitude used to stabilize this point is a limiting value for the effectiveness of the this method. For if the maximum allowable control is smaller than the control used to stabilize the base point, it will not be helpful to follow the targeting path back to the base point. The following figure shows the training data Poincare map with the selected target path points. We see that even with $M = 50$ targeting point, we achieve good coverage of the attractor.

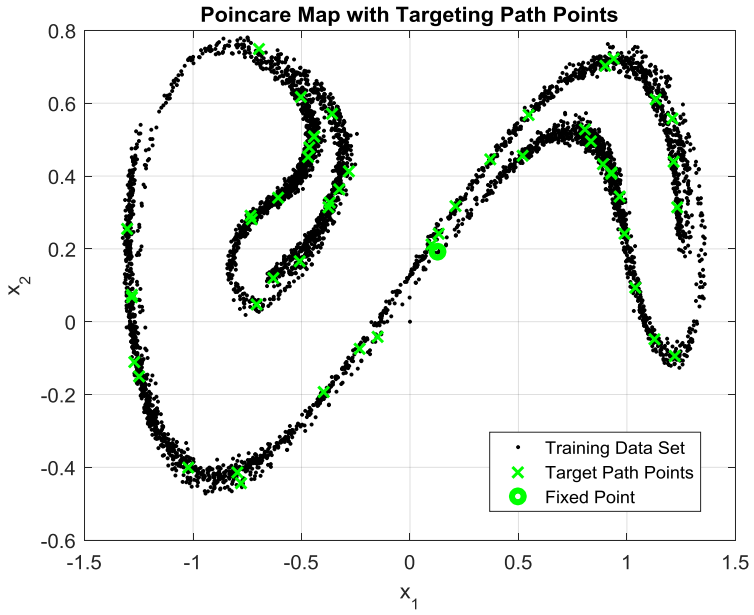

Figure 9: Poincare map illustrating the distribution of targeting path points

Once the nominal targeting path is established by tracing the base point back M steps, a neighboring optimal controller is designed. In designing the cost function, we wish to weight the state more heavily as the trajectory approaches the base point. When the trajectory is far away, we are not as interested in aggressively bringing the point onto the target path as we are in having the largest possible controllable regions around the path. In order to achieve this, we introduce a geometric growth factor $\gamma \geq 1$ into the cost function. Let $\Delta\boldsymbol{x}_n = \boldsymbol{x}_n - \bar{\boldsymbol{x}}_n$ and formulate the finite horizon optimal control problem using the linearized model dynamics in the neighborhood of each targeting point $\Delta\boldsymbol{x}_{n+1} = \hat{\Phi}_n \Delta\boldsymbol{x}_n + \hat{\Gamma}_n \boldsymbol{u}_n$.

$$J = \frac{1}{2}\sum_{n=1}^{M-1}\left[\gamma^{n-M}\Delta\boldsymbol{x}_n^T Q \Delta\boldsymbol{x}_n + \boldsymbol{u}_n^T R \boldsymbol{u}_n\right] + \frac{1}{2}\Delta\boldsymbol{x}_M^T Q \Delta\boldsymbol{x}_M \tag{86}$$

Consider the optimal value function and formulate Bellman's equations

$$V_k^*(\Delta \boldsymbol{x}_k) = \min_{\boldsymbol{u}_k,\dots,\boldsymbol{u}_{M-1}} \frac{1}{2}\sum_{n=k}^{M-1}[\gamma^{n-M}\Delta \boldsymbol{x}_n^T Q \Delta \boldsymbol{x}_n + \boldsymbol{u}_n^T R \boldsymbol{u}_n] + \frac{1}{2}\Delta \boldsymbol{x}_M^T Q \Delta \boldsymbol{x}_M\,, \quad k = 1,2,\dots,M-1 \tag{87}$$

$$V_k^*(\Delta \boldsymbol{x}_k) = \min_{\boldsymbol{u}_k}\left[\frac{1}{2}\gamma^{k-M}\Delta \boldsymbol{x}_k^T Q \Delta \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T R \boldsymbol{u}_k + V_{k+1}^*(\Delta \boldsymbol{x}_{k+1})\right], \quad k = 1,2,\dots,M-1 \tag{88}$$

The value function at the final time is

$$V_M^*(\Delta \boldsymbol{x}_M) = \frac{1}{2}\Delta \boldsymbol{x}_M^T Q \Delta \boldsymbol{x}_M \tag{89}$$

Try a value function taking the quadratic form $V_k^*(\Delta \boldsymbol{x}_k) = \frac{1}{2}\Delta \boldsymbol{x}_k^T P_k \Delta \boldsymbol{x}_k$ with $\boxed{P_M = Q}$. Substituting into Bellman's equations gives

$$V_k^*(\Delta \boldsymbol{x}_k) = \min_{\boldsymbol{u}_k}\frac{1}{2}\left[\gamma^{k-M}\Delta \boldsymbol{x}_k^T Q \Delta \boldsymbol{x}_k + \boldsymbol{u}_k^T R \boldsymbol{u}_k + \Delta \boldsymbol{x}_{k+1}^T P_{k+1}\Delta \boldsymbol{x}_{k+1}\right] \tag{90}$$

Substituting the linearized dynamics, the minimization can be carried out explicitly. After some algebra, the following is obtained

$$\begin{aligned} V_k^*(\Delta \boldsymbol{x}_k) = \min_{\boldsymbol{u}_k}\frac{1}{2}\big[&\Delta \boldsymbol{x}_k^T\big(\gamma^{k-M}Q + \widehat{\Phi}_k^T P_{k+1}\widehat{\Phi}_k\big)\Delta \boldsymbol{x}_k + 2\Delta \boldsymbol{x}_k^T\widehat{\Phi}_k^T P_{k+1}\widehat{\Gamma}_k \boldsymbol{u}_k \\ &+ \boldsymbol{u}_k^T\big(R + \widehat{\Gamma}_k^T P_{k+1}\widehat{\Gamma}_k\big)\boldsymbol{u}_k\big] \end{aligned} \tag{91}$$

The above equation is a quadratic function with positive definite quadratic term. The minimization is carried out by setting the derivative with respect to $\boldsymbol{u}_k$ equal to zero. Some manipulation gives the form of the optimal control applied at each point along the target path

$$\boxed{\boldsymbol{u}_k^* = -\big(R + \widehat{\Gamma}_k^T P_{k+1}\widehat{\Gamma}_k\big)^{-1}\widehat{\Gamma}_k^T P_{k+1}\widehat{\Phi}_k \Delta \boldsymbol{x}_k \triangleq -C_k \Delta \boldsymbol{x}_k}, \quad k = 1,2,\dots,M-1 \tag{92}$$

Substituting the form of the optimal control back into Bellman's equations gives us the update equation for P_k given P_{k+1}. After some algebra, we are left with

$$V_k^*(\Delta \boldsymbol{x}_k) = \frac{1}{2}\Delta \boldsymbol{x}_k^T\left[\gamma^{k-M}Q + \widehat{\Phi}_k^T P_{k+1}\widehat{\Phi}_k - \widehat{\Phi}_k^T P_{k+1}\widehat{\Gamma}_k\big(R + \widehat{\Gamma}_k^T P_{k+1}\widehat{\Gamma}_k\big)^{-1}\widehat{\Gamma}_k^T P_{k+1}\widehat{\Phi}_k\right]\Delta \boldsymbol{x}_k \tag{93}$$

Giving the following update equation

$$\boxed{P_k = \gamma^{k-M}Q + \widehat{\Phi}_k^T P_{k+1}\widehat{\Phi}_k - \widehat{\Phi}_k^T P_{k+1}\widehat{\Gamma}_k\big(R + \widehat{\Gamma}_k^T P_{k+1}\widehat{\Gamma}_k\big)^{-1}\widehat{\Gamma}_k^T P_{k+1}\widehat{\Phi}_k}, \quad k = 1,2,\dots,M-1 \tag{94}$$

In our application to the forced Duffing equation, the following design choices were made

$$\gamma = 1.1, \quad Q = C_{FP}^{T} G_W C_{FP} + (0.7) I_2, \quad R = (10^{-3}) I_5 \tag{95}$$

The same time to control experiment was performed using this targeting control scheme. The resulting times to control are plotted in the following figure. The times to control have been reduced by almost two orders of magnitude over the LRQ-OGY method applied only at the fixed point. In the following figure, the outlier has been removed to better fit the overall linear trend.

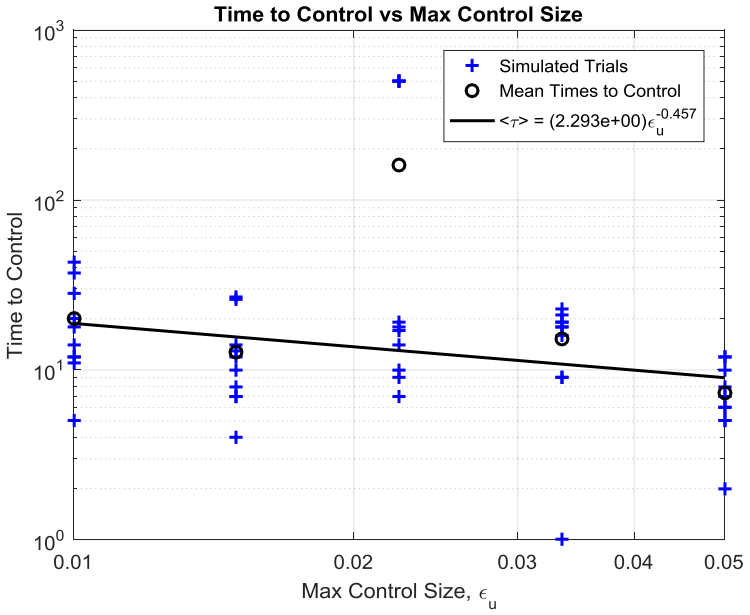

Figure 10: Time to control using neighboring optimal controller along 50 targeting path points. OGY-LQR was used to stabilize the fixed point.

The experimentally determined mean time to control scales with the maximum control action size according to

$$\boxed{\langle \tau \rangle^{(50\ Pt.\ Path)} = 2.293\ (\epsilon_u)^{-0.457} \quad \Longrightarrow \quad \mu\left(R_c^{(50\ Pt.\ Path)}\right) = 0.436\ (\epsilon_u)^{0.457}} \tag{96}$$

Another study which I did not have enough time to perform for this report, is to determine the scaling of time to control with the number of targeting path points used. This is done in (Otto, 2017b) for a similar type of sub-optimal control, though I expect the scaling exponents to be roughly the same.

The following figures show an example of a trajectory using the optimal controller starting from a random initial condition and using $\epsilon_u = 0.01$.

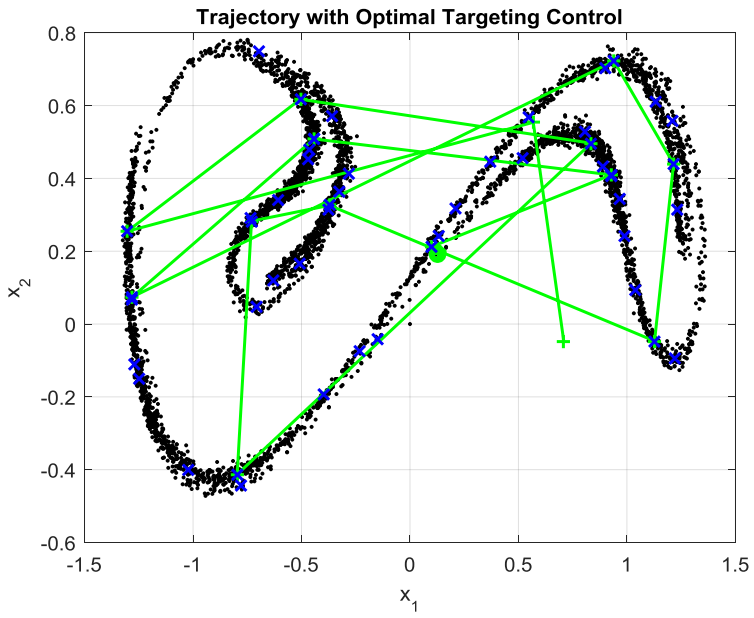

Figure 11: Example trajectory of the Poincare map with neighboring optimal targeting control near the existing path trajectory. A random initial condition was used

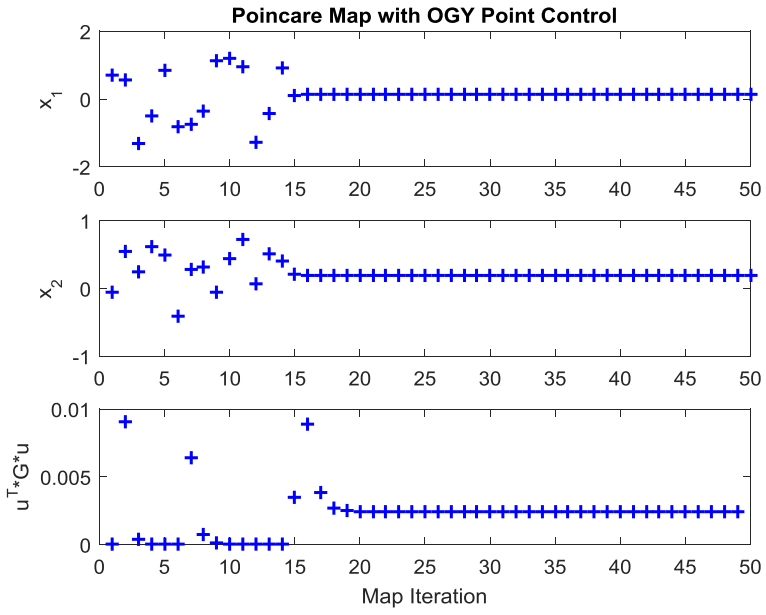

Figure 12: Example trajectory Poincare map with optimal control. The control action size is shown.

We see that the trajectory is captured very quickly by the neighboring optimal targeting controller. The spikes in the control occur when the trajectory skips points along the targeting path. This happens when the targeting path passes closely enough to itself that small control action can be used to jump forward on the path. The final spike happens when the OGY-LQR controller is activated in the neighborhood of the fixed point. Observe that because the approximate fixed point does not agree perfectly with the exact fixed point, the control does not go to zero when $\widehat{\boldsymbol{x}}_{FP}$ is stabilized.

6. Optimal Targeting Control using Fitted Value Iteration

We now approach the problem of designing a globally optimal controller. This is made difficult by the high degree of nonlinearity in the problem as well as a nonlinear albeit quadratic constraint on the control. Because of this, we cannot assume that the value function or optimal control take any special form over the domain. This leaves two options, either learn a value function or the optimal control itself over the entire attractor. The value function is lower-dimensional and can be used together with the dynamics model to find the optimal control action at any point. The approach taken here will be to learn a value function globally by interpolating over a collection of points. An iterative process will be used to update the value function at each point until the function converges. Once the value function is known, optimal control is accomplished by performing a constrained minimization of the value function under the modeled dynamics at each step. This process is expensive and several methods will be proposed to reduce the cost. Such methods will be the subject of future work. In this report, we focus on the core method using fitted value iteration and the full minimization at each step.

The following infinite horizon constrained optimal control problem is to be solved over the chaotic attractor. A geometric decay factor $0 < \beta \leq 1$ was introduced in the cost function to accelerate convergence and prevent the value function from exploding if trajectories do not approach the fixed point sufficiently quickly or precisely. It was found however to be unnecessary and is set $\beta = 1$ for the application discussed here. The problem is solved including the factor to retain generality for possible future use.

$$\boxed{\text{minimize } J = \frac{1}{2}\sum_{n=1}^{\infty} \beta^{n-1}[(\boldsymbol{x}_n - \boldsymbol{x}_{FP})^T Q(\boldsymbol{x}_n - \boldsymbol{x}_{FP}) + \boldsymbol{u}_n^T R \boldsymbol{u}_n] \quad s.t. \quad \boldsymbol{u}_n^T G_W \boldsymbol{u}_n \leq \epsilon_u^2} \tag{97}$$

Subject to the modeled dynamics $\boldsymbol{x}_{n+1} = \hat{\boldsymbol{f}}(\boldsymbol{x}_n, \boldsymbol{u}_n)$. The optimal value function is introduced

$$V^*(\boldsymbol{x}) = \min_{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots} \frac{1}{2}\sum_{n=1}^{\infty} \beta^{n-1}[(\boldsymbol{x}_n - \boldsymbol{x}_{FP})^T Q(\boldsymbol{x}_n - \boldsymbol{x}_{FP}) + \boldsymbol{u}_n^T R \boldsymbol{u}_n], \quad \boldsymbol{x}_1 = \boldsymbol{x} \tag{98}$$

Separating the first term in the above sum and using the modeled dynamics allows us to formulate Bellman's equations

$$\boxed{V^*(\boldsymbol{x}) = \min_{\boldsymbol{u}^T G_W \boldsymbol{u} \leq \epsilon_u^2} \frac{1}{2}[(\boldsymbol{x} - \boldsymbol{x}_{FP})^T Q(\boldsymbol{x} - \boldsymbol{x}_{FP}) + \boldsymbol{u}^T R \boldsymbol{u}] + \beta V^*\left(\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})\right)} \tag{99}$$

The value function over the entire attractor may be learned through a reinforcement learning algorithm called fitted value iteration. A subset of the training points $\{\overline{\boldsymbol{x}}_j\}_{j=1}^M$ will be used to define the value function though either regression or interpolation of the values at each point $V_j^* = V^*(\overline{\boldsymbol{x}}_j)$. Given a previous estimate of $V^*(\boldsymbol{x})$ over the attractor, the value function at the points $\{\overline{\boldsymbol{x}}_j\}_{j=1}^M$ is updated by performing the following minimization

$$\boxed{V_j^* \leftarrow \min_{\boldsymbol{u}^T G_W \boldsymbol{u} \le \epsilon_u^2} \frac{1}{2}\left[\left(\bar{\boldsymbol{x}}_j - \boldsymbol{x}_{FP}\right)^T Q\left(\bar{\boldsymbol{x}}_j - \boldsymbol{x}_{FP}\right) + \boldsymbol{u}^T R \boldsymbol{u}\right] + \beta V^*\left(\hat{\boldsymbol{f}}\left(\bar{\boldsymbol{x}}_j, \boldsymbol{u}\right)\right), \quad \forall j = 1,2,\dots,M} \tag{100}$$

The new optimal value function $V^*(\boldsymbol{x})$ is fit from the values V_j^* using regression or interpolation over the attractor. The process is repeated until the value function converges. Once the value function is determined, the optimal control at a point $\boldsymbol{x}$ is found by solving the constrained minimization problem

$$\boxed{\boldsymbol{u}^* = \underset{\boldsymbol{u}^T G_W \boldsymbol{u} \le \epsilon_u^2}{\operatorname{argmin}} \frac{1}{2}[(\boldsymbol{x} - \boldsymbol{x}_{FP})^T Q(\boldsymbol{x} - \boldsymbol{x}_{FP}) + \boldsymbol{u}^T R \boldsymbol{u}] + \beta V^*\left(\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})\right)} \tag{101}$$

$M = 500$ training points $\{\bar{\boldsymbol{x}}_j\}_{j=1}^{500}$ were chosen from the unforced training data set. These points formed a trajectory passing close the desired fixed point, though this particular choice is not essential. The following design choices were made

$$\beta = 1, \quad Q = C_{FP}^T G_W C_{FP} + (0.7) I_2, \quad R = (0.1) I_5, \quad \epsilon_u = 0.01 \tag{102}$$

The optimal value function was fit using natural neighbor interpolation over $\{\bar{\boldsymbol{x}}_j\}_{j=1}^{500}$. Fitted value iteration using the above parameters produced the value function plotted in the following figure. The minimization problem at each point was solved using Matlab's fmincon().

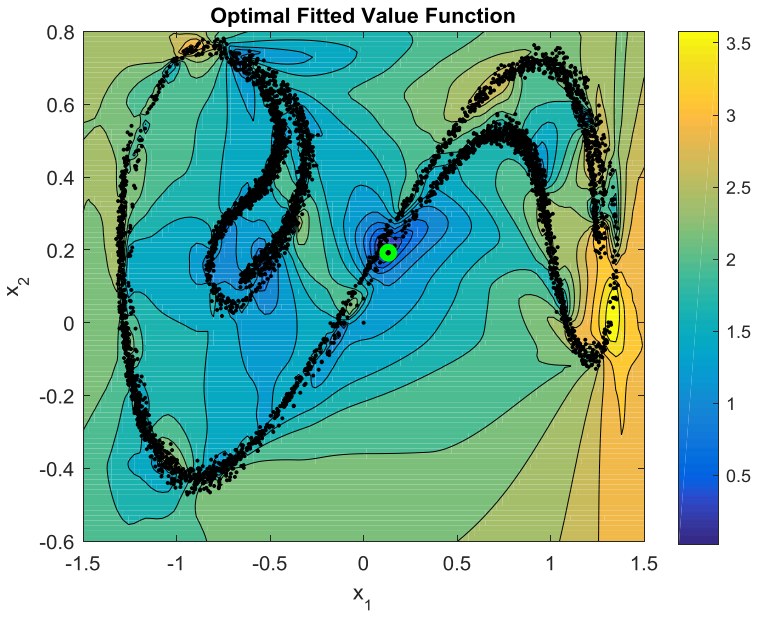

Figure 13: Interpolated optimal value function found using fitted value iteration and $\epsilon_u = 0.01$

This value function was used to find the optimal targeting control input by solving the full minimization with fmincon() at each step. Once the trajectory entered a sufficiently small neighborhood of the fixed point, the OGY-LQR controller was activated to stabilize the fixed point. The same numerical

experiment was done to estimate the average times to control as the maximum control action size was varied. The same optimal value function was used for each trial even though the maximum control size was changed. The times to control using this method are plotted in the following figure.

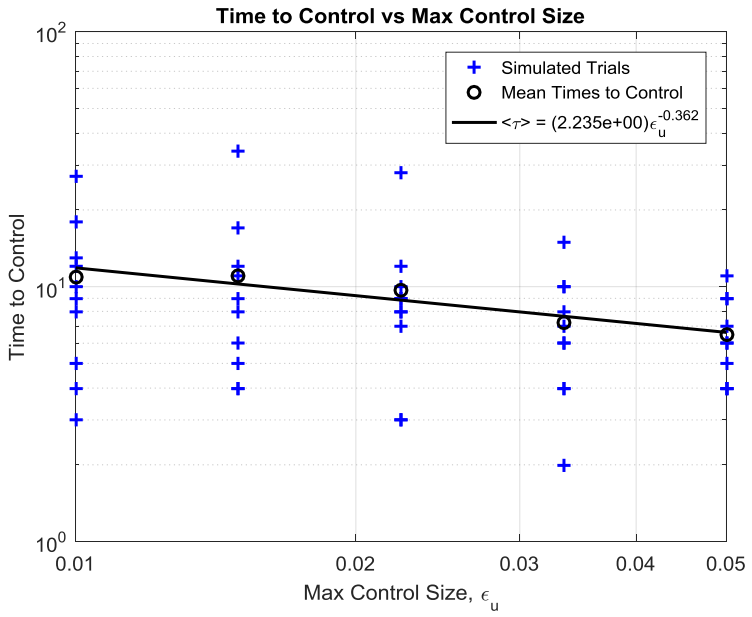

Figure 14: Time to control using modeled value function minimization at each step. Constrained minimizations solved with fmincon(). OGY-LQR control used to stabilize the fixed point.

The experimentally determined mean time to control scales with the maximum control action size according to

$$\boxed{\langle \tau \rangle^{(V-Fun)} = 2.235\ (\epsilon_u)^{-0.362} \quad \Longrightarrow \quad \mu\left(R_c^{(V-Fun)}\right) = 0.448\ (\epsilon_u)^{0.362}} \tag{103}$$

The following figure shows an example of a trajectory using the optimal controller starting from a random initial condition with $\epsilon_u = 0.01$. The forced duffing equation was simulated with the optimal control action computed using the optimal value function under modeled dynamics.

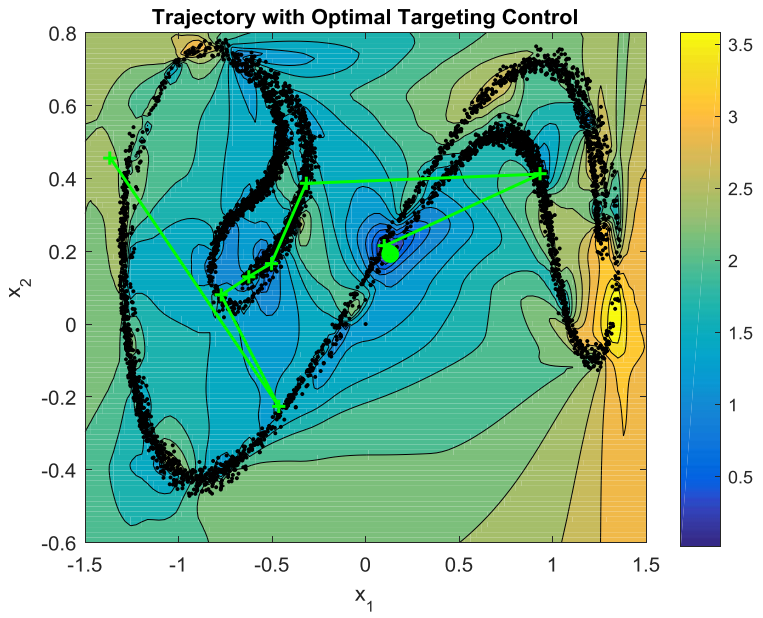

Figure 15: Example trajectory of the forced Duffing Equation with optimal control computed by minimizing the value function under modeled dynamics at each point

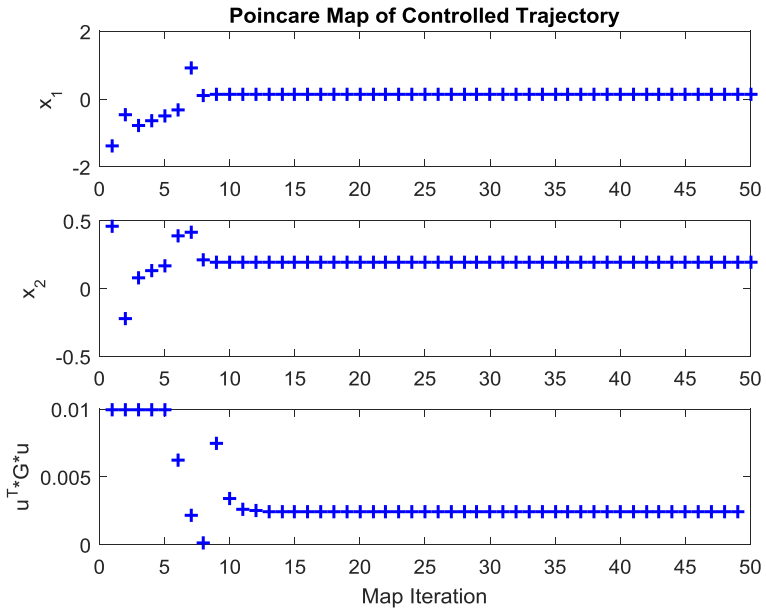

Figure 16: Example trajectory Poincare map with optimal control. The control action size is shown.

We observe that the trajectory is quickly stabilized to the fixed point $\widehat{\boldsymbol{x}}_{FP}$. However, because the approximate fixed point does not agree perfectly with the exact fixed point, the control does not go to zero when $\widehat{\boldsymbol{x}}_{FP}$ is stabilized. The following figure plots the optimal value function along the above trajectory. We see that the value function decreases almost monotonically with the exception of a jump after 6 steps. This is most likely due to imperfect modeling and interpolated optimal value function.

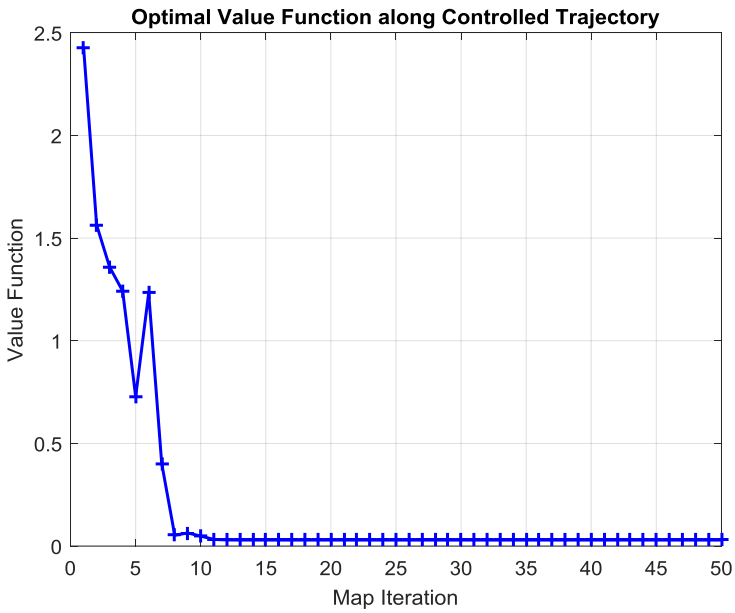

Figure 17: Optimal value function at points along the controlled trajectory. Nearly monotonic decrease is observed.

The method of optimal control proposed here achieves smaller mean times to control than the method using target path points. However, the minimization needed to determine the optimal control at each step is costly. Therefore, I will propose two alternatives to performing the full minimization at each step.

The first option is to fit the value function using twice-differentiable local nonlinear models. This could be done using a collection of kernel regression models fit using the EM algorithm discussed in section 3. Alternatively, spline interpolation or a Shephard surface might be used. Since the dynamics model and value function are twice differentiable, it is possible to form an inexpensive quadratic approximation of the value function at each point. We may minimize this quadratic function instead of the full value function to find an approximate optimal control. The constraint that $\boldsymbol{u}^T G_W \boldsymbol{u} \leq \epsilon_u^2$ may also be relaxed to a collection of linear inequality constraints. This will force the control to be inside the rectangular prism with the same semi-principal axes as the ellipsoid $\boldsymbol{u}^T G_W \boldsymbol{u} \leq \epsilon_u^2$. If linear inequality constraints are used together with a quadratic approximation of the value function, the optimization becomes an inexpensive quadratic program.

The second option is to use the optimal value function to obtain the optimal control actions over a much larger collection of training data points. From this expanded set of points, we may learn the optimal control function directly using a collection of nonlinear models and the EM algorithm. Training this model would proceed in exactly the same way as learning to model the dynamics. To find the optimal control at any new point, we need only evaluate the model a single time. This approach, though costly to train, is fast to evaluate. It will be a subject for future research.

8. Conclusion

In this paper an entirely data-driven approach to modeling and optimal targeting control is developed and applied to stabilize an unstable periodic orbit of the forced Duffing equation. The methods described herein are general enough to be applied to chaotic and hyperchaotic dynamical systems evolving in many dimensions – provided that there is enough training data to learn an accurate model. Discrete time chaotic dynamical systems present challenges for efficient system identification. Global models with enough capacity to represent the highly nonlinear dynamics are also expansive to evaluate and tend to over-fit the data. A collection of smaller nonlinear models are fit to the data using local nonlinear kernel regression. A Bayesian framework was introduced in order to quickly assign new points to the models most likely to capture their dynamics. The collection of nonlinear models was trained using minimum cross entropy and the EM algorithm. The models were fast to evaluate and were used to design the control systems present in this report. This modeling approach was demonstrated in the Poincare map of the forced Duffing equation, where it was able to accurately predict the dynamics several steps into the future. The method was also used generatively to produce a new Poincare map with only the modeled dynamics. The time series corresponding to this map agreed closely with the actual forced Duffing equation in terms of its autocorrelations over many time steps. Another advantage of this local Bayesian modeling approach is that we also learn a probability distribution which approximates the data using a Gaussian mixture. The Gaussian mixture model can be used to identify regions of interest or possibly to reconstruct the underlying data manifolds and analyze its topology. Future work on this method will involve fitting Gaussian mixtures in high-dimensional nonlinear feature space in order to capture curved and filamented densities locally in the original phase space. As discussed in section 3.3, it is possible to fit models to data with stochastic state-dependent control. This will enable an iterative design process for nonlinear mixture models and optimal controllers and will be a subject of future work.

The learned model of the dynamics was used to locate an unstable periodic orbit corresponding to a fixed point of the Poincare map. This orbit was stabilized with the OGY method and a Linear Quadratic Regulator (LQR) designed using the model's linearization. In accordance with the philosophy of the OGY method, LQR control was activated only when the trajectory passes sufficiently close to the desired orbit to be stabilized with small control action. The mean time to control was investigated theoretically and experimentally. A numerical experiment was conducted to determine how the mean time to control scales with the maximum allowable control action.

After designing the OGY-LQR controller to stabilize the fixed point, the problem of optimal targeting control was approached in two ways. In the first method, an existing trajectory from the training data was used as a targeting path to guide neighboring points to a small neighborhood of the desired fixed point. Once the trajectory was in a sufficiently small neighborhood, OGY-LQR control took over. A neighboring optimal controller was designed to capture points near a targeting path consisting of 50 unforced training data points. This pragmatic technique proved to be highly successful and reduced the experimental time to control by nearly two orders of magnitude with little added computational cost. It is extensible to trees of targeting path points which would surely result in further reductions in time to control.

The second method for targeting approaches the problem from the standpoint of global optimal control using reinforcement learning. Fitted value iteration is used to learn an optimal value function over

the entire chaotic attractor. Optimal control requires solving a constrained optimization problem at each step to minimize the value function under the modeled dynamics. Numerical experiments show that this method produces the smallest mean times to control. However, as it is currently implemented, it is very computationally expensive. Future work on this method will focus on reducing the computational cost associated with determining the optimal control actions in real time. To this end, I propose using the optimal fitted value function to pre-calculate the optimal control actions at a large collection of training data points. A mixture of nonlinear models may then be fit to the optimal control in precisely the same way the dynamics were modeled. To execute the optimal control action at any point in the attractor, the model is simply evaluated at the point. This method therefore defines an optimal controller everywhere in the chaotic attractor which is inexpensive to use in real time.

The modeling and control framework outlined in this report may also be extensible to optimal nonlinear state estimation. The mixture of nonlinear models fit to the data gives us local linearizations and disturbance covariance matrices for free. Therefore, it may be possible to implement an extended or quasilinear Kalman filter based on the modeled dynamics. The multiple model estimation technique might also be used to pick between filters designed for each local sub-model. Such an approach will be taken up in future work and used to do data-driven optimal state estimation and control of chaotic dynamical systems.

References

Aejaz, S. (2004). Modeling Chaos in the Heart. *National Conference on Emerging Technologies*, 88–92. Retrieved from http://szabist.edu.pk/ncet2004/Docs/Session V Paper No 2 (P 88-92).pdf

Arecchi, F. T., & Meucci, R. (2008). Chaos in Lasers. In *Scholarpedia*. Retrieved from http://www.scholarpedia.org/article/Chaos_in_lasers

Babloyantz, a, & Destexhe, a. (1986). Low-dimensional chaos in an instance of epilepsy. *Proceedings of the National Academy of Sciences of the United States of America*, *83*(10), 3513–7. http://doi.org/10.1073/pnas.83.10.3513

Boccaletti, S., Grebogi, C., Lai, Y., Mancini, H., & Maza, D. (2000). The control of chaos : theory and applications. *Physics Reports*, *329*, 103–197. http://doi.org/10.1016/S0370-1573(99)00096-4

Creanga, D., Nadejde, C., & Gasner, P. (2011). Dynamical analysis of heart beat from the viewpoint of chaos theory. *Romanian Journal of Physics*, *56*, 177–184. Retrieved from http://www.nipne.ro/rjp/2011_56_1-2.html

Epureanu, B. I., & Dowell, E. H. (2000). Optimal Multi-Dimensional OGY Controller. *Physica D: Nonlinear Phenomena*, *139*(1-2), 87–96. http://doi.org/10.1016/S0167-2789(99)00201-8

Guckenheimer, J., & Holmes, P. (1983). *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (1st ed.). New York: Springer.

Iasemidis, L. D., & Sackellares, J. C. (1996). REVIEW : Chaos Theory and Epilepsy. *The Neuroscientist*, *2*(2), 118–126. http://doi.org/10.1177/107385849600200213

Kanamaru, T. (2008). Duffing Oscillator. In *Scholarpedia*. Retrieved from http://www.scholarpedia.org/article/Duffing_oscillator

Koralov, L. B., & Sinai, Yakov, G. (2007). *Theory of Probability and Random Processes*. Springer-Verlag Berlin Heidelberg. http://doi.org/10.1007/978-3-540-68829-7

Ott, E., Grebogi, C., & Yorke, J. a. (1990). Controlling chaos. *Physical Review Letters*, *64*(11), 1196–1199. http://doi.org/10.1103/PhysRevLett.64.1196

Otto, S. (2017a). An EM Algorithm for Learning Local Dynamics Models In a Bayesian Framework.

Otto, S. (2017b). Controlling Chaos A Review of Methods with Application to the Forced Duffing Equation.

Takens, F. (1981). Detecting strange attractors in turbulence. In D. Rand & L.-S. Young (Eds.), *Dynamical Systems and Turbulence, Warwick 1980: Proceedings of a Symposium Held at the University of Warwick 1979/80* (pp. 366–381). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/BFb0091924