LEVERAGING DIFFERENTIATION OF PERSISTENCE DIAGRAMS FOR PARAMETER SPACE OPTIMIZATION AND DATA ASSIMILATION

By

Maxwell Chumley

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Mechanical Engineering—Doctor of Philosophy
Computational Mathematics, Science and Engineering—Dual Major

2025

ABSTRACT

Persistent homology, the flagship tool from Topological Data Analysis (TDA) has been successfully utilized in many different domains despite the absence of a differentiation framework. Only recently a differential calculus has been defined on the space of persistence diagrams thus unlocking new possibilities for combining persistence with powerful solvers and optimizers. This work explores harnessing persistence differentiation for topologically driven data assimilation and for optimally navigating the parameter space of dynamical systems. Specifically, in Chapter 1, I give an overview of this thesis and present background on optimization and persistence optimization. In Chapter 2, I introduce a new topological data assimilation framework, and demonstrate the capabilities of this new method. In Chapter 3, I show how persistence-based cost functions can be constructed and used to optimally traverse the parameter space of a dynamical system. The cost functions are designed by specifying criteria that correspond to the structure of a desirable target persistence diagram while penalizing undesirable persistence features. Other applications of persistent homology are also presented in Chapter 4 where a texture analysis pipeline was developed to quantify specific features of a texture using TDA. Finally, in Chapter 5, I present a time delay framework for modeling metabolic oscillations in Yeast cells and numerical methods are used to locate parameters of the system that lead to limit cycles.

This dissertation is dedicated to my parents and my fiancée, Breanna.
Thank you for always supporting me.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND PERSISTENCE OPTIMIZATION

1.1 Introduction

Topological Data Analysis (TDA) is a field that is focused on quantifying shape or global structure information from data. One of it most common tools, persistent homology has been used across many domains such as damping parameter estimation [1], bifurcation detection [2] and chatter detection in machining [3]. These are only a few of the many successful applications of persistent homology. Due to the inherent connection between dynamical systems and topology, persistence is an ideal tool for studying dynamical systems and developing automatic methods for analyzing time series signals. While the success of persistent homology has been wide reaching, it has been limited by the lack of a calculus on the space of persistence diagrams. Recently a framework for differential calculus has been introduced and studied in the context of optimization on the space of persistence diagrams [4–6] that enables a gradient descent optimization of persistence based functions. Overall, my work is organized into the five chapters shown in Fig. 1.1 where the overarching tools from persistent homology represent a common theme between most of this work. The projects are color coded according to where they fit into my research plan. Chapter 1 contains the necessary optimization background and presents a novel data assimilation algorithm harnessing the power of persistence optimization for optimizing data driven model forecasts using TDA. To date, the vast majority of work in topological data analysis has been in the absence of calculus and leveraging the recent advancements made in the differentiability of persistence diagrams unlocks an entirely new class of problems that can be solved with TDA. I will start by introducing the relevant optimization background theory in Section 1.2. I will then introduce topological data analysis and persistent homology with persistence optimization and numerical examples in Section 1.3. Then, in Chapter 2, I introduce a new data assimilation algorithm using persistence optimization to optimally update data driven forecasting models for dynamical systems in Section 2.1. An application to high-fidelity Hall Effect Thruster (HET) simulation data from the Air Force Research Laboratory (AFRL) is then presented in Section 2.2. For the second application

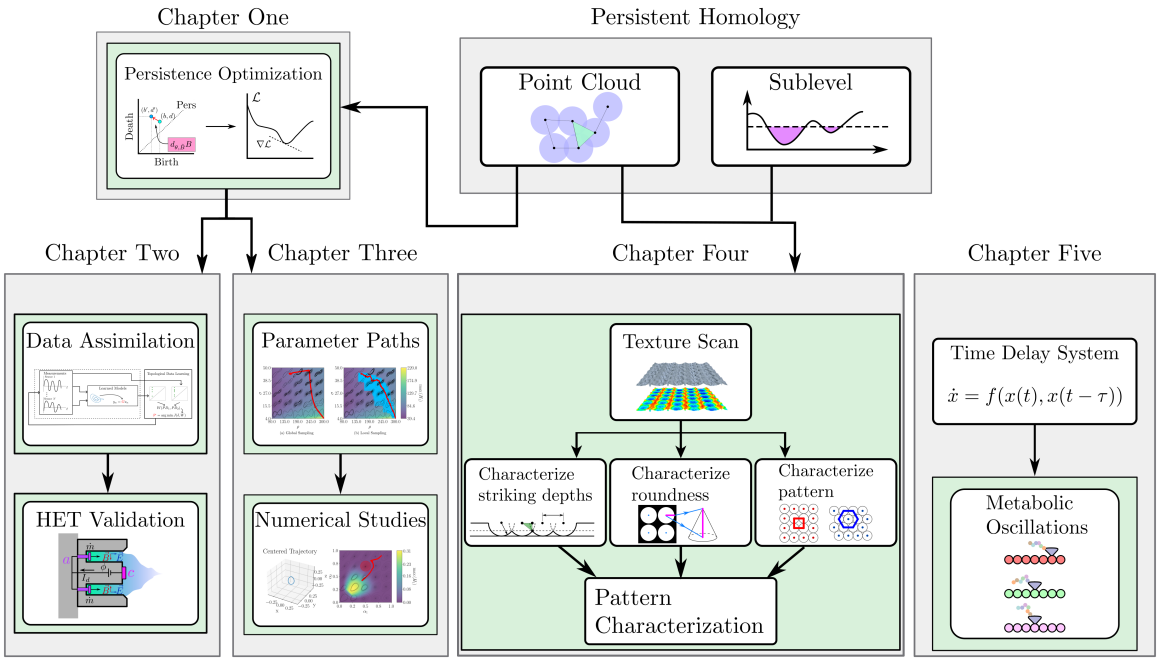


Figure 1.1 Overview of the work presented in this thesis.

of persistence optimization, I introduce a framework for optimal parameter space navigation of dynamical systems in Chapter 3. The behavior of a dynamical system is heavily governed by the topological structure of its state space response, so it is paramount that the connection between these domains is well understood. In general, a dynamical system may contain many parameters that control the qualitative behavior of the system and performing this analysis can be tedious and require expert level decisions for reducing the dimensionality of the problem. Leveraging topological features of the system trajectories in a data driven approach can allow for intuitively specifying desired performance characteristics of the system without the need for a model. I introduce a method for implementing dynamical system parameter spaces into the persistence differentiation framework to allow for the full inverse problem to be solved. This method is demonstrated on a simple dynamical system and the results are shown in Section 3.1. In Chapter 4, I present my work on performing texture analysis using TDA. Data was analyzed from a novel manufacturing process called Piezo Vibration Striking Treatment (PVST) where plastic deformation is induced on the surface of a part to create a texture of indentations that can be controlled using various system

parameters. The system parameters are tightly linked to characteristics of the resulting texture and mechanical properties. Prior to this work the textures were analyzed manually by inspecting the texture scans. I aimed to quantify three texture features using TDA: striking depths, roundness and pattern shape. Scores were developed to quantify these features from the image data using sublevel persistent homology. The first paper in Section 4.1 introduces the methods for quantifying the depth and roundness features and this work is published in [7]. For the second paper in Section 4.2, the method for quantifying pattern shape of a texture is presented and this work is published in [8]. Lastly, in Chapter 5 my work on a nonlinear delay model for metabolic oscillations is shown. Experimentally, it was observed that in a state of limited resource, protein production rates of yeast cell colonies oscillate in approximately 40 minute intervals. I set out to model these oscillations using a time delay framework. Due to the immense complexity of time delay systems, I explored three different numerical methods for searching for parameters that resulted in limit cycle oscillations along with verifying that the solution was periodic. I also extend the model to include three coupled proteins to observe how the protein production rates behaved when using a shared resource pool. The work introducing and analyzing this model is published in [9].

1.2 Optimization Background

Many different methods have been developed for solving optimization problems. This section reviews the basic concepts for classical optimization methods to review the background needed for performing optimization using topological data analysis and persistent homology. Scalar optimization methods are discussed in Section 1.2.1 and vector optimization methods are discussed in Section 1.2.2.

1.2.1 Scalar Optimization Methods

Before the differentiability of persistence based functions can be defined, it is crucial to understand classical optimization approaches to provide a point of comparison for my methods. Using classical scalar optimization as a starting point, given an objective function $f : \mathbb{R}^n \to \mathbb{R}$, I am interested in methods that can be used to solve the following constrained optimization problem: $\min_{\vec{x}\in\Omega} f(\vec{x})$, where $\Omega \subseteq \mathbb{R}^n$ is the feasible region, $\vec{x} \in \mathbb{R}^n$ is the input vector, and n is the dimen-

sion of the input space. If the objective function is chosen for a specific problem using features of the data, an optimal solution can be obtained to satisfy the desired requirements. Methods for solving this problem in general are highly dependent on the form and properties of the objective and constraint functions. Two main categories exist for optimization methods: derivative based, and derivative free. Derivative based methods are summarized in Section. 1.2.1.1, and derivative free methods are explored in Section. 1.2.1.2.

1.2.1.1 Derivative Based Methods

The simplest form of derivative based optimization is when the objective function is scalar-valued and is differentiable on its entire domain. In this case, the gradient can be used to reach a local minimizer of the function where the critical points are computed by analytically solving $\nabla f(\vec{x}) = 0$. The solutions to this equation correspond to local minima or *critical points* of the objective function f. Specifically, a local minima is defined as follows,

Definition 1. *Let $D(f)$ be the domain of f. Given a point $\vec{x}_0 \in D(f)$, we say that x_0 is a* local minimizer *of f if and only if there exists $\varepsilon > 0$ such that $f(\vec{x}_0) \leq f(\vec{x})$ for all $\|\vec{x} - \vec{x}_0\| < \varepsilon$.*

In other words, small perturbations of $\vec{x}$ near $\vec{x}_0$ lead to larger objective function output values making $\vec{x}_0$ a local minimizer.

Definition 2. *We say $\vec{x}_0 \in D(f)$ is a* global minimizer *if and only if $f(\vec{x}_0) \leq f(\vec{x})$ for all $\vec{x} \in D(f(\vec{x}))$.*

If Ω is comprised of a set of convex inequalities and affine equality relationships, and the domain of the objective function is a convex set, the problem is said to be a convex optimization problem. In this case, the local minimizer is a global minimizer of the problem.

Definition 3. *A set Ω is* convex *if and only if for all $x, y \in \Omega$, $tx + (1-t)y \in \Omega$ $\forall t \in [0,1]$.*

Definition 4. *A function f is* convex *if and only if $D(f)$ is a convex set, and for all $\vec{x}_1$, $\vec{x}_2 \in D(f)$, $f(t\vec{x}_1 + (1-t)\vec{x}_2) \leq tf(\vec{x}_1) + (1-t)\vec{x}_2$ $\forall t \in [0,1]$.*

Definition 5. *A function g is* affine *if and only if for some $\vec{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$, $g(\vec{x}) = \vec{a} \cdot \vec{x} + b$.*

In the case where the critical points of the objective function cannot be determined analytically, the **gradient descent** algorithm is used. This algorithm takes steps in the input space of the function in the direction opposite of the gradient or in other words stepping in the direction of steepest descent [10]. A learning rate η is used as a hyperparameter to determine the size of the steps in the input space and steps are taken in the input space until a local minimizer is reached if it exists. There are three main variants of this process, batch, stochastic, and mini-batch gradient descent [10]. **Batch gradient descent** is the original gradient descent method and the updated estimate of the local minimizer can be computed as,

$$\vec{x}_{n+1} = \vec{x}_n - \eta \nabla f(\vec{x}_n),$$

where $n \in \mathbb{Z}$ is the current index up to N the number of steps. This algorithm is guaranteed to converge to a local minimizer for non-convex objective functions, and global minimizers for convex functions [10]. In practice, the gradient can be expensive to evaluate over many directions to find the direction that yields the largest descent. **Stochastic gradient descent** mitigates this problem by computing the updated point using a single, randomly chosen direction to evaluate the gradient [11]. The addition of a stochastic component in the algorithm does not change the convergence properties if the learning rate is slowly decreased, and it also may converge to a more desirable local minimum due to the overshooting [10]. The third method is **mini-batch gradient descent**. Mini-batch combines the batch and stochastic gradient descent algorithms by performing a series of small batch updates by estimating the gradient using a subset of randomly chosen data points allowing for faster computations while decreasing the variance in the updates making it a more stable algorithm [10]. Stochastic methods are much more common when using gradient descent techniques because the randomness in the process allows for potentially jumping to a better local minima if the objective function has multiple minima, whereas batch gradient descent is more likely to approach the minimizer closest to the starting point [11].

1.2.1.2 Derivative Free Methods

Gradient computation is difficult if the objective function contains noise and impossible in the case where the function is not differentiable. In the general case, optimization methods that

do not rely on derivatives are necessary. Derivative free optimization (DFO) is centered around using alternative methods to solve optimization functions that do not require the gradient of the objective function. There are two main approaches to optimization without differentiation: direct-search, and model-based [12]. Direct-search methods leverage algorithmic function evaluations and comparisons to locate potential solutions whereas model-based methods employ surrogate models to approximate the objective function and analytically solve the surrogate optimization problem [12].

Direct-Search Methods: Within the area of direct-search methods, three types of algorithms are commonly used: line-search, discrete grids, and simplex methods [13]. The **line-search** method takes a given starting point $\vec{x}_k$ and direction $\vec{d}_k$. The objective function is then evaluated along the line $\phi(\alpha) = f(\vec{x}_k + \alpha\vec{d}_k)$ until the condition $f(\vec{x}_k + \alpha\vec{d}_k) \leq f(\vec{x}_k)$ is satisfied. Once the condition is met, the new point is $\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{d}_k$. This process is continued until the minimizing α is below a specified tolerance [13]. It is important that the chosen directions can eventually span $\mathbb{R}^n$ so that all potential points can be reached by the algorithm. For this reason, the unit coordinate directions are typically chosen in successive order and are cycled through for each step [13]. This method does not have any guarantees on reaching a limit point and depending on the step sizes can eventually begin to cycle through values [13]. To avoid this issue, an additional condition is imposed that if $\|x_{k+1} - x_k\|$ is bounded away from zero, then $f(\vec{x}_{k+1}) - f(\vec{x}_k)$ is also bounded away from zero [13]. This condition prevents movement in the input space causing no change in the output of the function resulting in cyclic behavior. **Discrete grid** methods bound the input variables within an n-dimensional rectangular grid where $a_i \leq x_i \leq b_i$ for $i = 1...n$. The iterative process searches for $\vec{x}_{k+1}$ on the grid such that $f(\vec{x}_{k+1}) \leq f(\vec{x}_k)$ [13]. The evaluation points are typically chosen by sequentially stepping along the coordinate directions to determine which direction to move [13]. **Simplex** methods are the third direct-search approach where a set of points are generated in the search space to create a simplicial complex. The algorithm from [14] takes the vertex with the largest function value in the complex and either moves remaining vertices toward the current largest one or reflects the simplicial complex through the hyperplane spanned by the

remaining vertices repeating the process until an optimal value is reached. There are other methods that utilize simplices to solve optimization problems such as the simplicial homology global optimization (SHGO) algorithm in [15] where a directed simplicial complex is formed based on directing the edges according to comparing the magnitude of the function values at the vertices. The set of minimizers in this case is then defined by all vertices that have every edge pointing toward the vertex itself. This algorithm guarantees finding a global optimizer in finite time if the simplicial sampling method is used to locate the vertices of the complex. It also does not require that the function be smooth or continuous to find the optimizer. Notably, this method does not have any smoothness requirements if the simplicial sampling method is used whereas other sampling methods may require the function to be Lipschitz smooth [15].

Model-Based Methods: The second class of DFO methods is a model-based approach. Rather than using the full objective function, a surrogate model is used to approximate the cost function for solving the optimization problem [12]. Many different model-based approaches exist for DFO such as simply using interpolation or regression on output data points of the original function [16]. Many of the commonly used model-based DFO algorithms were developed by Powell [16]. These algorithms are all trust region methods meaning that a series of smaller optimization problems are solved near the current point using radius r_k with a given trust region defined by $\|x-x_k\| \leq r_k$ [16]. Within the trust region, the objective function is then approximated as $f_k(x)$ with a linear or quadratic surrogate model and the surrogate model is then minimized within the intersection of the trust region and feasible region for the overall problem. The model $f_k(x)$ is determined from sample points within a trust region. Using a linear model about a base point $y^b \in \mathbb{R}^n$, $f_k(x)$ is computed using a Taylor expansion as $f_k(x) = f(y^b) + (x-y^b)^T \nabla f_k(y^b)$ and fitting this model requires $n+1$ sample points in $\mathbb{R}^n$. A similar model is obtained for quadratic approximations by adding the term $\frac{1}{2}(x-y^b)^\intercal \nabla^2 f_k(y^b)(x-y^b)$ and sampling $\frac{1}{2}(n+1)(n+2)$ points within the trust region [16]. We see that the number of required sample points increases as $\mathcal{O}(n^2)$ which can become impractical for computationally expensive function evaluations. To avoid this issue, undetermined quadratic interpolation is used where a regularization problem is solved with respect to the previous iteration

$k-1$ to allow for a smaller interpolation set [16]. Powell described five model-based optimization schemes: Constrained Optimization BY Linear Approximation (COBYLA), Unconstrained Optimization BY Quadratic Approximation (UOBYQA), LINearly Constrained Optimization Algorithm (LINCOA), Bounded Optimization BY Quadratic Approximation (BOBYQA), and NEW Unconstrained Optimization Algorithm (NEWUOA) [16]. See [16–22] for more details.

1.2.2 Vector and Multi-objective Optimization

All of the methods discussed in the previous section require a scalar-valued objective function and the resulting solution depends strongly on how well that function is defined. In general, the field of vector optimization deals with solving problems such as $\min_{\vec{x}\in\Omega} f(\vec{x})$ where in general $f:\mathbb{R}^n \to \mathbb{R}^p$ and Ω is defined using inequality and equality constraints as with the scalar methods [23]. In an ideal case, one would solve decoupled optimization problems to individually minimize each $f_i(\vec{x})$, however, this is not always possible with a general feasible set.

Pareto Optimization: The concept of Pareto optimal points was introduced for this case where the solution $\vec{x}_0$ is considered optimal if $\forall \vec{x} \in \Omega,\ \nexists\ f_i(\vec{x}) \leq f_i(\vec{x}_0)\ i = 1,...p$ and at least one $f_j(\vec{x}) < f_j(\vec{x}_0)$ [23]. Multiple Pareto optimal solutions may exist for a given multi-objective optimization problem and all of the solutions have been shown to be on the boundary of the feasible set [24,25]. The concept of weak Pareto optimal solutions is also commonly used when true Pareto solutions are difficult to find. A solution $\vec{x}_0$ is weakly Pareto optimal if and only if $\nexists \vec{x} \in \Omega$ such that $f_i(\vec{x}) < f_i(\vec{x}_0)$ [24]. Many methods exist for solving for Pareto and weakly Pareto optimal points that are summarized in [24], but some of the most common methods involve the process of *scalarization* where the vector valued objective function is reduced to a scalar using a combination of all of the objective function components. In particular, the weighted sum method is highlighted in [24] where the objective function is converted to a scalar valued function as $U = \sum_{i=1}^{p} w_i f_i(\vec{x})$ and it has been shown that if $w_i > 0\ \forall i$, that the solution is Pareto optimal [26]. The weights are typically constrained to sum to one and if any of the weights are equal to zero the solution may be weakly Pareto optimal [24]. However, choosing the weights is a nontrivial task that can have a significant impact on the final solution. Another approach that does not require any input from the user is to

convert the objective function to a scalar by taking its largest component [24].

1.3 Topological Data Analysis and Persistence Optimization

This section reviews the needed concepts from topological data analysis: persistent homology (Section 1.3.1) and persistence optimization (Section 1.3.2). Topological data analysis (TDA) quantifies structure in data. This section reviews the basics of persistent homology for point clouds in $\mathbb{R}^n$. More specifics can be found in [27–34]. While the theory is presented in terms of a general point cloud, it is helpful to think of the points being states of an n-dimensional dynamical system.

1.3.1 Persistent Homology

Homology groups can be used to quantify structure or shape in different dimensions on a simplicial complex K. This is done using homology, $H_p(K)$, which is a vector space computed from the complex, where p is the dimension of structures measured. For example, in dimension 0, the rank of the 0 dimensional homology group $H_0(K)$ is the number of connected components. The rank of the 1-dimensional homology group $H_1(K)$ is the number of loops or holes, while the rank of $H_2(K)$ is the number of voids, and so on. For example, consider Fig. 1.2 (e) where we see that the simplicial complex contains two holes meaning that the rank of the 1D homology at this particular value of the connectivity parameter is 2.

I am interested in studying the structure of a changing simplicial complex (a generalization of a graph) by measuring its changing homology. This process is called persistent homology. In general, I assume we have a real valued function on the simplices of K whose values are used to generate the simplicial complex. For this chapter I focus on a specific simplicial complex called the Vietoris-Rips or simply Rips complex (VR) where the function on the point cloud $\{x_1, \cdots, x_N\} \subseteq \mathbb{R}^d$ becomes the Euclidean distance between points. The basic idea is that the Rips complex with parameter ε is a higher dimensional analogue of the proximity graph, where two vertices are connected with an edge if the distance between the relevant points in the point cloud are at most distance 2ε. This process forms a nested sequence or filtration of simplicial complexes $K_0 \subseteq K_1 \subseteq ... \subseteq K_n$ which induces a sequence of inclusion maps on the homology $H_p(K_1) \to H_p(K_2) \to \cdots \to H_p(K_n)$. An example can be seen in the series of simplicial complexes

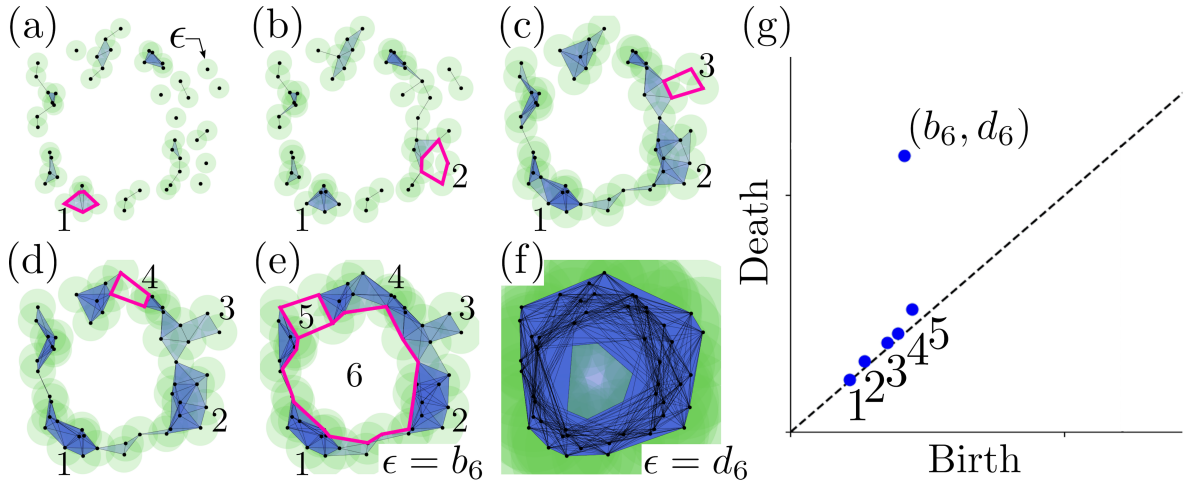


Figure 1.2 Persistence for point clouds. Each snapshot in (a)–(e) shows the rips complex for increasing values of a disc of radius ε. One prominent loop is formed (or born) at $\varepsilon = b_6$ in (e), and fills in (or dies) in (f) when $\varepsilon = d_6$. The other 5 loops are small and a result of noise so they are born and die at nearly equivalent values of ε. These loops are represented in the 1D persistence diagram (g) as (birth,death) pairs. Non-prominent loops form and die quickly as shown by the points near the diagonal.

in Fig. 1.2.

The appearance and disappearance of holes in the filtration is encoded in this sequence. This information is then represented in a persistence diagram, where the large loop appears at $\varepsilon = b_6$ and fills in at $\varepsilon = d_6$. This information is represented as a point in $\mathbb{R}^2$ at (b_6, d_6) in Fig. 1.2 (f). The other 5 loops in this particular point cloud are all born and die almost immediately so those persistence pairs show up near the diagonal. The collection of the points in the persistence diagram give a summary of the topological features that persist over the defined filtration. Points far from the diagonal represent structures that persist for a long time, and thus are often considered to be prominent features. Conversely, points close to the diagonal are often attributed to noise in the data and it is clear that loops 1–5 are due to the noise in Fig. 1.2.

1.3.2 Persistence Optimization

An emerging subfield of topological data analysis deals with optimization of persistence based functions by exploiting the differentiability of persistence diagrams. Persistence diagrams are commonly represented by many different scalar features used for machine learning such as the total persistence [4], $\text{totPers} = \sum_{i=1}^{p} |d_i - b_i|$, which gives a measure of how far the persistence

pairs are from the diagonal. In other words, this feature gives the sum of the persistence lifetimes $\ell_i = d_i - b_i$. These scalar representations are referred to as *functions of persistence* [4]. Other examples of functions of persistence include maximum persistence, $\text{maxPers} = \max_i |d_i - b_i|$, and persistent entropy $E = -\sum_i p_i \log_2(p_i)$ where $p_i = \frac{\ell_i}{\sum_i \ell_i}$ [35] which gives a measure of order of the persistence diagram. Features such as the Wasserstein or bottleneck distance can also be used for measuring dissimilarities between two PDs [4, 36]. In [4], it is specified that in order to have differentiability of the persistence map, the function of persistence must be locally Lipschitz and definable in an o-minimal structure or in other words definable using finitely many unions of points and intervals. An example of a set that fails this criteria is the Cantor set because it requires infinitely many operations to determine if a point is in the set.

Generally, a function of persistence is evaluated through the map composition,

$$\mathcal{C} : \mathcal{M} \xrightarrow{B} \text{PD} \xrightarrow{V} \mathbb{R}, \tag{1.1}$$

where the input space $\mathcal{M}$ can be a point cloud or image that is mapped to a persistence diagram using the filtration B [5]. An example of this mapping is shown in Fig. 1.3 where the square point cloud is mapped to a persistence diagram using the map B with VR filter function and the persistence diagram *PD* is mapped to the total persistence feature using the map V. The composition of these maps ($V \circ B$) allows for directly mapping the point cloud to persistence features. Reference [4] outlines the optimization of persistence-based functions especially via stochastic subgradient descent algorithms for simplicial and cubical complexes with explicit conditions that ensure convergence. A function of persistence is defined as a map from the space of persistence diagrams associated to a filtration of a simplicial complex to the real numbers such that it is invariant to permutations of the points of the persistence diagram.

The PD is represented as a real number by way of the chosen function of persistence V. $\mathcal{C}$ has enabled differentiability and gradient descent optimization of its members using the chain rule on $V \circ B$ to obtain desired characteristics of $\mathcal{M}$ [4–6]. B is differentiated by considering a local perturbation or lift of the input space $\mathcal{M}$, $\tilde{B}$. The space of possible perturbations is then mapped onto the PD, and for a particular perturbation of $\mathcal{M}$, the directions of change of the persistence

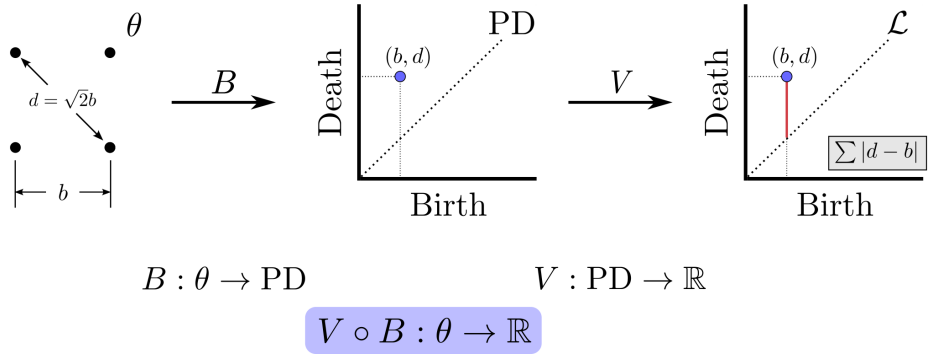


Figure 1.3 Mapping a point cloud θ to a real values persistence feature using the map composition $V \circ B$.

pairs form the derivative of B with respect to $\tilde{B}$ [5]. This process is pictorially represented using a simple point cloud in $\mathbb{R}^2$ consisting of a single loop in Fig. 1.4. The top row from left to right shows the original point cloud along with the simplicial complex where the loop is born σ, and where it dies σ'. The corresponding attaching edges where these events occur are labeled as b and d with vertices $w(\cdot)$ and $v(\cdot)$. The map B is used to map the point cloud to the persistence diagram. The bottom row of Fig. 1.4 demonstrates the same process as the top row but on a perturbed point cloud θ' where $p_2 \to p_2'$ along $\hat{u}$. The map $\tilde{B}$ represents the persistence map for the perturbed point cloud and the resulting change in the persistence pair forms the derivative of the persistence map B with respect to θ and $\tilde{B}$. Mathematically, this is represented as $d_{\theta,\tilde{B}}B$ [5].

This process is illustrated more generally and for 0D persistence in the example shown in Fig. 1.5. In this diagram, the space of infinitesimal perturbations of the point cloud P is shown in blue and this higher dimensional space is mapped onto a persistence diagram where the quotient of the space collapses to the original persistence pair. For the particular perturbation shown, we see that the edge length is increasing, so the derivative of B using the VR filtration with respect to the perturbation P', the corresponding persistence map $\tilde{B}$, is a vector in the vertical direction. For higher dimensional simplices or PDs such as in Fig. 1.4, the process is the same, however, we consider the rate of change of the attaching edge of the simplex or the edge whose inclusion results in the birth of the simplex [4, 5]. Attaching edges are the output of the corresponding filter

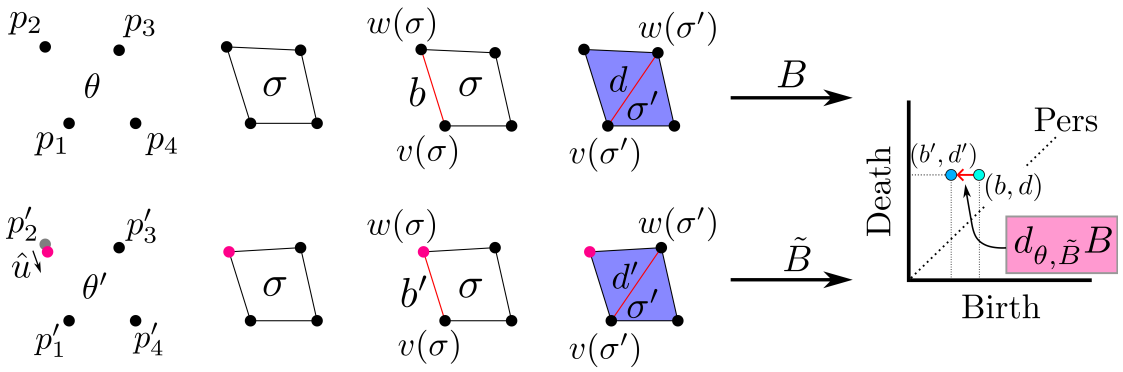


Figure 1.4 Persistence diagram differentiation process. The top row shows the process of tracking the birth and death of the loop from the original point cloud along with using the map B to obtain its persistence diagram. The bottom row performs the same process on a perturbed point cloud and demonstrates how the change in the persistence pair forms the derivative $d_{\theta,\tilde{B}}B$.

function chosen. For example, if the VR filtration is used, the filter function for a simplex σ is defined to be $F(P)(\sigma) = \max_{i,j\in\sigma} ||p_i - p_j||_2$ or the maximal distance between any two vertices in the simplex [5] where $||\cdot||_2$ is the l_2 norm. Before the connectivity parameter reaches $F(P)(\sigma)$, σ remains unborn in the filtration. In this case, the map B corresponds to the composition of the persistence map Dgm_p and the filter function F [5]. Conditions of differentiability must be considered for the input space being studied. If the input is a point cloud it must be in general position [5, 6] (i.e., no two points in the cloud coincide or are the same distance apart as any other pair). Nonetheless, if the general position condition fails then the derivative likely still exists for the specified perturbation. The issue is also mitigated numerically by CPU floating point precision and the constraints are highly unlikely to be violated with real data [6]. If either condition is violated, small artificial noise can also be introduced to guarantee the points are in general position and a unique perturbation exists.

For point cloud input data, the derivative of a persistence diagram is computed by labeling the vertices of attaching edges for the birth σ and death σ' of a simplex as $v(\sigma)$, $w(\sigma)$ and $v(\sigma')$, $w(\sigma')$ respectively for each attaching edge that results in the birth or death of a homology class. An arbitrary perturbation P' of the point cloud P is then considered. The attaching edge vertices are tracked in the process allowing for each persistence pair to measure the direction of change and

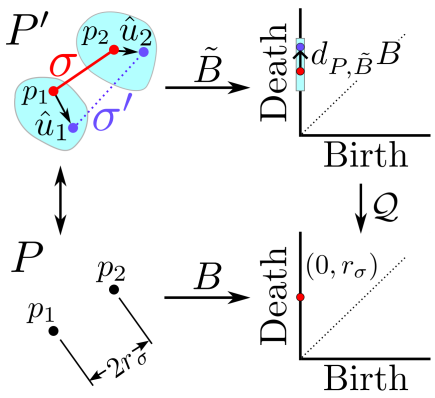


Figure 1.5 Persistence differentiation for point clouds. The point cloud P is perturbed to P' and the 0D persistence diagram is differentiated with respect to this perturbation.

construct the derivative of the persistence diagram with respect to that perturbation. The derivative is represented by a list of vectors (one for each persistence pair) that describe the variation in the persistence pairs with respect to a given perturbation P'. A unit direction vector $\hat{u}$ is used to store the perturbation directions for each point. The derivative is then computed via an inner product of the vector $P_{i,j} = \frac{p_i - p_j}{||p_i - p_j||_2}$ which describes the direction of change in length of the attaching edge (i, j) and the perturbation vector $\hat{u}$. Mathematically using the VR filtration, the derivative takes the form,

$$d_{P,\tilde{B}}B(\hat{u}) = \left[\left(P^T_{v(\sigma),w(\sigma)}\hat{u},\ P^T_{v(\sigma'),w(\sigma')}\hat{u} \right)_{i=1}^{m} \right], \tag{1.2}$$

where $d_{P,\tilde{B}}B$ is the derivative of the persistence map B with respect to the perturbation persistence map $\tilde{B}$ evaluated at the perturbation $\hat{u}$ [5]. Note that the form Eq. (1.2) has been represented for m finite persistence pairs generalizations are presented in [5] from parameterization by Rips filtration to present a formal framework for differentiation of persistence diagrams using maps between smooth manifolds $\mathcal{M}$ and $\mathcal{N}$ through space of persistence diagrams with a general filter function in [5]. This framework also includes generalizations to infinite persistence pairs, however, for this work I am mainly interested in finite persistence pairs using the VR filter function.

One of the primary applications of this optimization comes from [4] where a TensorFlow pipeline was developed using the Gudhi TDA library in Python to optimize the positions of points in a point cloud with gradient descent according to a predefined loss function. The loss function

in [4] was defined to maximize the total persistence or in other words expand the size of the loops in the 1D persistence diagram. A term was also added to the cost function to regularize by restricting the points to a square region of space. More loss functions can also be defined in terms of persistent entropy to promote fewer loops in the point cloud and using the Wasserstein distance to achieve a desired persistence diagram. The work in [6] outlines processes for carrying out optimization using persistence based functions in the specific case of Vietoris-Rips complexes defined on point clouds. Particularly, these methods allow for the user to supply a start and end persistence diagram along with the starting point cloud. Gradient descent is then used to optimally transform the original point cloud into a new point cloud that has the desired homology.

1.3.3 Persistence Optimization Examples

Functions of persistence can be used to engineer loss functions to achieve desired topological properties of a point cloud. This section includes examples of the persistence optimization process using the TensorFlow and Gudhi pipeline from [4].

Loop Expansion: The first example aimed to increase the size of loops in the point cloud by defining the cost function, $\mathcal{L} = -\sum_i |d_i - b_i| + \sum_i \max(|p_i| - 1, 0)$. The first term in $\mathcal{L}$ is the total persistence feature where (b_i, d_i) is the i-th persistence pair. The second term is a regularization to penalize points that are outside of a 2×2 region of space. Figure 1.6 shows the starting point cloud and 1D persistence diagram where points are sampled from a small circle. Performing the optimization using $\mathcal{L}$ over 3000 epochs yielded the results in Fig. 1.7. The point cloud expanded to fill the region until the regularization term prevented points from leaving the 2×2 region. In the persistence diagram, the 1D persistence pair moved vertically to have a final death time of approximately 2 and the loss function plot indicates that a minimum has been reached.

Combining Persistence Functions: Loss functions can be defined to simultaneously promote multiple topological features in the optimization process. For the first example the loss function was defined as in the first example but a term was added $\max \sum_i (\ell_i - \ell_{max})$ where ℓ_i is the i-th persistence lifetime. In other words, a lifetime larger than ℓ_{max} penalizes the cost function and for this specific example ℓ_{max} was set to be 1. The initial point cloud for this method is shown in

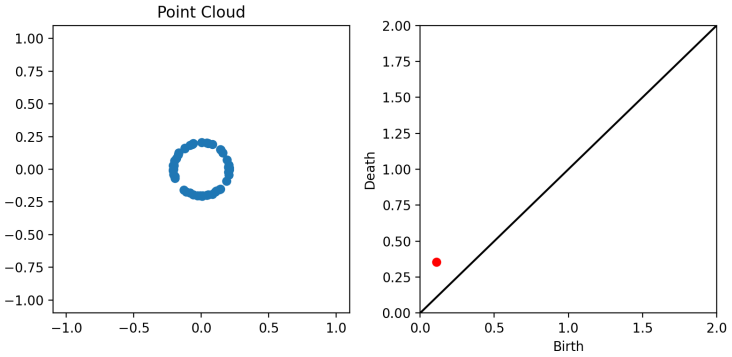


Figure 1.6 Persistence optimization expanding loop example. Initial circular point cloud and 1D persistence diagram.

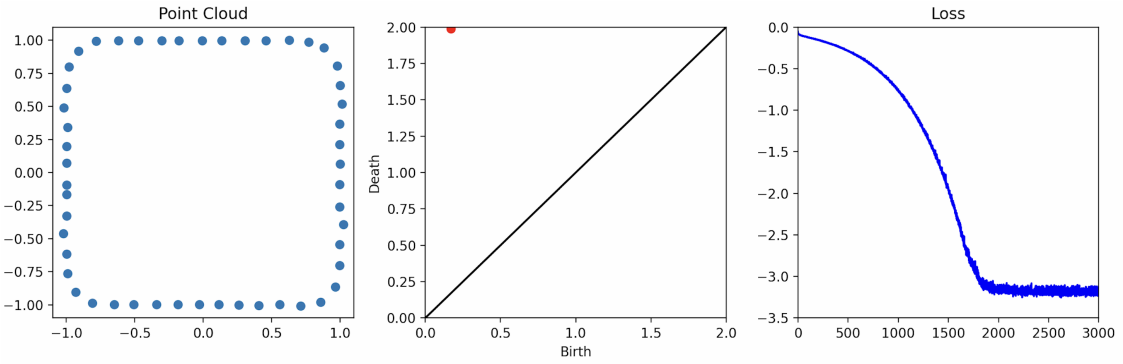


Figure 1.7 Persistence optimization expanding loop example. Resultant point cloud and 1D persistence diagram after 3000 gradient descent steps using the corresponding cost function.

Fig. 1.8 where points were randomly sampled within an annular region. The corresponding 1D persistence diagram is shown on the right. Performing the optimization in this case resulted in the point cloud and persistence diagram in Fig. 1.9 where the loops expanded in the point cloud while ensuring that all persistence pairs remained below a lifetime of 1. The loss function was then augmented to include a persistent entropy term $E(D) = -\sum_i p_i \log_2(p_i)$ to lead to a simpler solution to the problem. The initial point cloud was similar to the case in Fig. 1.8 and after performing the optimization in this case, the resulting point cloud is shown in Fig. 1.10. It is clear that the entropy term results in a point cloud with fewer loops and allows the loop sizes to get closer to the lifetime restriction due to there being more space to grow a loop. **Target Persistence Diagrams:** For the last example I show how the Wasserstein distance can be used to reach a point cloud with a target persistence diagram by minimizing the distance between two persistence diagrams. I started

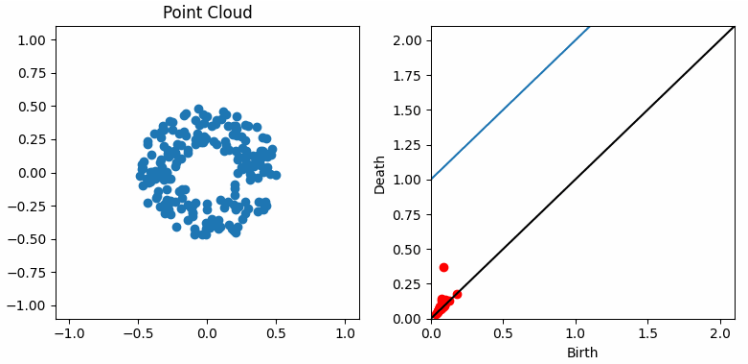


Figure 1.8 Initial point cloud for the second persistence optimization example. The cost function was defined in the same way as the first example with the addition of a term to penalize lifetimes larger than 1 or above the blue line.

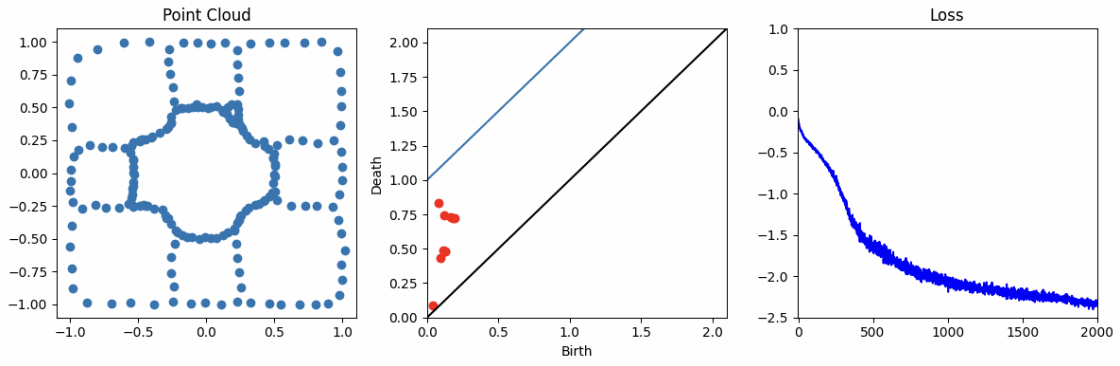


Figure 1.9 Final point cloud for the second persistence optimization example. The cost function was defined in the same way as the first example with the addition of a term to penalize lifetimes larger than 1 or above the blue line.

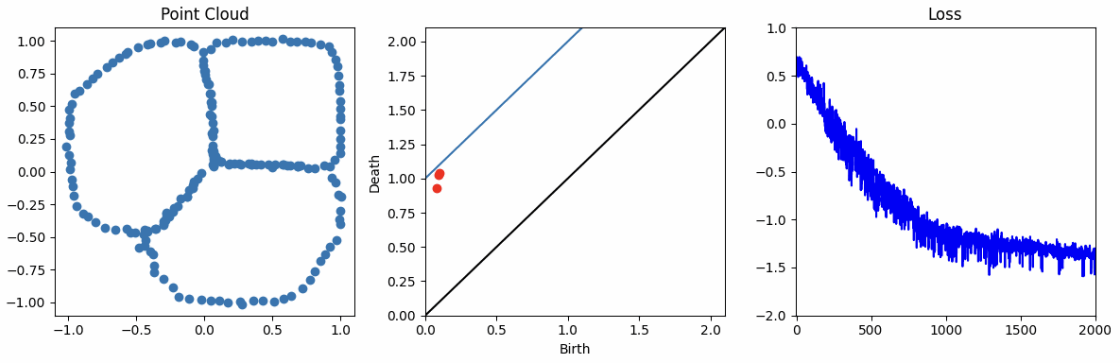


Figure 1.10 Resulting point cloud for the third persistence optimization example. The cost function was defined to maximize loop size while restricting persistence pairs to have a lifetime less than 1. A persistent entropy term was added in this case to give a point cloud with fewer loops.

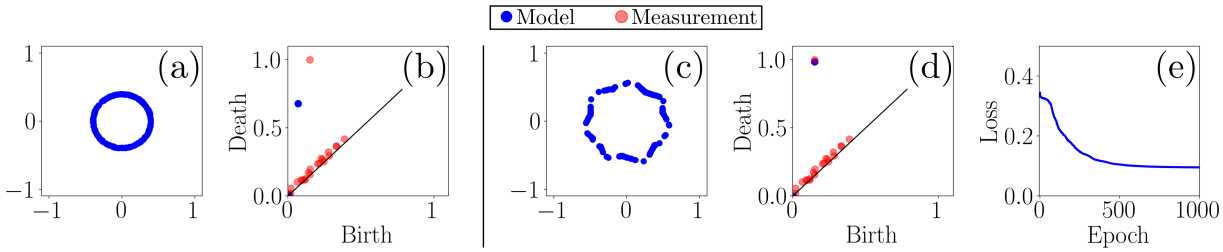


Figure 1.11 Example minimizing the Wasserstein distance persistence function to reach a point cloud with a target persistence diagram. (a) shows the original point cloud and (b) shows the original persistence diagram and target persistence diagram. (c) shows the optimized point cloud and (d) shows the optimized persistence diagram with the loss plot in (e).

with the circular point cloud shown in Fig. 1.11 (a) and the corresponding persistence diagram is shown in Fig. 1.11 (b) where the blue persistence pair indicates the starting 1D persistence diagram and the red points are a constructed target persistence diagram. Using a cost function of the form $J = W(PD_c, PD_t)$ where W is the Wasserstein distance, PD_c is the current persistence diagram (or forecast model persistence diagram) and PD_t is the target diagram (or measurement persistence diagram). J has a clear minimizer at 0 where the persistence diagrams are identical. Performing the optimization process yielded the point cloud in Fig. 1.11 (c) where the point cloud expanded to reach the target persistence diagram in Fig. 1.11 (d). It is clear in the loss function plot that the result converged in Fig. 1.11 (e) and the target persistence diagram contained artificial features due to noise that did not influence the optimization process and only the prominent topological features were learned.

CHAPTER 2

DATA ASSIMILATION USING PERSISTENCE OPTIMIZATION

2.1 Topological Approach for Data Assimilation

Many dynamical systems are difficult or impossible to model using high fidelity physics based models. Consequently, researchers are relying more on data driven models to make predictions and forecasts. Based on limited training data, machine learning models often deviate from the true system states over time and need to be continually updated as new measurements are taken using data assimilation. Classical data assimilation algorithms typically require knowledge of the measurement noise statistics which may be unknown. In this paper, I introduce a new data assimilation algorithm with a foundation in topological data analysis. By leveraging the differentiability of functions of persistence, gradient descent optimization is used to minimize topological differences between measurements and forecast predictions by tuning data driven model coefficients without using noise information from the measurements. I describe the method and focus on its capabilities performance using the chaotic Lorenz system as an example.

2.1.1 Introduction

Physics-based dynamical system modeling is a very powerful and interpretable tool that can be used to make predictions and can provide relatively low-cost insight into system design and rapid prototyping. However, a model is only as good as the physics being used to define it and if the true system exhibits multi-scale behavior that requires extreme fidelity for simulation, the computational complexity outweighs the benefits of modeling the system. Many systems of interest to researchers are inherently multi-scale and high dimensional making them difficult or impossible to accurately predict using high fidelity physics based models. As a result, researchers are relying more heavily on data driven modeling techniques using machine learning to gain insight and make predictions for systems without the overhead computational cost [37]. Many different data driven modeling techniques have been developed such as the AutoRegressive (AR), Moving Average (MA), and AutoRegressive Integrated Moving Average (ARIMA) models which assume a linear model form and learn coefficients from past training data [38, 39]. A more modern approach

to data driven modeling is rooted in deep learning and neural networks. While traditional Feed forward Neural Networks (FNN) are insufficient for time series forecasting due to the sequential ordering of points, Recurrent Neural Networks (RNN) are more suited for forecasting due to the dependence on previous states [40]. A form of RNN, the Echo State Network (ESN) has been used to construct data driven models from sparse measurements in [41]. A modified version of the RNN that is used in forecasting is called Long Short Term Memory (LSTM) model which uses basic building blocks of an RNN to recall states from previous steps and has been shown to give significant improvements in forecast horizon compared to ARIMA [42]. Another common forecasting approach is called Reservoir Computing (RC). RC works by mapping states into a high dimensional reservoir space and using linear regression to learn model coefficients as the features are mapped back into the original space [40]. RC based methods typically result in significant improvements to computation times while still accurately predicting future states of the system [40]. A special case of RC of interest for this paper is random feature map forecasting [43, 44]. This method is described in detail in Section 2.1.2.4. The quality of machine learning models is heavily dependent on the quality and quantity of training data. If the system changes states to a behavior that is drastically different form what was used to fit the coefficients, the corresponding model breaks down and the forecast will significantly deviate from the true system states. This issue is highlighted by any chaotic dynamical system where the forecast is accurate for a period of time after the training data and eventually deviates due to the finite model precision and training data.

To mitigate this problem, a concept called Data Assimilation (DA) is typically used. Data assimilation, or state estimation is a method for optimally combining observed data from multiple sources with model predictions to produce an improved prediction based on both [45–48]. It has been successfully used across many fields such as weather forecasting, oceanography, predicting the movement of pollution and forecasting wild fires [45, 46, 49, 50]. In [51] a sliding window approach is taken using the Proper Orthogonal Decomposition to estimate the prominent structures in the data at the current window combined with DA techniques to obtain optimal predictions using few dimensions, but this approach can be sensitive to noise. A common implementation of data

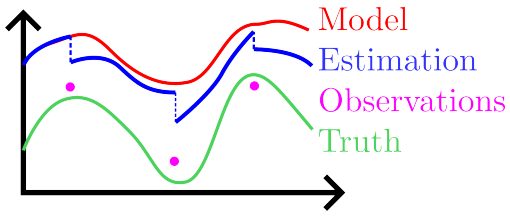


Figure 2.1 Data Assimilation Concept—Improving model results by considering observations to obtain an estimation closer to the ground truth.

assimilation is the Kalman filter applied to dynamical systems where noisy system states measurements are optimally estimated using information from the noise statistics such as the measurement covariance matrix and a system forecast is generated by combining information from all available measurements [44, 46]. This process is demonstrated in Fig. 2.1 where the observations move the model results closer to the ground truth to improve the predictions. In data assimilation, observed data streams and their uncertainties are used to update the model by solving for optimal weights with more importance given to sources with lower uncertainties. This is typically achieved by minimizing a cost function of the form [45],

$$J(\vec{x}) = (\vec{x}-\vec{x}_b)^T B^{-1}(\vec{x}-\vec{x}_b) + (\vec{y}-h(\vec{x}))^T R^{-1}(\vec{y}-h(\vec{x})), \tag{2.1}$$

where $\vec{x}_b \in \mathbb{R}^n$ is the vector of model prediction results, B is the covariance matrix for the model, $\vec{y} \in \mathbb{R}^m$ is the vector of observations and h is the operator that projects the input vector onto the space of observations, and R is the covariance matrix for the measurements [45]. Thus finding $\vec{x}_a = \arg\min J(\vec{x})$ yields an optimal combination of the model predictions and measurements [45] where x_a is referred to as the analysis solution. It was shown in [45] that $\vec{x}_a$ can be written as $\vec{x}_a = \vec{x}_b + K(\vec{y} - h(\vec{x}_b))$ for some weighting matrix K in terms of the covariance matrices and the operator h. Random feature map forecasting and data assimilation have been combined using the ensemble Kalman filter in the Random Feature Maps and Data Assimilation (RAFDA) algorithm [44,52] which optimizes the model sequentially on the training data using ensemble sampling of points that follow the noise distribution. Other approaches also utilize the ensemble Kalman filter such as in [53] where the ensemble Kalman filter is used with dynamic mode decomposition to reconstruct dynamical systems from data. These methods assume a Gaussian white noise distri-

bution with known statistics. Some filtering methods have been introduced for colored/correlated noise distributions in [54,55] but they still rely on known noise statistics. More recently, persistent homology from Topological Data Analysis (TDA) has been used to analyze complex data due to its compressive nature, vectorizability, and robustness to noise. Persistent homology was also shown to be robust to several noise distributions [56], which can provide important advantages for data assimilation. However, the possibility of incorporating TDA into data assimilation was unlocked only recently with the definition of differential calculus on the space of persistence diagrams [4–6]. I leverage these advances in auto-differentiation of persistence to define a Topological Approach for Data Assimilation (TADA). This approach integrates random feature map forecasting with topological cost functions and persistence optimization to update the system model with incoming measurements. One of the key steps in TADA is generalizing the cost function in Eq. (2.1) to use an arbitrary metric according to $J(\vec{x}) = d_b(\vec{x},\vec{x}_b) + d_y(\vec{y},h(\vec{x}))$, where d_b is the model discrepancy and d_y is the observation discrepancy. Using the Wasserstein distance as a metric to measure pairwise dissimilarity between the persistence diagrams of the measurement and model prediction enables the use of topology based cost functions [57]. I show that this approach successfully produces system forecasts that are resilient to white, pink and brown noise. This paper is organized such that the relevant theoretical background on time series forecasting is in Section 2.1.2, the TADA algorithm is then presented in Section 2.1.3 with results in Section 2.1.4.1.

2.1.2 Time Series Forecasting

This section includes the necessary background on time series forecasting. For this work, I focused on the method in Section 2.1.2.4, but for completeness I show other classical methods that could be used with this algorithm.

2.1.2.1 Autoregressive (AR) Forecasting

The autoregressive ($AR(p)$) model of order p works by assuming a model of the form,

$$X_n = \sum_{i=1}^{p} \varphi_i X_{n-i} + \varepsilon_n, \tag{2.2}$$

where X_n are the predicted system states at the next time step n, φ_i are the model coefficients learned from training data and ε_n is a noise term to prevent over fitting [58]. The order of the model is usually determined by the value of p that results in the autocorrelation function being close to zero [58].

2.1.2.2 Moving Average (MA) Forecasting

The moving average ($MA(q)$) model of order q works by assuming the model varies with respect to the average according to the model,

$$X_n = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{n-i} + \varepsilon_n, \tag{2.3}$$

where μ is the average of the signal and θ_i are the model coefficients [58].

2.1.2.3 Autoregressive Moving Average (ARMA) Forecasting

These two methods can be combined using the AutoRegressive Moving Average or ARMA model [58], which learns both φ and θ coefficients simultaneously and can be represented with a model of the form,

$$X_n = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{n-i} + \sum_{i=1}^{p} \varphi_i X_{n-i} + \varepsilon_n. \tag{2.4}$$

Using the combined model assumes that the future states are a function of past states along with a component that varies near the mean of the signal. The ARMA model can help reduce the total number of coefficients required for accurate forecasting [58]. Many extensions to this model have also been introduced such as the AutoRegressive Integrated Moving Average (ARIMA) which allows for non-stationary signals, and the Seasonal AutoRegressive Integrated Moving Average (SARIMA) allows for incorporating known a known period into the forecast [58].

2.1.2.4 Random Feature Map Forecasting

Random feature map forecasting is based on a machine learning approach that involves mapping the training data to high dimensional random features to learn a model in the new space [59]. This approach has been used for time series forecasting in [44] where random features of the form $\phi(\mathbf{u}) = \tanh(\mathbf{W}_{in}\mathbf{u} + \mathbf{b}_{in})$ where $\mathbf{W}_{in} \in \mathbb{R}^{D_r \times D}$ and $\mathbf{b}_{in} \in \mathbb{R}^{D_r}$ are the random weight matrix and bias vector for the features sampled from uniform distributions and are fixed for training [44]. D

is the system dimension and D_r is the reservoir dimension or dimensionality of the random feature space. The vector $\mathbf{u} \in \mathbb{R}^D$ is the vector of system states and is assumed to come from a system of the form $\dot{\mathbf{u}} = F(\mathbf{u})$. Mapping the training data into the random feature space using ϕ allows for obtaining a surrogate model propagator map of the form $\Psi_S = \mathbf{W}_{LR}\phi(\mathbf{u})$ where $\mathbf{W}_{LR} \in \mathbb{R}^{D\times D_r}$ optimally maps the random features back to the D dimensional space to predict future states of the system. $\mathbf{W}_{LR}$ is obtained using ridge regression and the optimal solution is computed as $W_{LR} = U\Phi^T(\Phi\Phi^T + \beta I)^{-1}$ where $U \in \mathbb{R}^{D\times N}$ is a matrix of system states, N is the number of training observations, $\Phi \in \mathbb{R}^{D_r\times N}$ is the matrix of random features I is the $D_r \times D_r$ identity matrix and β is the regularization parameter [58].

Random feature map forecasting was applied to the lorenz system $\dot{x}_1 = \sigma(x_2 - x_1)$, $\dot{x}_2 = x_1(\rho - x_3) - x_2$, $\dot{x}_3 = x_1x_2 - \beta x_3$ by setting parameters $\rho = 105$, $\sigma = 10$ and $\beta = 8/3$ to result in chaotic dynamics. A reservoir dimension of 500 was used along with sampling the random features from $(\mathbf{W}_{in})_{ij} \sim \mathcal{U}[-0.1, 0.1]$ and $(\mathbf{b}_{in})_i \sim \mathcal{U}[-4.0, 4.0]$. Using these parameters, a forecast for the lorenz system was generated using 1000 training data points resulting in the forecast shown in Fig. 2.2. We see that the forecast follows the true trajectory for about 150 time steps before it begins to deviate. In this example the time step chosen was 0.01 seconds. An inevitable truth of time series forecasting is that the forecast will always eventually deviate from the true system trajectory as long as the signal is not perfectly periodic which is uncommon for real data. In practice, the signal values stream in as measurements are taken and the new measurements contain valuable information for updating the forecasting model. However, measurement noise significantly hinders the forecast ability of the models presented in this section. These limitations prompt the need for a data assimilation approach for combining forecast models and measurement data to obtain an optimal forecast.

2.1.2.5 Long Short-Term Memory (LSTM) Network

The Long Short-Term Memory (LSTM) network is another advanced RNN approach for time series forecasting. LSTM networks are composed of LSTM units rather than neurons like a traditional FNN or RNN. Over time with RNNs, the backpropagation gradients will either explode or

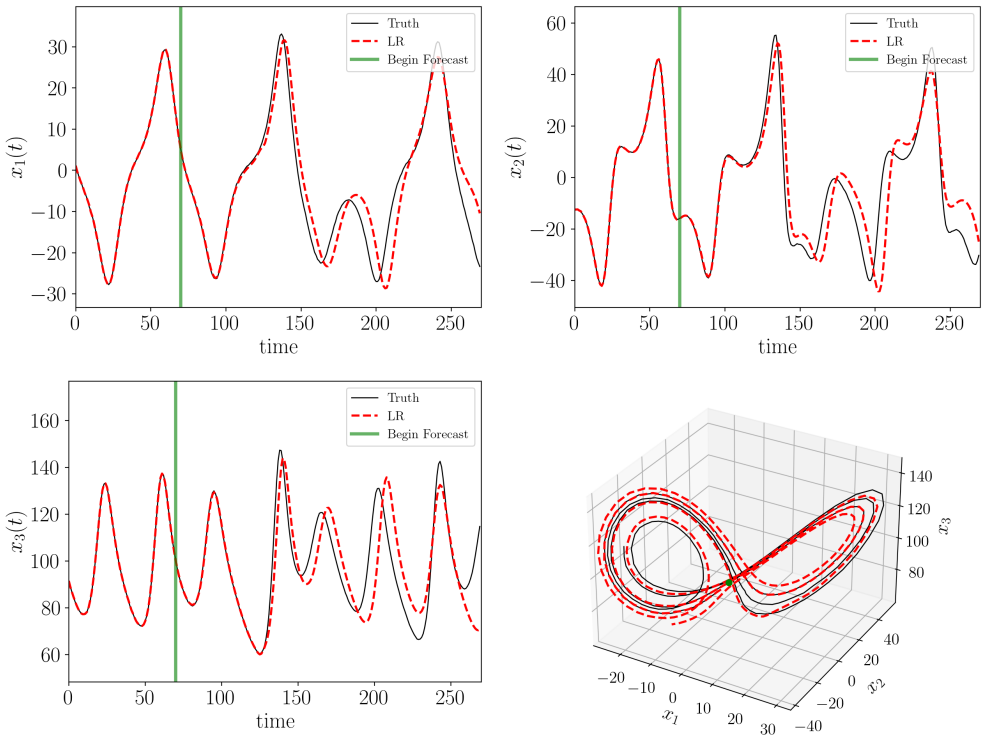


Figure 2.2 Random feature map forecast example using the chaotic lorenz system. the forecast begins at the green vertical line and follows the true trajectory for a period of time before deviating.

vanish due to the low memory bandwidth of these networks (generally 5–10 time steps) [60]. If the gradient explodes, this leads to extreme oscillations in the model weights that do not converge to give good predictions and if the gradient vanishes the model essentially stops learning [60]. LSTM aims to mitigate these issues by incorporating memory cells, input gates, output gates, and forget gates into the network architecture [60]. The inputs to the cell are split into long term and short term memories that consist of input signal states [61]. The long term memories are represented as c_t and short term memories are h_t. These memories get passed through a *forget layer* (f_t) which uses a sigmoid activation function σ to decide which memories to drop from c_t . This is represented mathematically as:

$$f_t = \sigma(W_{xf}^T x_t + W_{hf}^T h_{t-1} + b_f), \tag{2.5}$$

where W_{xf} and W_{hf} are weight matrices corresponding to the current input x_t and short term memory at the previous step h_{t-1} and b_f is a bias [61]. Next, the *main layer* g_t which takes the current

state and short term memory states and acts as a traditional RNN activation layer using corresponding weight matrices and bias vector with the activation equation [61],

$$g_t = \tanh\left(W_{xg}^T x_t + W_{hg}^T h_{t-1} + b_g\right). \tag{2.6}$$

The *input layer* is then used to determine which memories are added to the long term memory c_t using the sigmoid function [61],

$$i_t = \sigma(W_{xi}^T x_t + W_{hi}^T h_{t-1} + b_i), \tag{2.7}$$

with similar weights and biases. The long term memories c_t are updated by incorporating the output from the o_t and i_t using Eq. (2.8) [61],

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{2.8}$$

where $\odot$ is element-wise multiplication. Equation (2.8) serves as the sum of the *forget gate* and *input gate* [61]. Lastly, the short term memories are updated using an *output gate* to determine which part of the long term memories should be the output of the cell using the equation [61],

$$o_t = \sigma(W_{xo}^T x_t + W_{ho}^T h_{t-1} + b_o), \tag{2.9}$$

and the short term memories or predictions of the next signal states are then given by the output gate equation [61],

$$h_t = o_t \odot \tanh\left(c_t\right). \tag{2.10}$$

Therefore, using Equations (2.5) – (2.10) backpropagation can be used to update the model weights and biases using training data to provide future predictions for the states by passing in sequences of past training points with the outputs being the corresponding next points in the sequence.

2.1.3 Topological Approach for Data Assimilation (TADA)

This section presents a general framework for a new data assimilation algorithm leveraging persistence optimization and subsequent sections explore specific instances of the algorithm with different dynamical systems. To optimally combine measurement data and model predictions, I

integrate data driven forecasting models with a new data assimilation scheme driven by TDA to optimally combine the learned model from each signal taking into account data uncertainties in the form of topological differences. The cost function is defined in terms of common discrepancy measures between persistence diagrams such as the Wasserstein distance similar to [57]. Persistence optimization is then used to compute a new, optimal forecast model for all of the input time series signals. I call this method Topological Approach for Data Assimilation or TADA. The TADA algorithm works as follows. Given N sensor observations with additive noise, a general forecast model is generated using any of the methods shown in Section 2.1.2. In general, any forecasting method can be used if it fits the form of Eq. (2.11) and is differentiable with respect to the model weights. The forecast model can be defined as,

$$\mathbf{x}_{n+1} = G(\mathbf{X}_p, \mathbf{w}, \mu), \tag{2.11}$$

where $\mathbf{x} \in \mathrm{R}^N$ is a vector of system states, n is the time index, $\mathbf{X}_p = (\mathbf{x}_{n-p}, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n)$ is a matrix of the $p+1$ previous states, w is the set of model weights that determine the output of the forecast function G and μ is a set of hyperparameters such as the random feature weights in random feature map forecasting. All of the models shown in Section 2.1.2 can be written in this form. Typically, training data is used to minimize the error between the model predictions on the training set and the measurements by optimizing w. Once a framework for G is fixed, a forecast is generated $\mathcal{W}$ points into the future and state measurements are collected over the same forecast window. This gives us for the nth data assimilation window $\mathcal{W}_n$ two point clouds corresponding to the forecast states and their measured counterpart. Persistence diagrams are computed for these two point clouds and persistence optimization is used to update the weights of the forecast model so that it better fits the observed response. To update the model G using the nth assimilation window, the model weights w are varied such that the differences between the persistence diagrams of the predicted and measured states are minimized over $\mathcal{W}_n$. This is achieved using a cost function $J(W, \hat{W}) : \mathbb{R}^4 \to \mathbb{R}$ where $W = (W_0, W_1)^T$ is the vector with components consisting of the 0D and 1D Wasserstein distances between the model and measurement persistence diagrams, respectively. Similarly, $\hat{W}$ is the vector containing Wasserstein distances between the empty persistence diagram

and the model error persistence diagram or the persistence diagram of the point cloud generated by taking the differences between the model predictions and measurements. I found that both W and $\hat{W}$ were important because if only W is used, many solutions exist to the optimization problem with similar but time-shifted shape as the measurements leading to undesirable temporal shifts in the predictions.

J is taken to be the sum of the 0D and 1D Wasserstein distances for W and $\hat{W}$. Note that the Wasserstein distance between two persistence diagrams is computed using,

$$W_p(PD_1, PD_2) = \inf_{\pi} \left(\frac{1}{n} \sum_{i=1}^{n} \|X_i - Y_{\pi(i)}\|^p \right)^{\frac{1}{p}}, \tag{2.12}$$

where PD_1 and PD_2 are the persistence diagrams in the desired dimension, X_i are the persistence pairs of PD_1 and $Y_{\pi(i)}$ are persistence pairs of PD_2 optimally matched to PD_1 pairs to minimize the total distance (using a ℓ_2-norm metric) between the points with optimal transport. For this work I use $p = 1$ for simplicity. The model persistence diagrams are inherently functions of the model weights $\mathbf{w}_n$ so minimizing J will yield an updated forecast model weights $\mathbf{w}_{n+1}$ to get the analysis solution for the next assimilation window. The next assimilation window is obtained by taking the next $\mathcal{W}$ predictions and measurements shifted by a stride length $\mathcal{S}$. To minimize topological changes between windows, I set $\mathcal{S} = 1$ for this work. I apply this pipeline over many assimilation windows, and obtain optimized model weights $\hat{\mathbf{w}}$. This approach is pictorially represented in Fig. 2.3. Note that the size of the very first window starts with just two points and as new measurements are obtained it increases to the specified window size and slides across the signal after that point.

I note that the first example of TADA uses random feature map forecasting and is shown in Section 2.1.4. It is assumed that the model predictions in future times are generally close to the measurements and the weights have already been optimized on the training set. The weights are then further updated as new measurement data streams in. A slightly different approach is taken in Section 2.2 where an LSTM forecast framework is used for G and minimal model optimization is performed on the training data prior to running TADA. This approach is more general and

does not require the model forecast to initially be close to the training data. The only difference between these two approaches is a single term in the TADA cost function that further constrains the problem to the training set. The cost function in this case becomes $J = J_1(W, \hat{W}) + J_2(W, \hat{W})$, where J_1 corresponds to the original cost function using persistence diagrams of the sliding window point clouds and J_2 minimizes topological differences between randomly sampled model and measurement point clouds of fixed size on the training set. As the window slides along the signals measurements that exit the window are then added to the set of points that can be randomly sampled for J_2. Together, these two cost function terms allow the model to converge to the original model by minimizing errors on the training set, but also allows the model to learn from incoming data with the sliding window approach. This approach does not contain any restriction on

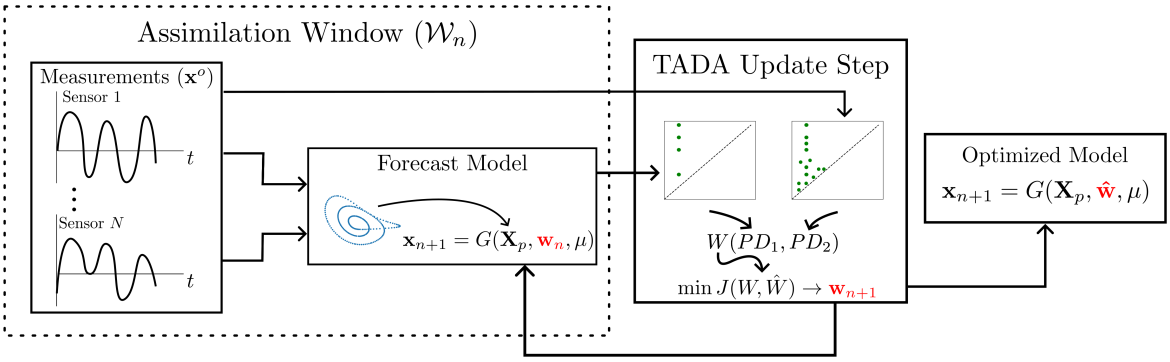


Figure 2.3 Assimilation window update diagram showing the updated model giving an improved forecast.

the noise model in the measurements, and when combined with forecasting methods that are also independent of the noise characteristics in the signal it enables a noise agnostic data assimilation. Figure 2.4 shows a snapshot of an assimilation window where the forecast (blue) deviates from the measurements (red). Persistence optimization is used in the second box to minimize the Wasserstein distance between the persistence diagrams giving a forecast model that is much closer to the measurement. The process then repeats to update the forecast weights for the next window.

2.1.4 TADA using Random Feature Maps

For the first application of TADA, I set the model to be $G = W_{LR} \tanh(W_{in}\mathbf{x}_n + b_{in})$ with $p = 0$ so only the previous state is used to predict the next, $w = W_{LR}$ and $\mu = \{W_{in}, b_{in}\}$ where W_{LR}

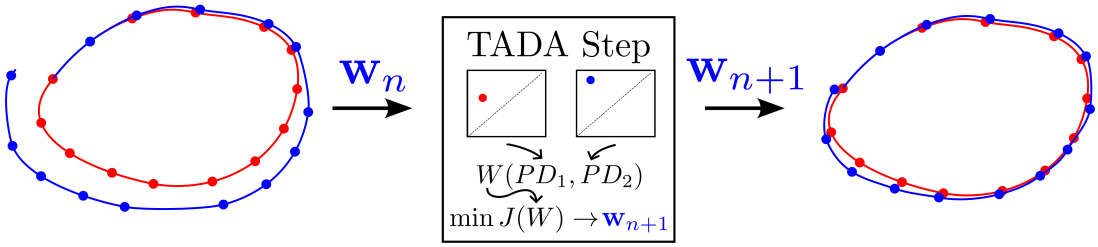


Figure 2.4 Assimilation window update diagram showing the updated model giving an improved forecast.

corresponds to the random feature map forecasting model that has already been optimized on the training data. In subsequent sections, I perform testing of this method on the chaotic Lorenz system to show that TADA extends the forecast time of the system.

2.1.4.1 Lorenz System Results

In this section, I focus on testing TADA using the chaotic Lorenz dynamical system defined as, $\dot{x} = \sigma(y-x)$, $\dot{y} = x(\rho - z) - y$, $\dot{z} = xy - \beta z$ where σ, ρ, β were set to 28, 10 and $8/3$ respectively for chaotic dynamics. Simulations were conducted using randomly selected initial conditions by sampling from the standard normal distribution. Simulations were sampled at a frequency of 50 Hz for 500 seconds and the last 6000 points were taken to remove transient behavior. Noise was added to the simulation signals using the form from [44] where the observational error covariance matrix is defined as Γ and the noise is added as $\Gamma^{1/2}\eta$ where $\eta \in \mathbb{R}^D$ is sampled from the noise distribution. In [44], the authors take $\Gamma = \eta I$, however this effectively applies different noise levels to each system state due to the differing amplitudes. Therefore, I chose to apply different noise amplitudes η_i for each state based on a specified Signal-to-Noise Ratio (SNR) in decibels (dB) using,

$$\eta_i = A^i_{\text{signal}} 10^{-\text{SNR}_{\text{dB}}/20}, \tag{2.13}$$

where A^i_{signal} is the RMS amplitude of signal i and SNR_{dB} is the signal-to-noise ratio in decibels. This makes Γ a diagonal matrix with different noise amplitudes to apply the same noise level to each state.

2.1.4.2 Random Feature Map Parameters

Data driven models were generated for each simulation of the Lorenz system using the random feature map method from Section 2.1.2.4. For consistency, I chose to conduct all tests using a reservoir dimension of 300 ($D_R = 300$). Random weights were sampled from the uniform distribution $\mathcal{U}(-0.005, 0.005)$ and the bias vector entries were sampled from $\mathcal{U}(-4.0, 4.0)$. The regularization parameter, β, was set to 4×10^{-5}. Accuracy of the random feature map method is heavily dependent on the parameters chosen for the random features and regularization [43]. In this work, I use the same parameters from [43] specific to the chaotic Lorenz system, but choosing these parameters for an arbitrary system is highly nontrivial. In all simulations, I used 4000 points for training the random feature map model.

2.1.4.3 Forecast Time

To quantify the accuracy of the forecasts and assimilation updates, I use the relative forecast error,

$$\mathcal{E}(t_n) = \frac{||u_{\text{valid}}(t_n) - u_n(t_n)||^2}{||u_{\text{valid}}(t_n)||^2}, \tag{2.14}$$

where $u_{\text{valid}}(t_n)$ is the validation set or measurement data and $u_n(t_n)$ is the analysis or optimal estimation. The forecast horizon is computed based on a threshold of the forecast error. The forecast time τ_f is the maximum time such that $\mathcal{E}(\tau_f) < \theta$. In [44] θ was taken to be 0.05 and the 2-norm was used for all computations. It is standard practice to quantify DA results in terms of Lyapunov times where the forecast time is scaled by the maximum Lyapunov exponent of the system λ. In this paper, all results are from the Lorenz system where the maximum Lyapunov exponent was taken to be $\lambda = 0.91$ [44].

2.1.4.4 TADA Forecast

The TADA algorithm was first applied to a single Lorenz system trajectory to demonstrate a successful DA update. I used a window size of 50 points with 50 optimization epochs so 100 new measurements are used for data assimilation. In Fig. 2.5 I show a two dimensional projection of the state space trajectory just after the training data. The blue points are the measurements, the dashed green curve is the initial forecast obtained using random feature maps and the solid red curve is

the analysis solution after 50 data assimilation updates. We see that the analysis solution appears to follow the measurements better generally. For the trajectory in Fig. 2.5, the corresponding time domain results are shown in Fig. 2.6 with the points used for assimilation being inside of the blue rectangles. The time domain results clearly show the improvement in the forecast accuracy of the analysis solution over the initial forecast. Using Eq. (2.14), the forecast accuracies are quantified in Lyapunov time units to be approximately 0.89 for the initial forecast and 4.08 for the optimized model. A learning rate decay of 0.99 was applied for obtaining these results. In subsequent sections I perform hyperparameter tuning to select optimal parameters for this system.

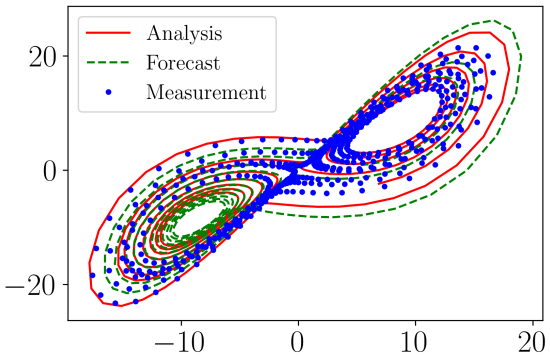


Figure 2.5 Two dimensional state space projection of noisy Lorenz system measurements (blue points) with the initial forecast (green dashed) and the TADA forecast (red solid). The initial forecast time was 0.89 and the TADA forecast time increased to 4.08 Lyapunov time units.

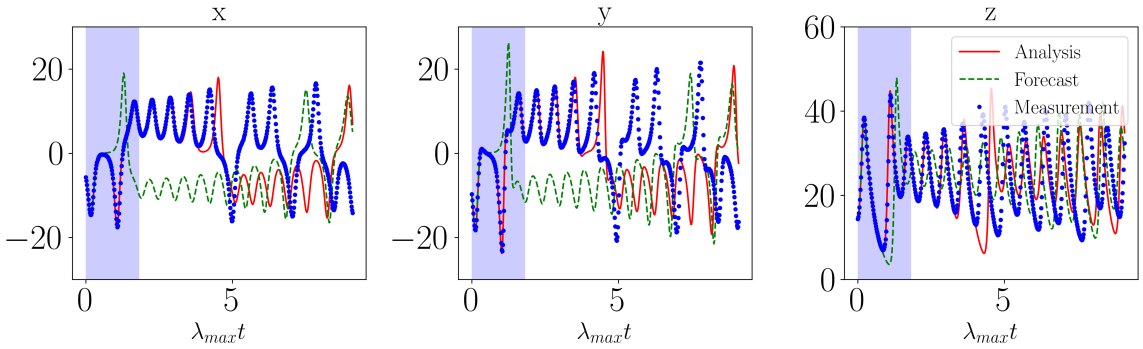


Figure 2.6 Time domain plots of noisy Lorenz system measurements (blue points) with the initial forecast (green dashed) and the TADA forecast (red solid). The initial forecast time was 0.89 and the TADA forecast time increased to 4.08 Lyapunov time units. The blue regions indicate the measurements that were used to improve the forecast.

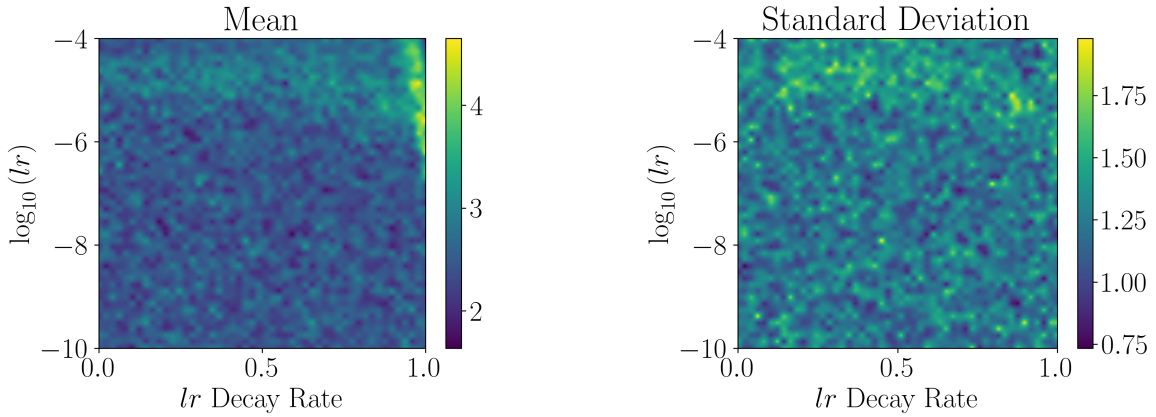
2.1.4.5 Learning Rate Dependence

The TADA algorithm requires choosing a learning rate and learning rate decay rate for the gradient descent operation to determine the step size in the loss function space and speed up convergence of the optimization problem. For a typical application of gradient descent, learning rates are chosen near 10^{-2} [62] because any smaller and many problems will not converge in a reasonable number of steps. However, for the TADA algorithm, the learning rates need to be chosen many orders of magnitude smaller than a traditional gradient descent problem. This is because for a chaotic system, the model coefficients are highly sensitive to changes and, if a change is significant enough, it results in a drastically different forecast prediction. Note that for all testing in this section a window size of 50 was arbitrarily chosen. So as measurements stream in, the window size grows until it reaches 50 and then slides across the signal for the final 50 steps. In total, 100 incoming measurement points are used for TADA update steps and the Adam optimizer was used for all gradient descent steps.

To determine the optimal learning rate, I chose to conduct testing over a range of learning rates and learning rate decay rates at different initial conditions using noise-free signals. This test was conducted by simulating the chaotic Lorenz system at 50 different initial conditions sampled from the standard normal distribution. The mean and standard deviation forecast times were computed on 50×50 grid in the learning rate and learning rate decay rate parameter space. The learning rate was varied from 10^{-10} to 10^{-4} and the decay rate was varied between 0 and 1. First, I generated these results using only the J_1 cost function term and the results were linearly interpolated to a resolution of 300×300 and the results are shown in Fig. 2.7. We see in Fig. 2.7 (a) that for learning rates between 10^{-6} and 10^{-4} and decay rates near one the average forecast time is more than 4 Lyapunov times whereas we will see the linear regression model forecast time is always near 2 Lyapunov times. However, the range of parameters where the forecast time increases is relatively small.

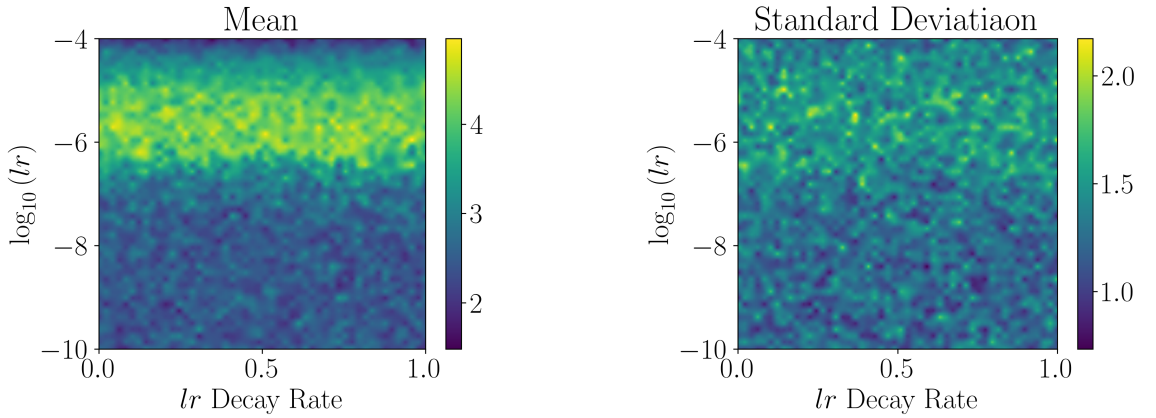
The standard deviation of the forecast times is shown in Fig. 2.7 (b) for reference. Conversely, when I introduce the J_2 cost function term in addition to J_1, the results are much more robust with

larger forecast times on average as shown in Fig. 2.8 (a) with the standard deviation in Fig. 2.8 (b). We see that the average forecast time between 10^{-6} and 10^{-4} is near 4 Lyapunov times for nearly all decay rates. These results demonstrate the importance of the J_2 term. I chose a decay rate of 0.99 for the rest of the results.



(a) (b)

Figure 2.7 TADA (J_1) learning rate and learning rate decay rate analysis at for the chaotic Lorenz system. The average forecast times over 50 iterations are plotted in Lyapunov time units with respect to the TADA learning rate used on a log base 10 scale and the learning rate decay rate.



(a) (b)

Figure 2.8 TADA ($J_1 + J_2$) learning rate and learning rate decay rate analysis at for the chaotic Lorenz system. The average forecast times over 50 iterations are plotted in Lyapunov time units with respect to the TADA learning rate used on a log base 10 scale and the learning rate decay rate.

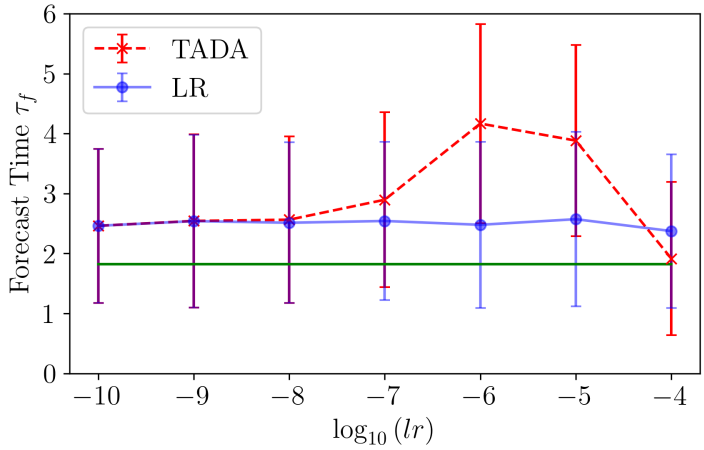


Figure 2.9 TADA learning rate analysis at for the chaotic Lorenz system. The average forecast times over 500 iterations are plotted in Lyapunov time units with respect to the TADA learning rate used on a log base 10 scale. Error bars indicate one standard deviation, the red x's are the TADA results and the blue points are results from random feature maps. The solid green line represents the amount of incoming data that was used to improve the model using TADA updates.

Next, we compare the TADA results at varying learning rates to the random feature map Linear Regression (LR) method. This was done by testing the algorithm at a noise level of 50 dB and varying the TADA learning rate with a decay rate set to 0.99 to decrease the learning rate by 1% for each assimilation step. Each learning rate was tested for 500 different initial conditions for this test and the results for noise free signals are shown in Fig. 2.9. We see that the forecast time has the largest improvement for learning rates of 10^{-6} and 10^{-5} with 10^{-6} being slightly higher on average. Note the green horizontal line indicates the 100 incoming data points that were used for TADA update steps after the original training set for the LR method. I chose to explore the implications of choosing 10^{-6} and 10^{-5} for the remainder of this paper. While this method requires significant parameter tuning to get improved forecast times, this analysis can be conducted on a training set before running the TADA algorithm on a specific system and because the results for one learning rate do not depend on another it can be conducted in parallel to find the learning rate that maximizes forecast time. This process is essentially measuring the sensitivity of the model to changes in the coefficients.

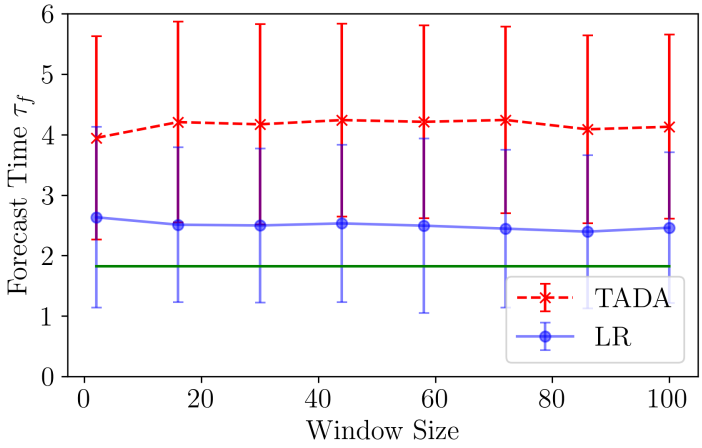


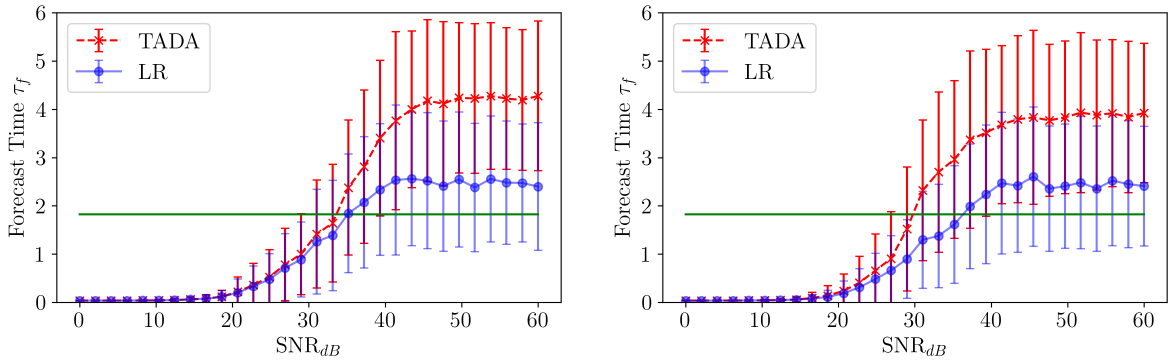
Figure 2.10 Average TADA forecast times plotted in Lyapunov time units with respect to the sliding window size for TADA. Red x's indicate the TADA results and blue points are the random feature map results averaged over 500 randomly chosen initial conditions with error bars indicating one standard deviation. The solid horizontal green line indicates the amount of incoming data that was used to improve the model.

2.1.4.6 Window Size Dependence

To use the TADA algorithm, a sliding window width must be chosen for computing persistence. If the window size is too small, persistence features will not show up in the persistence diagrams and if the window size is too large the computation times will be too long. To determine the optimal window size, the average forecast times for the chaotic Lorenz system over 500 iterations were computed with respect to the window size and the results are shown in Fig. 2.10. A learning rate of 1×10^{-6} was used for this analysis with a decay rate of 0.99. I computed the forecast times at 500 randomly chosen initial conditions for 8 different window sizes between 0 and 100. This analysis was performed on signals with 50 dB of added white noise and the results are shown in Fig. 2.10. These results suggest that the forecast times are independent of window size as long as some persistence features are present. I found that a window size of 50 points was a good balance of computation time and significant topological features being present in the persistence diagrams, but results may vary for other systems depending on the sampling rate so this parameter can be decreased to improve computation times if needed and increased if more persistence features are required.

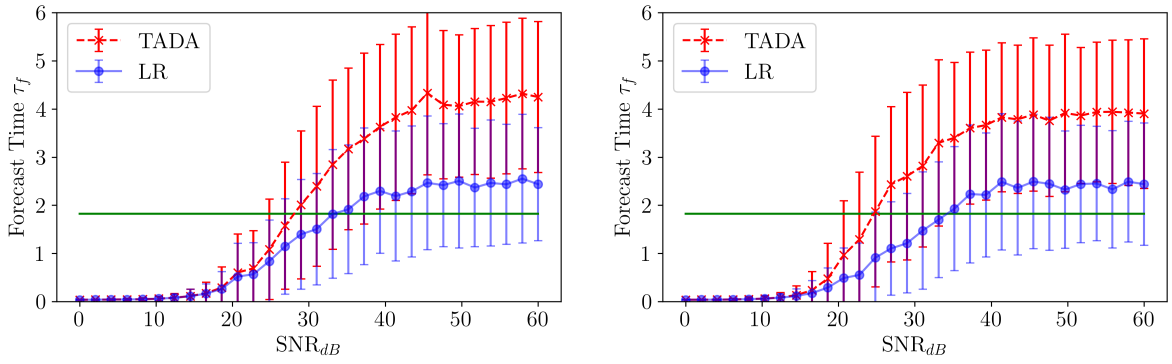
2.1.4.7 Noise Robustness

To test the noise robustness of TADA, I measure the forecast time of the optimized models when artificial noise is added to the training signals and measurements. Many DA methods make assumptions on the noise distribution in the signal or require knowledge of the noise statistics such as in [44], but TADA performs DA updates based only on topological differences between the forecast and measurements and in theory can work for any noise distribution with reasonable SNR. It is common for systems to have colored/correlated noise distributions in practice [63, 64] so the white noise assumption of other methods can be a limiting factor. To validate these claims, I used three different noise distributions: white, pink and brownian for our testing and applied the noise additively at varying SNR to the signals. These colored noise distributions were generated using the colorednoise python library which implements the algorithms from [65]. This test was conducted over a range of 30 signal-to-noise ratios varying from 0 to 60 dB and the forecast time was measured for 500 randomly chosen initial conditions at each SNR with learning rates of 10^{-6} and 10^{-5}. The results of this test are presented in Fig. 2.11. For Gaussian white noise, we see in Fig. 2.11 (a) that on average TADA improves the forecast time beyond the additional measurements used down to an SNR of approximately 32 dB with a learning rate of 10^{-6}. However, using a learning rate of 10^{-5} results in slightly lower forecast times at low noise levels, but the algorithm is more noise robust down to approximately 29 dB as shown in Fig. 2.11 (b). Similar behavior is observed for pink and brownian noise in Fig. 2.11 (c-f). Interestingly when using pink and brown noise distributions in Figs. 2.11 (d) and (f) TADA is more robust to noise down to an SNR of about 23 dB. Below these SNRs, the average forecast times are still improved over the LR method alone, but it does not improve the forecast time beyond the additional data that was used to improve the model (solid green horizontal line). For high SNR, the forecast time is improved on average by approximately 1.5 lyapunov times over the LR method and as more incoming measurements are used this difference increases if the correct learning rate and decay rate are chosen.



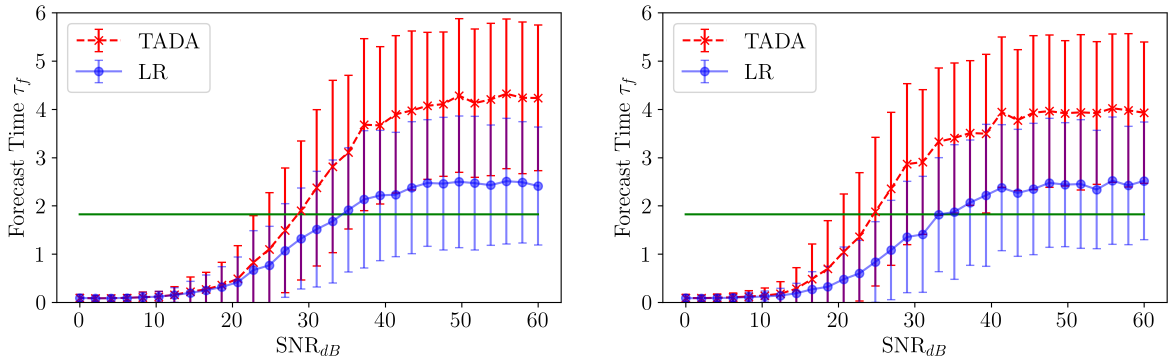
(a) White Noise (10^{-6})

(b) White Noise (10^{-5})



(c) Pink Noise (10^{-6})

(d) Pink Noise (10^{-5})



(e) Brownian Noise (10^{-6})

(f) Brownian Noise (10^{-5})

Figure 2.11 Average TADA forecast times plotted in Lyapunov time units with respect to the SNR of the signals using three different noise distributions and two learning rates. Red x's indicate the TADA results and blue points are the random feature map results averaged over 500 randomly chosen initial conditions with error bars indicating one standard deviation. The solid horizontal green line indicates the amount of incoming data that was used to improve the model.

2.1.4.8 Computation Times

The TADA computation time was measured for the noise robustness testing in Section 2.1.4.7 by timing 500 iterations at each point with each iteration containing 100 DA update steps. These computations were done in parallel with 50 cores/jobs using an Intel Xeon Silver 4216 2.10 GHz CPU with 64 Gb of RAM. The average and standard deviation time for 500 iterations was computed for each noise color. To get the average iteration time I assume that each core/job performed 10 of the iterations and divide the average time by 100 steps and to get the TADA step time. In other words I divide the average time for 500 iterations by 1000 to get the average TADA step time. The resulting computation times are shown in Table 2.1. While these computation times are likely not fast enough to run the algorithm in real time, some systems may only obtain one measurement per second or even longer so in those cases this method can be used online and if the number of model parameters is optimized further the step time can be reduced.

Table 2.1 TADA Computation Times over 500 iterations for each noise distribution tested. The DA step time was determined by assuming that each of the 50 cores used in the parallel computation handled 10 of the 500 iterations and that time was then divided by the 100 DA steps in each iteration.

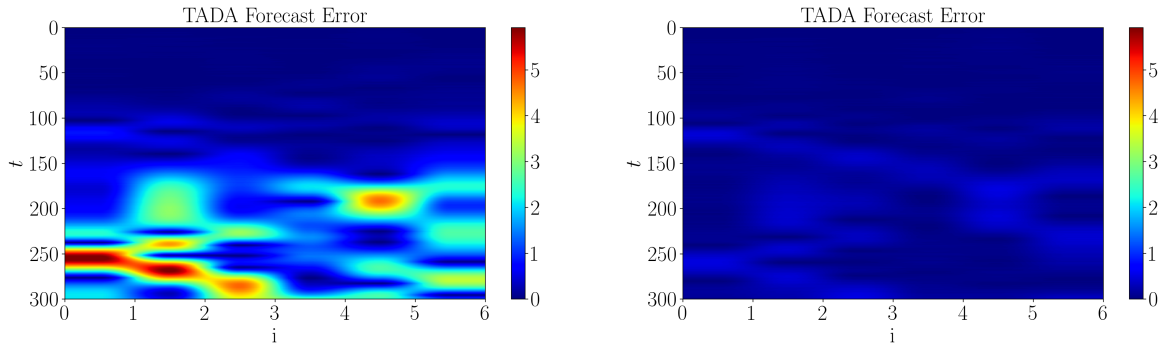
Noise Color	500 Iteration Mean (sec)	500 Iteration Standard Deviation (sec)	TADA Step Time (sec)
White	1067.31	11.28	**1.07**
Pink	1069.51	7.11	**1.07**
Brownian	1077.59	3.55	**1.08**

2.1.5 Lorenz 96 Example

All of the results to this point have been from the 3D Lorenz 63 system. Here I demonstrate the TADA forecast capability on a higher dimensional system by generating optimal models for a variant of the Lorenz 96 system. The Lorenz 96 model is given by:

$$\frac{dX_i}{dt} = (X_{i+1} - X_{i-2})X_{i-1} - X_i + F, \tag{2.15}$$

where X_i is the i-th state, with $i = 1, \cdots, N$, F is the forcing parameter and $X_{-1} = X_{N-1}$, $X_0 = X_N$, $X_{N+1} = X_1$ with $N \geq 4$ [66]. Interestingly, this system allows for specifying the dimension as a system parameter. Physically, this system represents climate dynamics on a circle and I use $F = 8$



(a) Initial Error (b) Final Error

Figure 2.12 (a) Initial and (b) final 6 dimensional Lorenz 96 TADA error between the updated TADA model and the measurements after 100 TADA update steps at a learning rate of 1×10^{-6}.

to yield a chaotic response [67]. For these results, I choose $N = 6$ to start and rather than using the forecast time from Eq. (2.14) I use the absolute difference between the states and measurements to plot the error in space and time. I simulated the Lorenz 96 system for 200 time units using a time step of 0.01 and removed 25 time units to allow transient behavior to dissipate. 170 time units were used for training the random feature map model with $w = 0.01$, $b = 0.1$ and a reservoir dimension of $D_r = 500$ was used. White noise was added to the signals with an amplitude of $\eta = 0.01$ and 100 TADA steps were taken at a learning rate of 1×10^{-6} with a decay rate of 0.99. The initial and final forecast absolute difference errors are shown in Fig 2.12 (a) and (b) respectively. We see that the initial forecast had low error out to about 150 time steps and after 100 TADA steps the model and measurement discrepancy approached zero out to 300 time steps. I note that this example is presented to demonstrate that our method works on higher dimensional systems, but I emphasize the importance of the hyperparameter tuning for this method as arbitrarily setting the parameters will likely yield poor results.

2.2 Hall Effect Thruster Forecasting using Topological Approach for Data Assimilation

This section serves as validation for TADA using high-fidelity Hell Effect Thruster (HET) simulation data from the Air Force Research Laboratory (AFRL) rocket propulsion division in California. A number of generalizations for TADA are also addressed by this application such as using a different forecast function (LSTM), forecasting very high-dimensional field data and

integrating training data into the TADA pipeline so the initial forecast does not need to be initially close to the measurements.

2.2.1 Hall Effect Thruster Background

HETs are a class of ion thrusters that generate thrust by accelerating ions through an electromagnetic field to eject heavy ionized gas particles from the spacecraft. These thrusters are highly complex due to the interplay of electrodynamics, fluid dynamics, fluid-structure interaction, and quantum mechanics which makes them difficult or impossible to model analytically. This is further complicated in experimental settings where the thruster is placed in a vacuum chamber to simulate the environment of space. The electromagnetic field interacts with the chamber in these experiments leaving researchers with data that likely does not accurately represent the thrusters behavior in space due to ground effects. HETs have shown great potential for future space flight due to their ability to greatly increase the lifespan of the thruster to over 10,000 hours [68]. However, some of the operating modes in the HETs lead to undesirable system dynamics such as high amplitude, low frequency breathing mode oscillations in the thrust produced. This phenomena is due to a complex interaction between neutral and ionized particles and leads to sub-optimal performance of the thruster [69]. Another behavior that these thrusters exhibit is a result of high energy ions causing erosion of critical surfaces for the thruster and the space craft which is detrimental to many of the components on board [69]. These operating modes are induced by changes to the system parameters (shown in Fig. 2.13) such as the discharge voltage V_d (the voltage between the anode a and cathode c), mass flow rate of gas $\dot{m}$, magnetic field strength and topology $\vec{B}$, electric field strength $\vec{E}$ and discharge current I_d [68, 69]. Thus the ability to develop accurate and optimal data-driven models to predict future system states based on measurement data is crucial for safe operation of these thrusters making HETs a good candidate for time series forecasting and data assimilation methods such as TADA.

2.2.1.1 LSTM Forecast Function

For this application, I chose to use the LSTM forecast function due to its ability to maintain long term memories and they can be trained faster than traditional RNN methods [61]. The LSTM

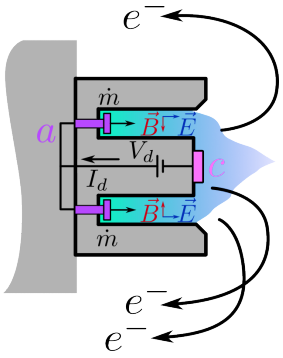


Figure 2.13 Hall-Effect Thruster (HET).

framework also allows for inputs and outputs to be sequences of points rather than just a single point which provides a more reliable forecast because more information is contained in the sequence. The LSTM forecast function can be written in the form $\mathbf{x}_{n+1} = G(\mathbf{X}_p, \mathbf{w}, \mu)$ by combining Eqs. (2.5)–(2.9) with $c_0 = 0$ and $h_0 = 0$ to get the following update relationships for the cell and hidden states of the network,

$$\begin{aligned} c_n =& \sigma(W_{xf}^T \mathbf{X}_p + W_{hf}^T h_{n-1} + b_f) \odot c_{n-1} \\ &+ \sigma(W_{xi}^T \mathbf{X}_p + W_{hi}^T h_{n-1} + b_i) \odot \tanh(W_{xg}^T \mathbf{X}_p + W_{hg}^T h_{n-1} + b_g) \\ h_n =& \sigma(W_{xo}^T \mathbf{X}_p + W_{ho}^T h_{n-1} + b_o) \odot \tanh(c_n). \end{aligned} \tag{2.16}$$

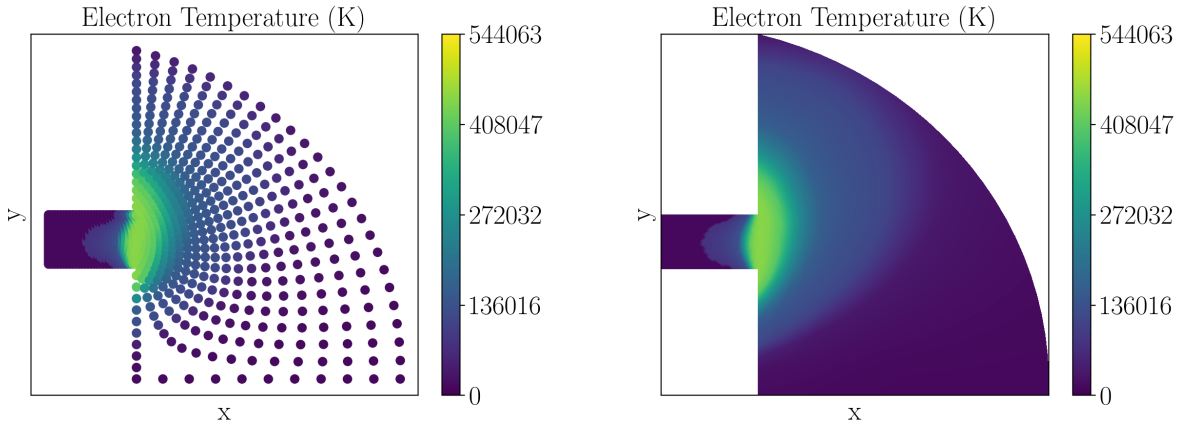
These relationships are then combined with a dense or fully connected layer to map back to the dimension of the input matrix $\mathbf{X}_p$ to predict the next point using the equation,

$$\mathbf{x}_{n+1} = W_d h_n + b_d, \tag{2.17}$$

where W_d and b_d are the corresponding learned weight matrix and bias vector for the dense layer to predict the next state of the system. These three update rules combined fit the form $\mathbf{x}_{n+1} = G(\mathbf{X}_p, \mathbf{w}, \mu)$ for use with the TADA algorithm where all of the model weights and biases are contained in $\mathbf{w}$ and the initial cell and hidden states are captured in μ. Note that there is not an easily expressed explicit form for G because the update rule for c_n is recursive. Using backpropagation, the model parameters are trained by minimizing the error between predictions and the training data and an optimal model is learned. Note that for this work the inputs are fixed to being the previous 5 states and the output is only a single point for consistency with Eq. (2.11).

2.2.1.2 HET Data

The data from AFRL was generated using a software package called HPHall which uses a hybrid Particle In Cell (PIC) method to model particles on different scales in the system [70] and the Direct Simulation Monte Carlo (DSMC) method for simulating collisions [71]. Specifically, the SPT-100 thruster was simulated for this work at a mass flow rate of $\dot{m} = 5.01 \times 10^{-6}\ kg/s$ at a discharge voltage $V_d = 300\ V$. While this data is obtained from a simulation, the highly complex behavior of these thrusters make them incredibly difficult to accurately model analytically, and the methods used for simulation ultimately leave the data contaminated with noise. Though the noise distribution may be known by the researchers at AFRL, for this work, I assume that I do not have any information about the noise statistics by using TADA. Simulation data was provided over 20,000 time steps at 1,250 locations in the thruster plume measuring 7 field states at each positional location. The measured states include electron temperature (T_e), electric potential (ϕ), neutral number density (n_n), electron number density (n_e), ion production rate ($\dot{n}_i$), axial ion velocity (v_{iz}) and radial ion velocity (v_{ir}). Due to the radial symmetry of the thruster, the simulation data forms a radial slice in the thruster plume. Figure 2.14 shows plots of the thruster plume at a single point in time using the electron temperature field. We see that the grid of points is highly nonuniform due to



(a) Data Locations (b) Contour Plot

Figure 2.14 Example plots of HET data using a single frame of the electron temperature signals. (a) shows a scatter plot of the spatial locations at one moment in time and (b) shows the corresponding contour plot of the field.

the optimizations performed by HPHall. When using the provided spatial locations for each point, the data can be arranged into this form to view the behavior of the thrust field, but in practice it is easier to work with a data matrix $X \in \mathrm{R}^{n_x \times n_t}$ where $n_x = 1,250$ is the number of spatial locations and $n_t = 20,000$ is the number of time points. Slices of this data matrix can also be visualized as images to study spatio-temporal patters in the data. Figure 2.15 shows the transpose of the data matrix for the first 2,500 time points for the electron temperature. We see the clear oscillating pattern present in the temporal signals and this figure also highlights many of the spatial locations remaining constant in time. These points largely correspond to the lower right corner of Fig. 2.14.

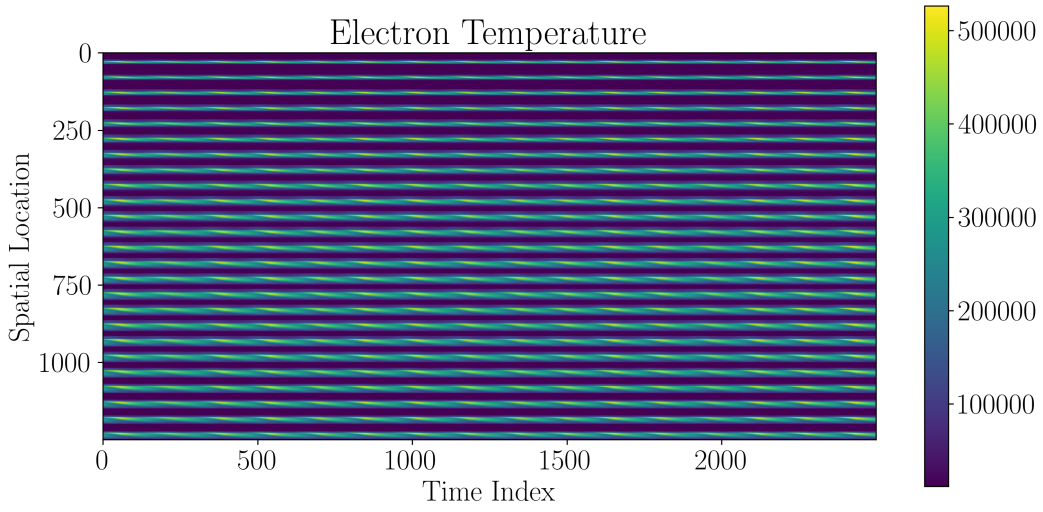


Figure 2.15 Electron temperature data matrix for the first 2,500 time points.

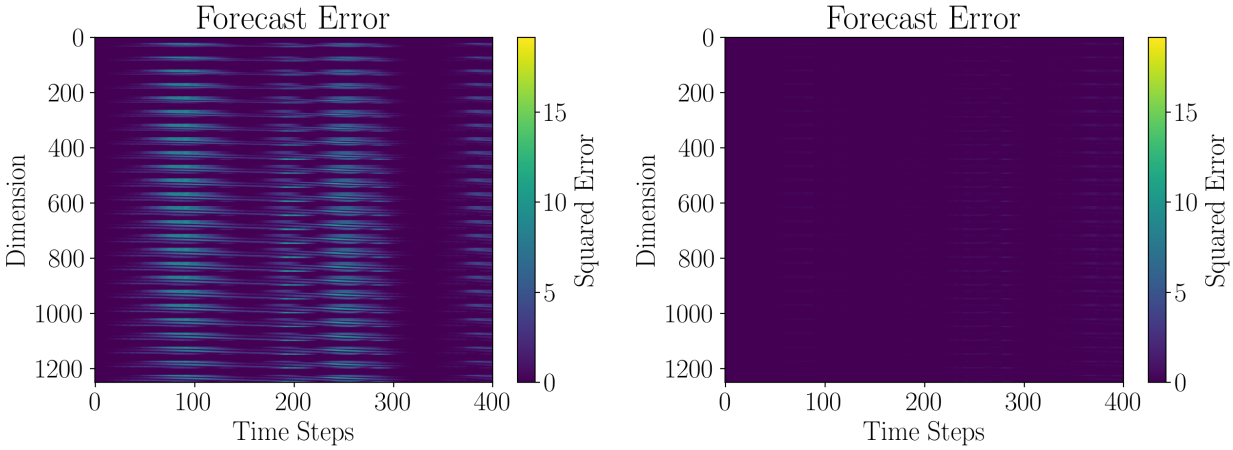
2.2.2 Hall Effect Thruster TADA Results

LSTM networks were trained using 5000 HET measurement points and 500 LSTM units were used to generate weights for the system. The models were trained on sequences of 5 points which were used to predict the next point. To demonstrate the robustness of TADA on this data, the models were fit using only 5 optimization epochs with batches of size 50 and the full TADA cost function ($J = J_1 + J_2$) was used to improve the model. A window size of 200 points was used

for these results to capture the topology of the signals. These results demonstrate the flexibility of the TADA algorithm on periodic data by showing successful forecast improvements on high dimensional spatiotemporal data, using a different forecast function with the ability to predict future points based on sequences of previous points and using the full TADA cost function to show that the model predictions do not necessarily need to be close to the measurements in this case. In this section, the forecast accuracy is quantified using the squared differences between the measurements and model predictions and the color scale on the error plots is set based on the largest difference in the first TADA step. This is because many of the states remain very close or equal to zero so dividing by the measurements to compute a percent error leads to skewed error results. Many machine learning models perform better on data that is scaled or normalized so for these results, all signals were scaled using their corresponding z-scores.

2.2.2.1 Electron Temperature (T_e)

The electron temperature (T_e) was tested first using this approach. Figures 2.16 (a) and 2.17 show the initial forecast results for the electron temperature. It is clear from these results that the forecast does not accurately reflect the measurements and the model started quite far from the measurements. TADA was then applied to this model using a learning rate of 10^{-5} with a decay of 1% for every step. Because the J_2 cost function term was used, the learning rate decision is not as critical. If the learning rate is too high the predictions oscillate around the measurements eventually converging as the learning rate decays. For this application the learning rate was manually tuned to 10^{-5} as this was found to converge quickly while minimizing oscillations of the forecast around the true minimum of the loss function. After 200 TADA steps, the resulting forecast error and forecast for dimensions 80–82 are shown in Figs. 2.16 (b) and 2.18. We see that the forecast error decreased dramatically with a maximum initial error of approximately 20 and the final maximum error was 0.98. The forecast shown in Fig. 2.18 also shows a significantly more accurate prediction that more closely matches the measurements. While 200 points were used for data assimilation, the forecast remains accurate outside of the DA window through the next 200 time points without over fitting to the noise in the signal.



(a) Initial T_e error

(b) Final T_e error

Figure 2.16 Error plots for the electron temperature before and after 200 TADA steps.

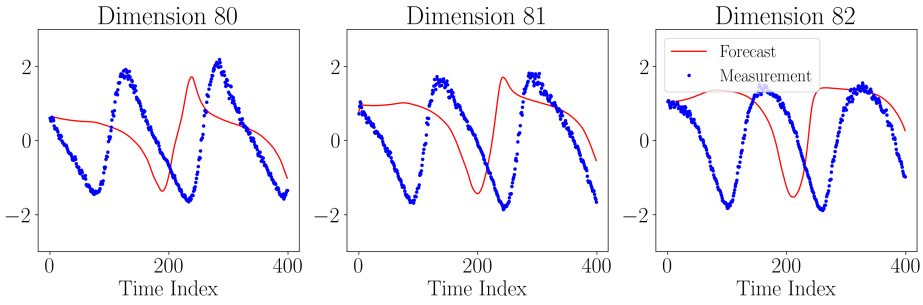


Figure 2.17 Initial T_e forecast for dimensions 80-82.

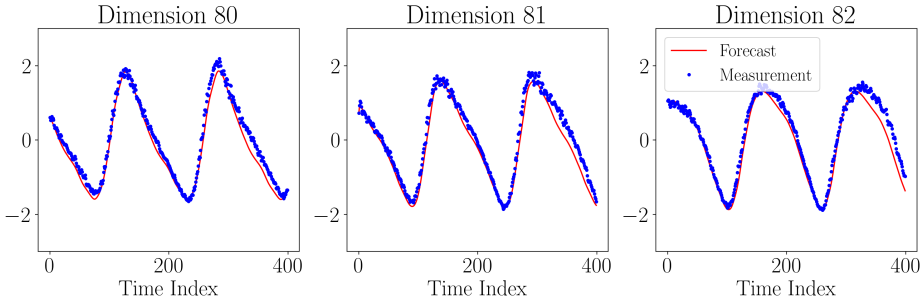


Figure 2.18 T_e forecast for dimensions 80-82 after 200 TADA steps.

2.2.2.2 Electric Potential (ϕ)

The electron temperature data was analyzed first because it is the most well behaved variable in this data set. By well behaved I mean it contained minimal outliers. This is not the case for other HET variables such as the electric potential. If I naively generate LSTM models from the ϕ field data, the model does not converge to an accurate solution and running TADA on this model does not improve the forecast due to some of the states containing outliers. This can be easily visualized by plotting the scaled ϕ fields. We see in Fig. 2.19 (a) that the original scaled data with no smoothing appears to have most z-scores relatively close to zero, but some points in the data are as much as 20 standard deviations from the mean which means the data is not evenly scaled and contains significant outliers. To minimize outliers in the data, I chose to apply a data smoothing algorithm where each point is replaced with the average of the n points on either side of it and itself. We see in Fig. 2.19 (b) with $n = 1$ that the data appears much more evenly scaled, but some points as still as far as 8 standard deviations from the mean. Likewise with $n = 2$ in Fig. 2.19 (c) there are points as far as 6 standard deviations away. Using $n = 3$ results in the most even scaling overall with a the scale being approximately symmetric. While this data smoothing does result in some minor information loss, as long as n remains relatively small the prominent signal topology should be retained. I chose to use $n = 3$ for generating forecasting models and for use with TADA for the remainder of this section. The data smoothing also allows for more flexibility in the input model for TADA because the training data is cleaner and more accurately represents the structure in the signals. Therefore, I generated an LSTM model using the same parameters used for the electron temperature, but this time with only one epoch of optimization. The remainder of the forecasting and DA is handled by TADA. The initial forecast and error plots are shown in Figures 2.20 (a) and 2.21 where we see that the forecast for the unoptimized model is significantly different from the measurements and it does not capture any oscillations from the training data yet. TADA was applied to this model using a larger learning rate of 10^{-4} this time due to only one epoch of optimization occurring prior to using TADA. After 200 TADA steps, the resulting forecast error and forecast for dimensions 80–82 are shown in Figs. 2.20 (b) and

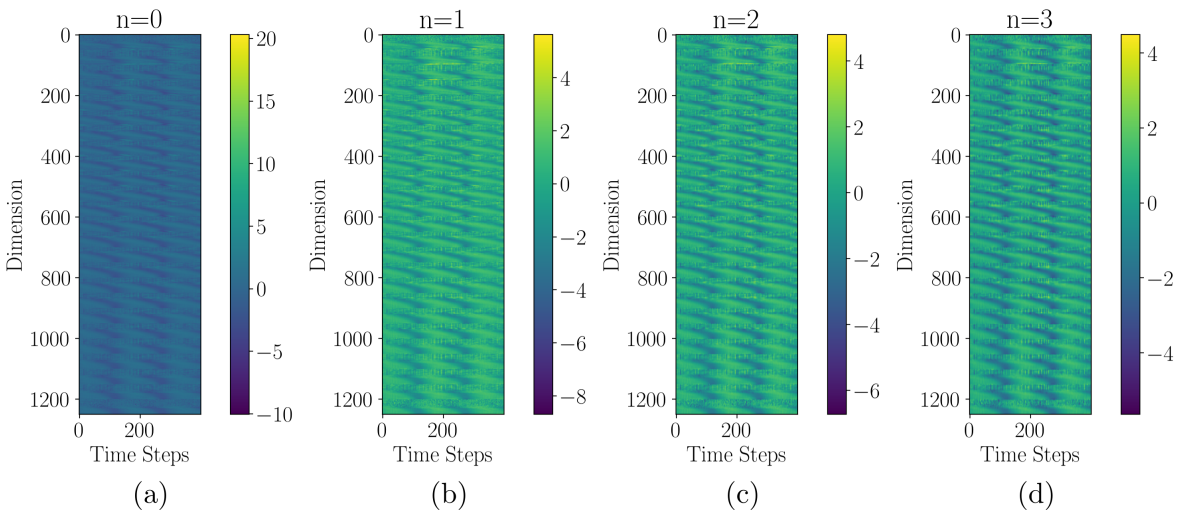
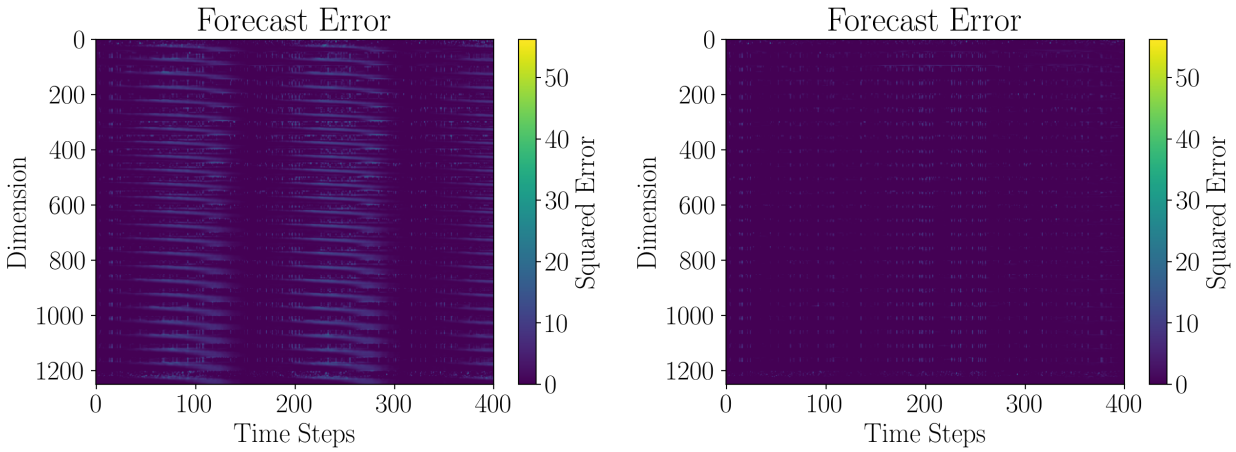


Figure 2.19 Scaled ϕ plots with different levels of smoothing. The plot in (a) contains no smoothing, (b) was generated using $n = 1$ smoothing, (c) $n = 2$ and (d) $n = 3$.



(a) Initial ϕ error

(b) Final ϕ error

Figure 2.20 Error plots for the electric potential before and after 200 TADA steps.

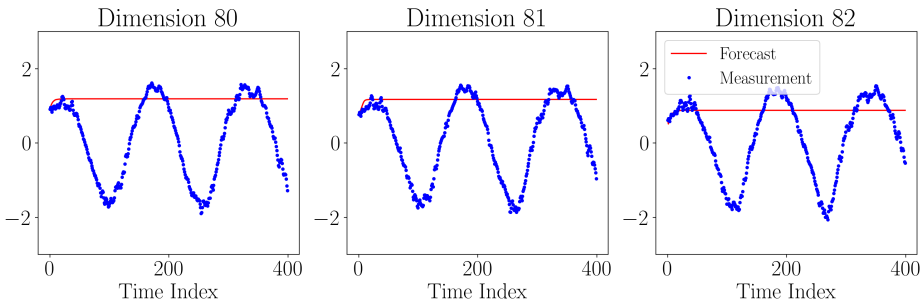


Figure 2.21 Initial ϕ forecast for dimensions 80-82.

2.22. We see that the forecast is significantly improved after applying TADA. The improvement

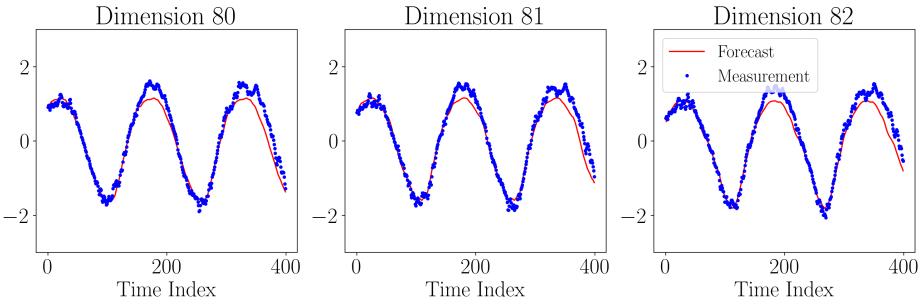


Figure 2.22 ϕ forecast for dimensions 80-82 after 200 TADA steps.

compared to electron temperature is not as good due to there being significantly more noise in the ϕ field, TADA was still able to optimize the model such that it captures oscillations in the signal and provide a much more accurate forecast for this system. Note that the maximum error was initially over 50 and the final maximum error was approximately 27 with other points being significantly smaller. In this section, the TADA algorithm was successfully applied to high-fidelity, high-dimensional simulation data for an SPT-100 Hall Effect Thruster and many generalizations of the original algorithm were presented. I fit the LSTM network forecast function to the generalized equation for TADA and used this forecast function to generate data driven models for HETs. These models were then optimized using the full TADA cost function beginning with a forecast that is not close to the measurements and using persistence optimization to simultaneously learn from the training data and incoming measurements to provide an accurate forecast. Two HET field variables were tested with the algorithm, electron temperature and electric potential. It was found that if the data contains significant outliers after scaling that the forecast functions and TADA will not give accurate predictions. To fix this issue, a data smoothing algorithm was first applied to the data prior to scaling and this allowed TADA to accurately tune the model weights so the forecast accuracy improves.

CHAPTER 3

PARAMETER PATH OPTIMIZATION

This chapter contains another application of optimizing of functions of persistence. In this case, I am interested in navigating the parameter space of a dynamical system to optimize the topological properties of the system's response. This is a challenging problem because the topological properties of a system's response are often difficult to predict, and the parameter space of a dynamical system can be very large and high-dimensional. I show that by optimizing functions of persistence, the parameter space of a dynamical system can be navigated by optimizing specific topological features of the system's response. In Section 3.1, I introduce the relevant background theory for dynamical systems and parameter space optimization. I also present a dictionary of persistence-based cost function terms that are mapped to specific dynamical system behaviors in Section 3.2. Preliminary results are included as motivation for this work in Section 3.3. Section 3.4 shows the theoretical framework for integrating a dynamical system parameter space into the persistence optimization pipeline for gradient descent. Lastly, in Section 3.5, I show results applying these cost function terms to drive systems to a desired behavior through examples using four different dynamical systems.

3.1 Background

For the second project using persistence optimization, I modify the pipeline in Section 1.3.2 to perform optimal parameter space navigation of dynamical systems. The necessary dynamical systems and bifurcation background are given in Section 3.1.1 and 3.1.2.

3.1.1 Dynamical systems

I assume throughout that I have access to sampled realizations $X = [x_1, \cdots, x_N]$ where $x_i \in \mathbb{R}^n$ of a nonlinear dynamical system, and that $\vec{\mu} \in \mathbb{R}^D$ is the vector of system parameters. I aim to locate optimal paths in this space that connect an initial state to a desirable state while avoiding regions of the space that lead to unsafe or undesired dynamics. While for this work I generate X from a model, in theory X could be generated from experimental data, but many experiments need to be conducted to adequately sample the parameter space. To chart paths in the parameter space

for dynamical systems it is important to consider bifurcation theory and timeseries analysis. As these tools are standard in the field, I touch briefly on these topics and direct the interested reader to texts such as [72–75] for further details.

3.1.2 Bifurcations in dynamical systems

Bifurcations can occur in both continuous and discrete time dynamical system and they are characterized by qualitative changes in the response as one or more parameters (called the bifurcation parameters) are varied. Bifurcations often indicate that the system is transitioning from one state to a topologically differing state in the phase space. One visual tool for finding bifurcations is the bifurcation diagram, which shows local extrema of a given system over a varying control parameter while keeping other parameters fixed. As more bifurcation parameters are added, locating bifurcations becomes more difficult as infinitely many paths exist between any two points in the parameter space.

3.1.3 Parameter Space Navigation Using Persistent Homology

When the governing equations for a deterministic dynamical system are available, then there are tools that facilitate tracking the bifurcations as a parameter varies; although, exhaustively tracking all the bifurcations is not a trivial task. One such tool is numerical continuation [76–78], which is a path following approach that tracks the solution branches as system parameters are varied. However, if the governing equations are not available or are too complicated, then sometimes it is possible to track the solutions and the bifurcations of the underlying dynamical system using Control-Based Continuation (CBC) [79–84]. CBC was successfully used in many scientific domains including biochemistry [85], physics [86], mechanics [87], and fluid dynamics [88]. In this setting, numerical continuation is applied to a feedback-controlled physical experiment such that the control becomes non-invasive [84]. Treating the physical system as a numerical model, control-based continuation allows systematic investigations of the bifurcations in the system by treating the control target as a proxy for the state. Nevertheless, existing tools for tracking bifurcation or exploring dynamic changes in state space remain limited to small spaces with most of the time one and at most two bifurcation control parameters. High-dimensional parameter spaces of dynamical

systems have been explored using features of numerical solutions in [89], but this method relies on choosing a feature of the system response which is unintuitive and it also uses random exploration and interpolation of the parameter space with interpolation to extract information from the system.

Therefore, there is a need for an intuitive, data-driven approach to navigating high dimensional dynamical system parameter spaces to guide the system to an acceptable response. I set out to develop a framework with persistence optimization at its core to meet this need and will result in an understanding of the map between parameter space dynamics and topological persistence. I accomplished this goal in three phases. First, I developed a dictionary of cost function terms to promote different persistence diagrams that map to dynamical system responses. For the second phase, I show preliminary derivative-free optimization results moving from chaotic behavior to periodic in the Lorenz system. For phase 3, I perform the optimization using gradient descent with the cost function library from phase 1.

3.2 Cost Function Library

Currently, there is only a basic understanding of the general shape of a persistence diagram for a given dynamic state. For example a periodic response often contains a single 1D persistence pair with a long lifetime. I aim to create a dictionary of persistence diagrams with different traits that will allow the user to impose constraints on the problem. By combining these criterion, a *desired* persistence diagram will be obtained effectively designing an objective function for the optimization problem using topological characteristics of acceptable system behavior. So in this setting, I assume that there is a target persistence diagram that corresponds to desirable criteria for the system response. In this work, I focus on the ability to promote or deter the following behaviors: periodic solutions, fixed point stability, chaotic behavior and allowing for specifying regions of the parameter space that are off limits or unsafe for the system.

3.2.1 Periodic Solutions

To promote periodic solutions, it is intuitive to see that the persistence diagram should contain at least one long lifetime pair far from the diagonal. Therefore, the maximum 1D persistence lifetime feature can be used to control the size of the largest loop. Maximizing the maximum

persistence encourages larger loops in the state space. This feature is computed using,

$$\text{maxPers}_1 = \max_i (\ell_i^1), \tag{3.1}$$

where ℓ_i^1 is the lifetime of the ith 1D loop in the persistence diagram. Another feature that is typically used to quantify the prominence of persistence pairs is the total persistence where instead of taking the maximum lifetime, the sum of all lifetimes is used. However, to promote periodic solutions, I argue that the maximum lifetime is more important than the total lifetime because if all lifetimes are maximized simultaneously this could also promote chaotic behavior. The maximum persistence could also promote chaos so it will need to be combined with other cost function terms such as persistent entropy if there is chaotic behavior in the system.

3.2.2 Fixed Point Stability

The second criterion is to promote fixed point stability. In this case, the persistence diagram should contain loops that are close to the diagonal. This is because fixed points are typically represented by points in the state space that are close in proximity. To quantify this behavior, the maximum persistence can also be used and minimizing this term will result in loops that are low lifetime. Another feature that could be used is the total persistence feature of the 1D persistence diagram to promote all loop lifetimes approaching zero. This feature is computed as,

$$\text{totPers}_1 = \sum_i \ell_i^1. \tag{3.2}$$

The average persistence can also be used which is the total persistence divided by the number of persistence pairs to normalize the feature. In some cases these features may be biased by persistence pairs near the diagonal so in this case maximum persistence should be used or the n longest lifetime persistence pairs to filter out low lifetime features.

3.2.3 Chaos

By definition, chaos in a dynamical system is sensitive dependence of initial conditions or parameters. In other words, changing a parameter or starting point by an infinitesimal amount will yield drastically different system trajectories over time, but the topologies of the trajectories should be similar. A 1D point cloud persistence diagram for a chaotic trajectory typically consists

of many loops that are close and moderately far from the diagonal. To control chaos using persistence diagrams, I suggest using persistent entropy which is the Shannon entropy for a probability distribution of persistence lifetimes [90]. Specifically, persistent entropy is computed as,

$$E = -\sum_i p_i \log_2(p_i), \tag{3.3}$$

where ℓ^1 is the set of 1D persistence lifetimes, $p_i = \ell_i^1 / L$ and $L = \sum_i \ell_i^1$. Persistent entropy gives a measure of order for the distribution of persistence lifetimes so if the lifetimes are more unevenly distributed, persistent entropy is larger and for lifetimes that are more concentrated, E is smaller. Persistent entropy was shown to be stable under small perturbations in [90] and is a Lipschitz function so it can be used with persistence optimization. Note that persistent entropy is biased by the number of persistence pairs in the persistence diagram so to reduce this effect it is often normalized by $\log_2(N)$ where N is the number of pairs in the persistence diagram. This ensures that only the level of disorder is being quantified. While in theory this function can be used for quantifying disorder in persistence diagrams and allow for moving to parameters with a more ordered lifetime distribution, we will see in Section 3.4 that there are complications in computing the gradients of a system trajectory in a chaotic region of the parameter space that need to be mitigated using different optimization techniques. Examples using this term to define cost functions are shown in Section 3.5.

3.2.4 Forbidden Regions

The last cost function term I consider deals with allowing the user to specify regions of the parameter space that are forbidden. In other words, if the parameters enter these regions, a penalty is applied to the cost function. I assume that there is a function $f(\vec{\mu}) : \mathbb{R}^m \to \mathbb{R}$ that is positive for values of $\vec{\mu}$ that are inside of the forbidden region bounded by f and negative for $\vec{\mu}$ outside of this region. Here, $\vec{\mu}$ is a vector of system parameters and m is the number of system parameters or the dimension of the parameter space. A penalty term $\mathcal{L}_p$ can be added to the overall cost function in the form of,

$$\mathcal{L}_p = \exp(a\, f(\vec{\mu})) \tag{3.4}$$

where $a \in \mathbb{R}^+$ is a parameter that is chosen based on tolerances for how close the parameters are allowed to be to the boundary. For example, if a is close to zero, μ will be strongly forced away from f, but for large a the parameters are allowed to get very close to the boundary and once it is crossed a large penalty is applied. a needs to be balanced to ensure that gradients are not too discontinuous. For this work I use a value of $a = 100$ unless otherwise specified. This logic can also be applied to the persistence pairs. For example, if we replace $\vec{\mu}$ with a persistence diagram and define f to penalize persistence pairs in a forbidden region of the birth–death plane, the allowable areas for persistence pairs can also be specified. In all examples shown in Section 3.5, penalty terms are used to regularize the paths by ensuring that the parameters stay in a specified region of the space. In this case, each f is defined as a difference between each parameter in μ and its maximum and minimum allowable value. If the parameter leaves this range the penalty term increases. More examples of applying forbidding regions are shown in Section 3.5.1.

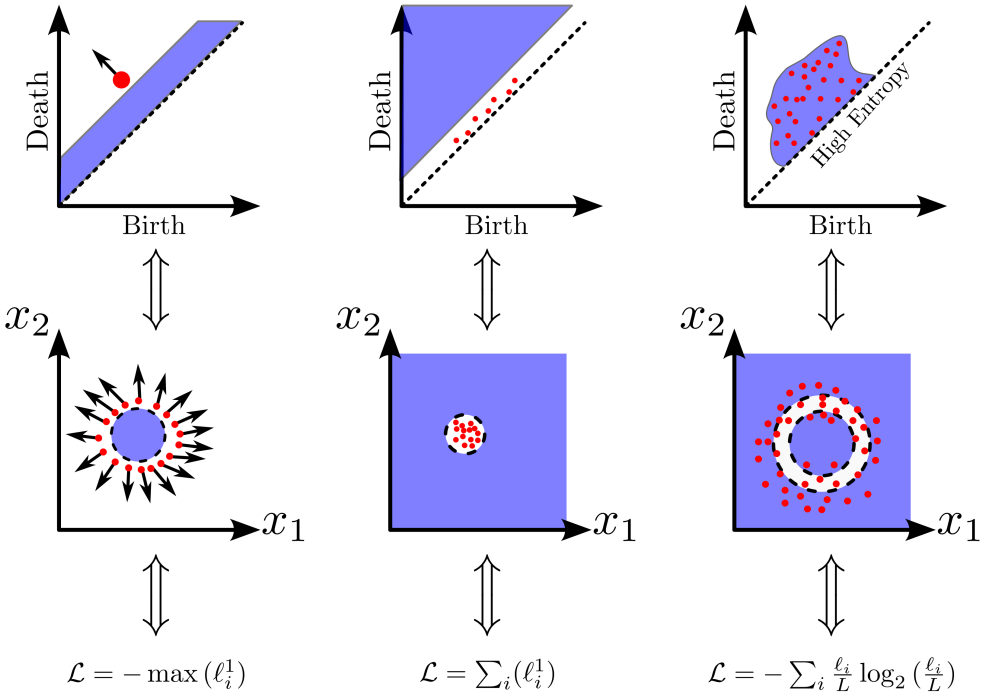


Figure 3.1 Example response criteria mapped into persistence diagrams. (Left) Maximizing maximum persistence to promote a large loop in the state space, (Middle) Limiting persistent loops to be close to the diagonal to encourage fixed point behavior, and (Right) Using high persistent entropy to classify chaotic regions in the persistence diagram.

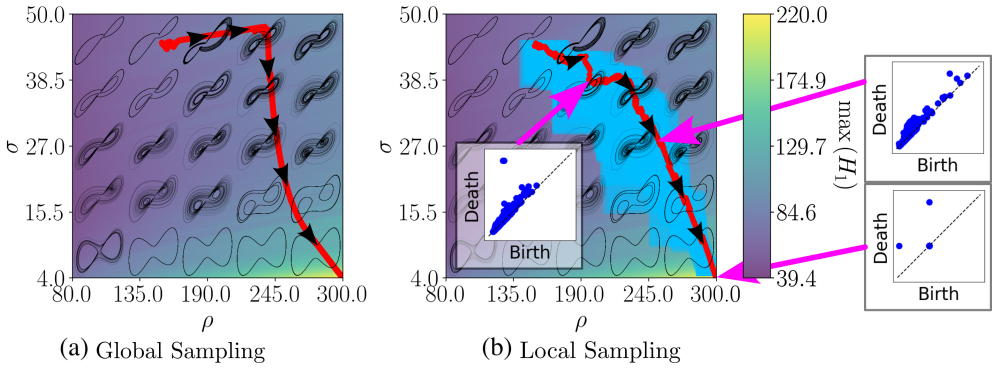
These criteria can be combined to build cost functions that promote desired responses. For example, the user may be searching for parameters that will result in a periodic solution with an amplitude less than a certain value. It is easy to see that this would map into the persistence diagram space as a 1D loop with a limited lifetime or death time minus birth time. In this case the maximum persistence can be used to promote a large loop as shown in Fig. 3.1 (left) and can be combined with a penalty term from Eq. (3.4) to deter lifetimes above a specified value which directly corresponds to the size of the loop or amplitude. A corresponding state space representation of this idea is shown in the middle row and potential cost function terms for achieving these behaviors are shown on the bottom row.

Another desired behavior could be for the system to have fixed point stability. In this case the desired persistence diagram should have all of the loops close to the diagonal as shown in Fig. 3.1 (middle) and the state space plot would have localized points to promote steady state stability. The third case shown in Fig. 3.1 (right) is an avenue for classifying chaotic behavior using the persistence diagram by computing persistent entropy as in [91]. In the state space this could correspond to a safe region being within a small annular region. Together these criteria are specified by the user form the desired characteristics of the target persistence diagram which are used for intuitive loss function engineering for computing the optimal path in the parameter space.

3.3 Preliminary work

Once the desired persistence diagram is identified by defining a loss function to promote desired features, the objective is to move to a point in the parameter space that results in obtaining a response that has a persistence diagram closest to the desired diagram or in other words minimizes the loss function. Before gradient descent is performed with this cost function, I show preliminary results to motivate the need for this approach and show that it is feasible using derivative free optimization methods.

Figure 3.2 shows preliminary results for this project. Here, I considered the response optimal if it had the largest 1D persistence lifetime in the region of the parameter space being searched. This objective was constructed to promote periodic solutions over chaotic by maximizing the maximum



(a) Global Sampling (b) Local Sampling

Figure 3.2 Lorenz system optimal parameter space paths using the global and local updating schemes. Corresponding persistence diagrams are shown at three points to demonstrate the topological differences between dynamic states.

persistence feature. Consider the Lorenz system given by $\dot{x} = \sigma(y-x),\ \dot{y} = x(\rho - z) - y,\ \dot{z} = xy - \beta z$, where σ, ρ and β are system parameters and x, y, and z are the system states. I restrict the parameter space to the plane $\beta = 8/3$ for visualization and search for an optimal two dimensional path in the (ρ, σ) space. I further limit the parameter space by setting $\rho \in [80, 300]$ and $\sigma \in [4, 50]$. The system was then simulated over a 500×500 grid of parameters in the parameter space and the maximum 1D persistence lifetime was plotted as an image along with a subset of the system trajectories in two dimensions as shown in Fig. 3.2. For a given starting point in the parameter space, I aim to navigate this space optimally to reach the point with the largest 1D persistence lifetime. In this region of the parameter space, the optimal point was found to be $(\rho, \sigma) = (300, 4)$ using the simplicial homology global optimization (SHGO) method.

3.3.1 Navigation Schemes

To generate the next point along the path towards the target state in Fig. 3.2, I used conventional global optimization algorithms to find the maximum 1D persistence in a local region near the starting point. A smaller sampling region highlighted in blue in Fig. 3.2b was chosen to obtain a smoother path to the optimal point. I describe these two possible sampling schemes in the following sections.

Global Sampling: The first sampling method works by forming a rectangular region that grows from the starting point in the parameter space and solving for the global optimizer within

that region. Let (x_0, y_0) be the starting point in the 2D parameter space and the global problem domain $\Omega = \{\vec{\mu} = (x,y) \in [x_{min}, x_{max}] \times [y_{min}, y_{max}]\}$. The local search region is then generated as a fraction of the global region by the sequence, $\Omega_k = \{(x,y) \in \frac{1}{N}[(N-k)x_0 + kx_{min}, (N-k)x_0 + kx_{max}] \times \frac{1}{N}[(N-k)y_0 + ky_{min}, (N-k)y_0 + ky_{max}] : k = 1...N\}$ where N is the number of desired path steps. So as the step index k increases, the feasible region grows to fill the entire global region when $k = N$. At each step k, we solve $\mu_k = \arg\max_{\mu \in \Omega_k} f(\vec{\mu})$ to find the direction vector relative to the current point. For this example, $f(\vec{\mu}) = \max(H_1)$ is the maximum 1D persistence lifetime of the system simulation point cloud at a parameter input $\vec{\mu}$. Let $\hat{\mu} = \frac{\mu_k - \mu_{k-1}}{||\mu_k - \mu_{k-1}||}$ be the optimal unit direction from the local optimization problem assuming the optimal point is not identical to the current point. If this is the case, the path remains at the current point. If $\hat{\mu}$ is nonzero, the next point on the path is computed as $(x_k, y_k) = \alpha\hat{\mu}$ where α is the step size. Applying this updating scheme to the Lorenz system with a starting parameter vector of $(\rho, \sigma) = (153, 45)$ with a constant step size of 0.1 and path length of 2500 steps, I obtained the path shown in Fig. 3.2 (a). It is clear that as more steps are taken in the path, the algorithm eventually moves in the direction of the optimal point and approaches it by the final step demonstrating optimal movement in the parameter space to move the system to a periodic response.

Local Sampling: The global sampling path method required data from the full parameter space sampling region for solving the 2500 optimization problems. Simulation data is not always abundantly available so it is important to have an algorithm that also minimizes the search region size for the individual problems. To improve the path generation algorithm, I aim to use sampling regions that are centered around the current point essentially forming a rectangular trust region. The trust region is defined similar to Ω_k in the global sampling approach with two critical modifications. First, the region is based around (x_k, y_k) instead of (x_0, y_0), and second, I multiply the region by a confidence factor $\gamma \in [0,1]$ to allow for the size of the region to depend on the overall confidence in the new direction vector rather than the step size. Together these changes make up the region $\Omega_k = \{(x,y) \in (1-\gamma)[x_{k-1} - x_{min}, x_{max} - x_{k-1}] \times (1-\gamma)[y_{k-1} - y_{min}, y_{max} - y_{k-1}] : k = 1...N\}$. As $\gamma \to 1$, we are more confident in the updated direction so the search region for the next step can

be reduced in size. Conversely, as $\gamma \to 0$, we are less confident in the direction so the search size approaches the full parameter space. To prevent the full parameter space from being used on the first step, γ is initially set close to 1. To update the confidence factor, I use the component-wise standard deviations of the direction vectors of the previous five steps. Because the direction vectors are unit vectors, the largest standard deviation is bounded at one in the case that 50% of the points are at -1 and the other 50% are at 1. For a general system with D-dimensional parameter space $\vec{\mu} = (\mu_1, ..., \mu_D)^\intercal$ the confidence factor can be computed as $\gamma = 1 - \prod_{i=1}^{D} \sigma_i^{(p)}$ where σ_i is the standard deviation of the previous p direction vectors of component i. Performing this updating algorithm on the lorenz system from the same starting point, the path shown in Fig. 3.2 (b) is obtained where the blue region around the path shows the significantly smaller sampling region used by the navigation scheme to generate the path.

We see from these preliminary results that using derivative free optimization works quite well for finding a periodic solution of this system, but it required sampling points in the vicinity of the current path point. Sampling the loss function is very expensive in this case because it requires a full numerical simulation of the system. This motivates the need for generating these paths using gradient descent to minimize the number of loss function samples required for moving through the space. The trade-off here is computing the gradient of the loss function. However, this is much more practical for driving a physical system because it only requires small changes in the parameters to estimate the gradient.

3.4 Gradient Descent Parameter Space Navigation using TDA

In Section 1.3.3, I show a number of examples optimizing the positions of points in a point cloud using cost functions defined from functions of persistence. This pipeline is represented by $\mathcal{M} \xrightarrow{B} \text{Pers} \xrightarrow{V} \mathbb{R}$ where $\mathcal{M}$ is the point cloud and B maps the point cloud to its persistence diagram. V is the function of persistence that maps to a real number. Using the chain rule, the map $V \circ B$ is differentiated to perform gradient descent and perturb $\mathcal{M}$ to minimize $V \circ B$. In this chapter, the main goal is to augment this pipeline with the parameter space of a dynamical system. Specifically, the map becomes $\mathcal{D} \xrightarrow{B'} \mathcal{M} \xrightarrow{B} \text{Pers} \xrightarrow{V} \mathbb{R}$, where $\mathcal{D}$ is the parameter

space of a dynamical system and B' is the map from the parameter space to the state space point cloud. The rest of the pipeline remains the same because $\mathcal{M}$ is still a point cloud, but its movement is governed by the parameter space and dynamics of the system. To have differentiability in the original pipeline from [5], the authors placed restrictions on the positions of point to ensure the derivative exists. Specifically the point cloud must be in general position. Similar restrictions need to be considered for B' before this augmented pipelined can be used. B' is essentially the numerical integrator for the system that generates the state space of the system. Differentiation of ODE solvers has been enabled using the *adjoint sensitivity method* [92] where the gradient of the loss function is computed for each state of the system by solving another ODE whose solution gives the gradient of each state of the system trajectory. This method has been implemented in many different solvers in the torchdiffeq python library with pytorch compatibility [92]. Typically this method is used for neural ODEs which are a continuous analog of traditional neural networks, but for this work I only need the ability to compute the gradient of B'. The method works by assuming we have a dynamical system $\dot{x} = f(x(t), t, \mu)$ with some loss function that depends on the states of the system $x(t)$ given by,

$$L(x(t)) = L\left(x(t_0) + \int_{t_0}^{t_f} f(x(t), t, \mu)dt\right). \tag{3.5}$$

The goal is to obtain the gradient of L with respect to the system parameters μ. In [92], the authors define an adjoint state $a(t) = \partial L/\partial x(t)$ and show a different ODE that governs its dynamics. From [92], the gradient of the loss function with respect to μ is then given by the integral,

$$\frac{\partial L}{\partial \mu} = -\int_{t_0}^{t_f} a(t)^T \frac{\partial f(x(t), t, \mu)}{\partial \mu} dt, \tag{3.6}$$

where $a(t)$ is obtained by solving,

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial f(x(t), t, \mu)}{\partial x}. \tag{3.7}$$

We see that ultimately, the gradient of the loss with respect to system parameters depends on how much the states $x(t)$ change with respect to changes in the system parameters so if a slight change

in parameters yields drastically different system states, $\partial L/\partial \mu$ can be very large and cause complications in the optimization process. However, this method shows that it is possible to compute the gradient of the map B' in our pipeline allowing for the full inverse problem to be solved as shown in Fig. 3.3 where a 3-dimensional parameter space is shown in Fig. 3.3 (a) with starting point in red and to move to the next path point, a step is taken toward a minimizer of the loss function in Fig. 3.3 (d) and the step is propagated back using gradient descent through the persistence diagram (Fig. 3.3 (c)) and state space (Fig. 3.3 (b)) to obtain a direction in the parameter space for the next point on the path. This enables a gradient descent approach to moving through the parameter space using the full persistence diagrams to move to a set of parameters where the response meets the criteria defined from Section 3.2. As a result, the gradient of the full map $V \circ B \circ B'$ is computed to move through the parameter space leveraging the differentiability of persistence diagrams [4–6] and the adjoint sensitivity method from [92].

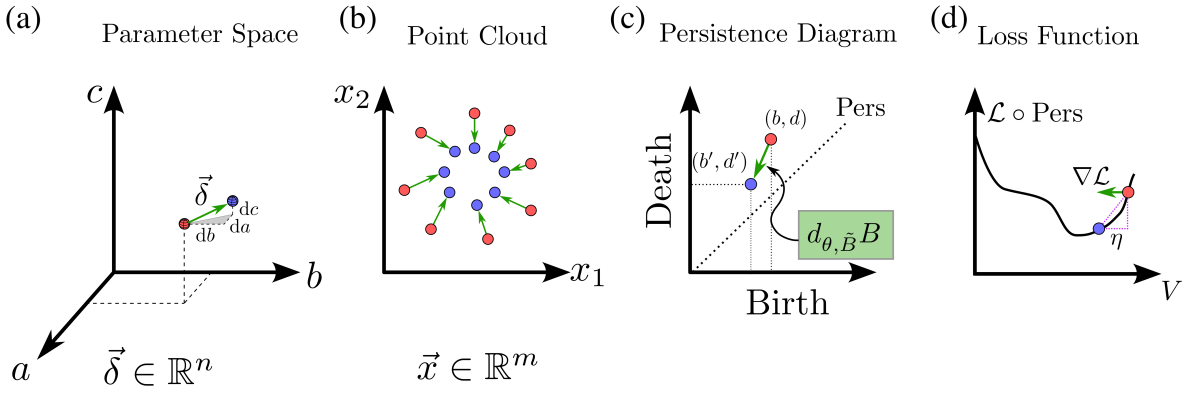


Figure 3.3 Diagram demonstrating the map from the parameter space to the loss function as solving the inverse problem from taking a step against the gradient of the loss function to reach a new set of parameters in the parameter space propagated through the persistence diagram and the state space point cloud.

3.5 Numerical Validation

In this section, I include numerical studies using four dynamical systems to demonstrate the capabilities and limitations of this method. Note that all results utilize the Adam optimizer for performing gradient descent to leverage the momentum advantages for avoiding local minima in the loss functions.

3.5.1 Goodwin System

The first system I studied is a biological oscillator called the Goodwin model which is a negative feedback control process model developed by Brian Goodwin [93]. This model has been used to model many different biological processes such as modeling circadian rhythm clocks [94] and has also been used to model genetic processes [95]. Essentially, the model describes the rates of change and interaction between a gene mRNA strand, protein and another molecule that inhibits the production of the protein [95]. The dynamics are described by the nonlinear differential equations,

$$\begin{aligned} \frac{dx}{dt} &= \frac{1}{1+z^m} - \alpha_1 x, \\ \frac{dy}{dt} &= x - \alpha_2 y, \\ \frac{dz}{dt} &= y - \alpha_3 z, \end{aligned} \tag{3.8}$$

where x, y, and z represent the concentrations of the mRNA, protein, and inhibitor, respectively [96]. The parameters α_1, α_2, and α_3 are the decay rates for the mRNA, protein and inhibitor respectively [95, 96]. Note this variant of the Goodwin model is from [96], but other variants also exist with more parameters. The hill coefficient m describes how quickly z inhibits x and it has been shown that for $m > 8$ there can be limit cycle oscillations in the system [95]. Due to the simplicity of this system, I chose to study it as the first example performing optimal parameter space navigation using persistent homology. First, I decided to reduce the dimensionality of the parameter space so the results could be visualized. From [96], I set $\alpha_3 = 0.2$ and from [95] I used $m = 10$. I then allowed α_1 and α_2 to vary between 0 and 1 and plotted the maximum persistence over this range. The system was simulated using an initial condition of $(x, y, z) = (0.1, 1.1, 3.0)$ and was integrated over a time span from 0 to 200 time units sampling every 0.04 time units and taking the last 500 points as the steady state response. The maximum persistence is plotted in Fig. 3.4. We see that for these decay rates between approximately 0.1 and 0.6 the system exhibits limit cycle oscillations and for all other points in this region the response approaches a fixed point. The first goal was to start in the top right corner of this parameter space and search for the limit cycle using persistence optimization. I set the cost function to the opposite of the maximum persistence to

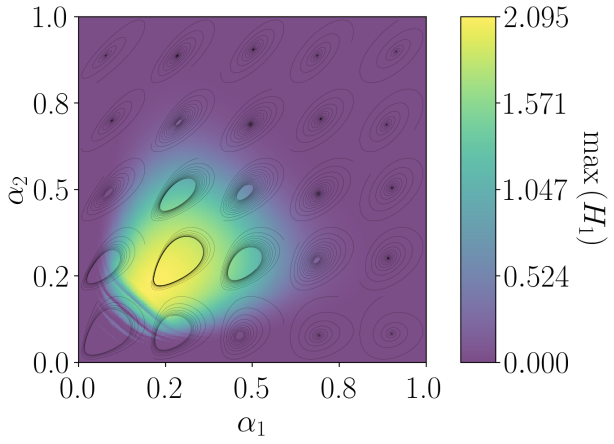


Figure 3.4 Maximum persistence of the Goodwin system over a range of α_1 and α_2 values.

promote large loops in the persistence diagram. Appropriate regularization was also used from Section 3.2.4 to ensure the parameters remained in the specified region. The decay rates were started at 0.8 and 0.9 respectively and the learning rate was set to 0.01. After 275 epochs of optimization the resulting parameter space path is shown in Fig. 3.5 along with the final trajectory and the corresponding 1D persistence diagram. We see that the path starts in a region with fixed

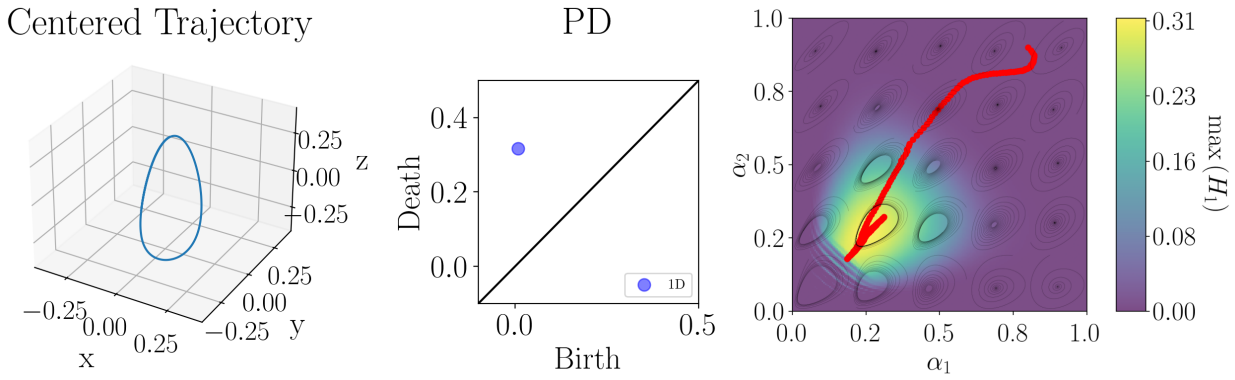


Figure 3.5 Optimal parameter space paths for the Goodwin system using gradient descent. The paths start from the top right corner and move towards the region with limit cycle oscillations.

point stability and optimally traverses the parameter space to find the limit cycle and converges in that region. The result in Fig. 3.5 did not use any learning rate decay so the path eventually hit the bottom left corner of the limit cycle region and bounced back. This can be avoided by introducing learning rate decay at 1% per epoch and the results are shown in Fig. 3.6. We see that once the path

reaches the limit cycle it slows down and does not reflect off of the bottom left of the limit cycle region anymore. Next, I chose to further leverage the technique from Section 3.2.4 by limiting

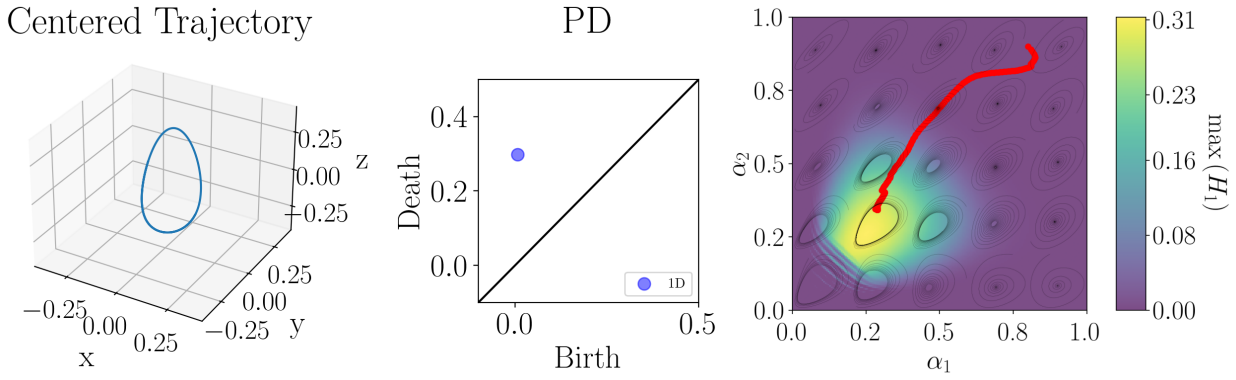


Figure 3.6 Optimal parameter space paths for the Goodwin system using gradient descent with learning rate decay of 1% per epoch. The paths start from the top right corner and move towards the region with limit cycle oscillations.

certain behaviors. Specifically, I defined a circle in the parameter space centered at $(0.5, 0.7)$ with a radius of 0.1. So using Eq. (3.4), I set $f(\vec{\mu}) = (\alpha_1 - 0.5)^2 + (\alpha_2 - 0.7)^2 - 0.1^2$ with $a = -1000$. This cost function term essentially creates a cylindrical wall in the loss function space to prevent the parameters from entering this region. For this example, I also decided to use Eq. (3.4) to limit the maximum allowable lifetime for the persistence diagram to 0.2. This was defined by setting $f(\vec{\mu}) = (maxPers - 0.2)$ where in this case $\vec{\mu}$ corresponds to the 1D persistence diagram. The optimization was performed using the same conditions as Fig. 3.5, but with the added forbidden regions shown in red. The results are shown in Fig. 3.7 where we see that the path successfully avoided the circle by moving around it and the persistence lifetime remained below 0.2 as specified by the loss function terms.

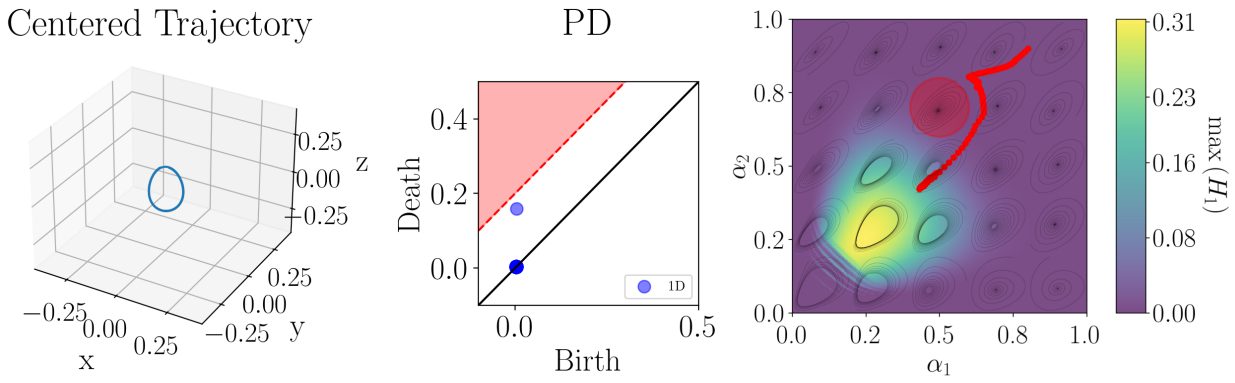


Figure 3.7 Optimal parameter space paths for the Goodwin system using gradient descent with a circle defined in the parameter space that is off limits and the maximum persistence lifetime is limited to 0.2 indicated by the red regions. The paths start from the top right corner and move towards the region with limit cycle oscillations.

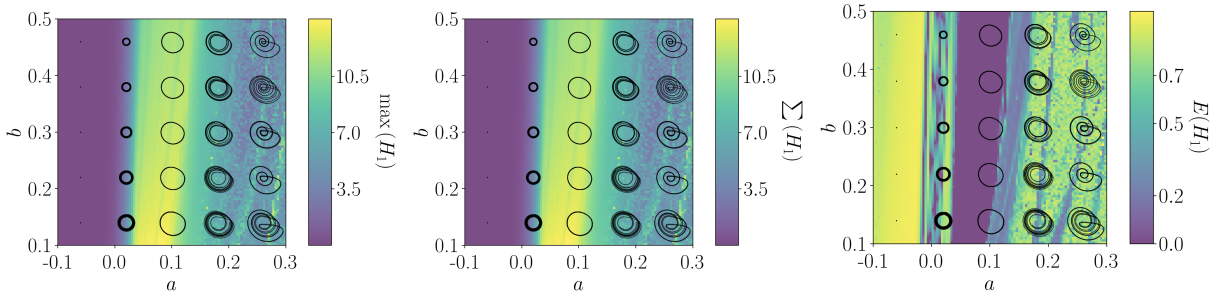
3.5.2 Rössler System

The second system I studied is the Rössler system, which is a well understood chaotic system [97] described by the following set of differential equations:

$$\begin{aligned} \dot{x} &= -y - z, \\ \dot{y} &= x + ay, \\ \dot{z} &= b + z(x - c), \end{aligned} \tag{3.9}$$

where a, b, and c are system parameters. For this analysis, I chose to vary a and b while keeping c fixed at 5.7. The system was simulated using an initial condition of $(x, y, z) = (-0.4, 0.6, 1)$ and integrated over a time span from 0 to 200 time units, sampling every 0.04 time units and taking the last 500 points as the steady state response. Chaotic systems present a difficulty in that the trajectories deviate exponentially for an infinitesimal change in parameters. In theory this should be avoided because the overall topology of the trajectory should remain similar. However, since persistence pairs are being differentiated and not the overall shape or distribution of the pairs, the gradients explode and inconsistently vary by multiple orders of magnitude in chaotic regions of the parameter space. In practice, this issue is commonly alleviated with gradient clipping where the norm of the gradient is saturated at a specified magnitude and this has been shown to greatly improve training and exploding gradient issues in machine learning [98]. The maximum

persistence, total persistence and normalized persistent entropy are plotted over a range of a and b values to identify regions of chaotic and periodic behavior as shown in Fig. 3.8. We see that for larger a values the system appears to be chaotic and as a decreases it moves to periodic and fixed point responses. In the chaotic region of the parameter space, all three features appear to vary significantly suggesting that it will be difficult to find an optimal path in that region. This is further complicated by the entropy being high in the fixed point region. Even though the entropy was normalized, it is still at a maximum value in this region due to the significant number of low-lifetime persistence features. Intuitively, the entropy in this region is small because the loops are so small but in order for this to be detected with persistence it requires very long simulation time and makes computing the gradient using this pipeline computationally expensive. Nonetheless, I generated different results for this system to visualize and attempt to find optimal paths in this parameter space.



(a) Maximum Persistence (b) Total Persistence (c) Persistent Entropy

Figure 3.8 Rössler system persistence features plotted over a range of a and b values.

3.5.2.1 Chaos → Periodic

For the first example, I aimed to move from the chaotic region of the parameter space to find a periodic solution using the persistent entropy and maximum persistence features. Specifically, I set the loss function to minimize persistent entropy and maximize maximum persistence ($\mathcal{L} = E - \text{maxPers}_1$). This leads to another complication from solving this multi-objective optimization problem. We see from Fig. 3.8 that the maximum persistence and entropy are on different scales so optimizing this loss function will lead to an incorrect solution because it is not balanced. A common approach for mitigating this issue is to introduce scaling for each feature in the loss

function as $\mathcal{L} = \sum_i \lambda_i \mathcal{L}_i$ where each λ_i is a scale factor applied to the loss for each objective [99]. For this work, I divide each persistence feature by its current value (detached from the gradient with pytorch) to ensure that each loss is on the same scale before being differentiated. This technique is presented in [100]. With this method, the value of each loss term will always be one, but the terms are all balanced and the gradients still vary in magnitude. I started the path at $a = 0.2$, $b = 0.2$ and used a learning rate of 0.01 with a gradient norm clip of 1.0. Because of the learning rate clip, it is critical to apply learning rate decay to make sure the solution converges so a decay of 1% per epoch was applied. The initial and final results are shown in Figs. 3.9 and 3.10 where we see that the path successfully exited the chaotic region and approached a periodic solution, but due to the learning rate decay the optimizer does not explore the parameter space enough and it settles on a more complicated periodic solution with two prominent loops in the persistence diagram.

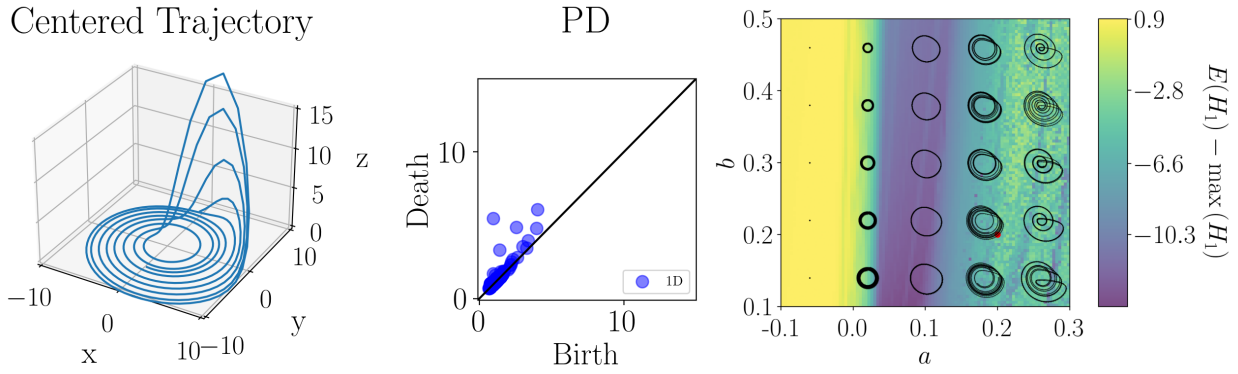


Figure 3.9 Initial Rössler trajectory starting at $a = 0.2$ and $b = 0.2$.

For the next test, I used an identical setup, but change the learning rate decay rate to 0.999 to allow for more exploration and the resulting path is shown in Figs. 3.11. We see that the path converged on a simpler periodic solution in this case, but required many more steps due to the slower learning rate decay. Interestingly, the path seemed to oscillate in the periodic region of the parameter space without entering the fixed point region or chaotic region again. Due to starting in the chaotic region the resulting path is also drastically different from the other learning rate decay which demonstrates the difficulty of optimizing when responses are chaotic.

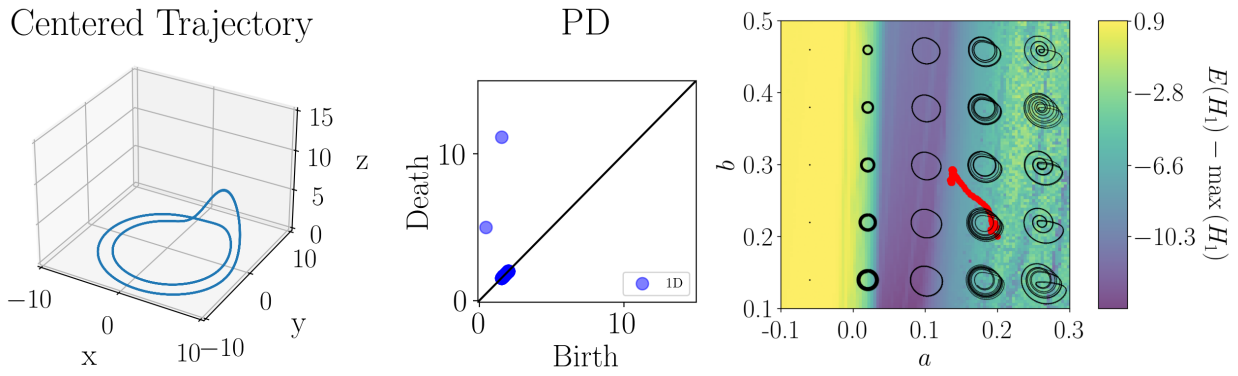


Figure 3.10 Final Rössler trajectory and parameter path after 225 epochs minimizing the persistent entropy and maximizing the maximum persistence using a learning rate decay of 0.99.

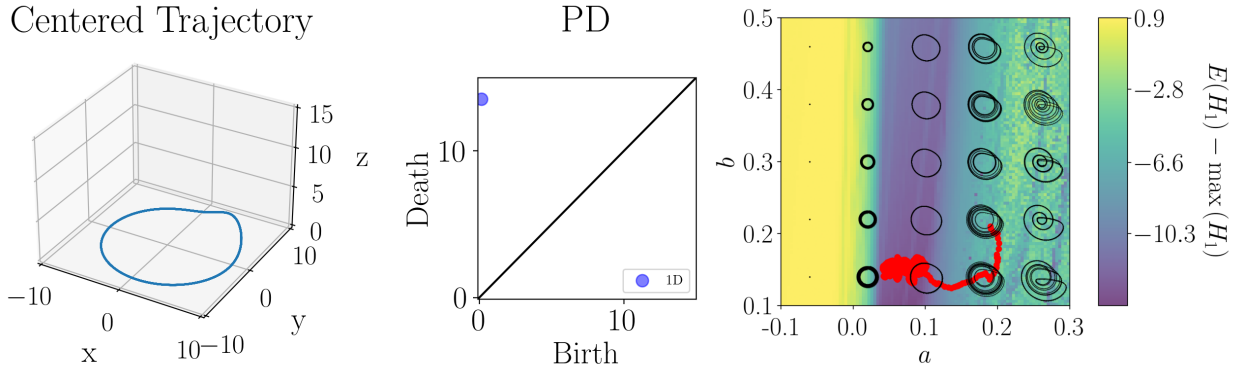


Figure 3.11 Final Rössler trajectory and parameter path after 450 epochs minimizing the persistent entropy and maximizing the maximum persistence using a learning rate decay of 0.999.

3.5.2.2 Periodic → Fixed Point

Next I show examples attempting to move from a periodic response to fixed point response using the features from Section 3.2. Initially, the goal was to minimize the total persistence to reach the desired response. I set the cost function to be the total persistence and started the path at $a = 0.1$ and $b = 0.2$. The initial trajectory and persistence diagram are shown in Fig. 3.12. Using the total persistence cost function and a learning rate of 0.01 with decay rate of 0.99, the path in Fig. 3.13 was obtained after 142 epochs. We see that the path correctly moves toward the fixed point region but continues to hit the lower bound on a. The path oscillates on this line before converging on parameters slightly outside of the region. This result demonstrates a limitation of this method in the way the Adam optimizer works. Because the optimizer had momentum moving

toward the wall the path was able to exit the region. This is why when defining forbidden regions using the method in Section 3.2.4, it is important to give some buffer space as it is possible for the path to go beyond the boundary in some cases.

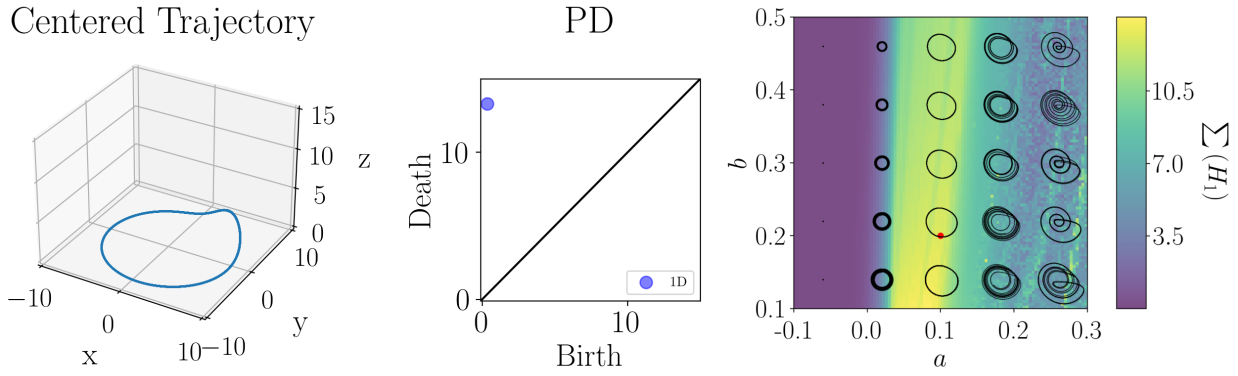


Figure 3.12 Initial Rössler trajectory at $a = 0.1$ and $b = 0.2$.

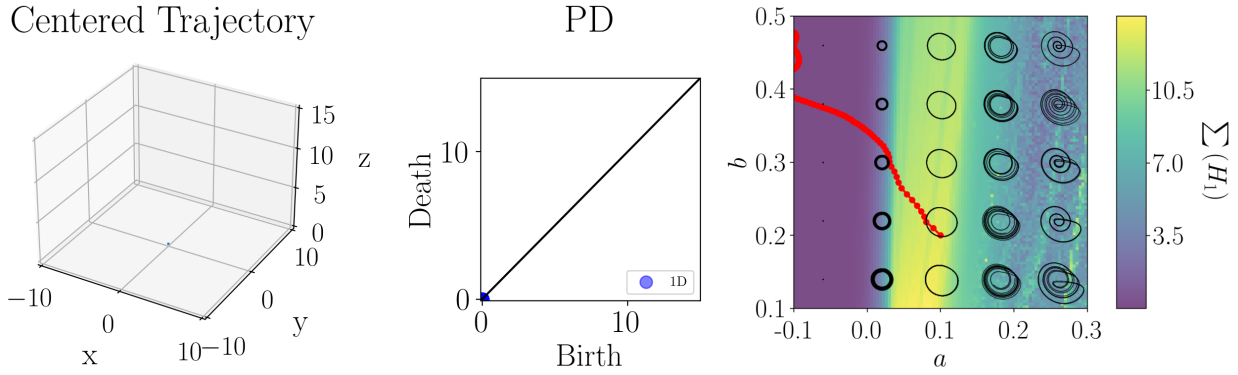


Figure 3.13 Final Rössler trajectory and parameter path after 142 epochs minimizing the total persistence using a learning rate decay of 0.99.

Next, I ran the same test but reduced the learning rate decay to 0.95 and the path in Fig. 3.14 was obtained after 106 epochs. We see that the total persistence loss landscape provides some resistance to the fixed point region because the total persistence slightly increases as more low lifetime persistence pairs appear. The path was never able to enter the fixed point region in this example. This demonstrates a limitation of the total persistence feature that is mentioned in Section 3.2 where the total persistence can be biased by the number of pairs in the persistence diagram. To attempt to mitigate this issue, I chose to run the same test using maximum persistence and the

results are shown in Fig. 3.15 where we see the path converges on the parameters $a = -0.0664$ and $b = 0.3136$ after 147 epochs.

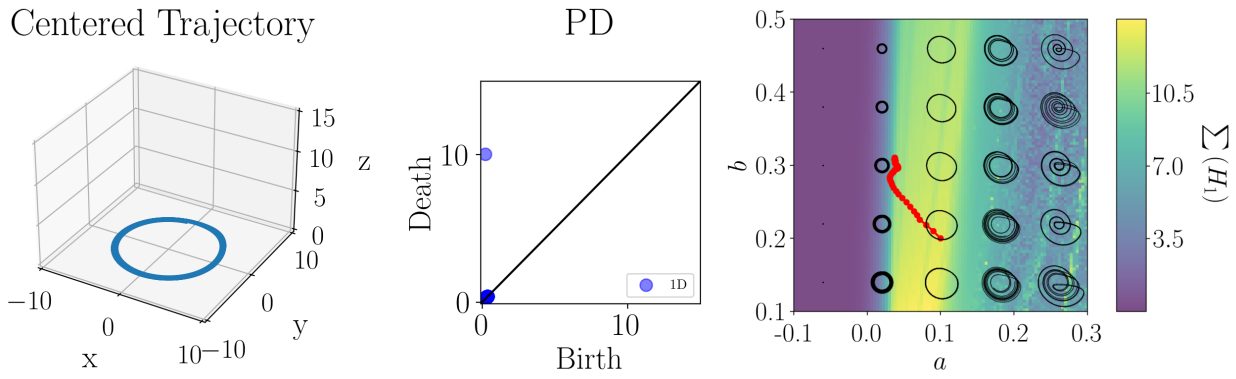


Figure 3.14 Final Rössler trajectory and parameter path after 106 epochs minimizing the total persistence using a learning rate decay of 0.95.

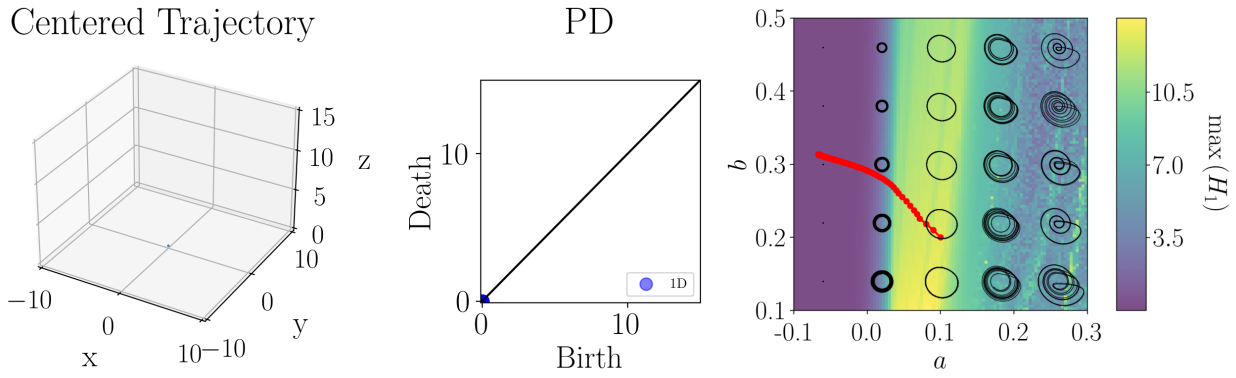


Figure 3.15 Final Rössler trajectory and parameter path after 147 epochs minimizing the maximum persistence using a learning rate decay of 0.95.

3.5.2.3 Chaos → Fixed Point

For the final Rössler system test, I aimed to move from the chaotic region to a fixed point response. I used the same setup as the previous tests by starting the path at $a = 0.2$, $b = 0.2$ and set the learning rate to 0.01 with a decay rate of 0.999. Because the persistent entropy reaches a minimum in the periodic region, I chose to only use total persistence in this case for the loss function. The initial trajectory is shown in Fig. 3.9 and after 165 epochs of optimization the path is shown in Fig. 3.16. The path in this case correctly exited the chaotic region of the parameter space,

but was not able to enter the fixed point region likely due to momentum issues again. Once the path entered the periodic region, interestingly, it moved in the vertical b direction and it is believed that this is also a consequence of the optimizers momentum. The gradient in the a direction is much larger than the gradient in the b direction, so the Adam optimizer conservatively moves vertically and due to the small learning rate it is not able to enter the fixed point region. To fix this issue, I increased the learning rate to 0.02 and ran the same test. The resulting path is shown in Fig. 3.17. The resulting path successfully reached a fixed point solution from chaos with the larger learning rate, but also required regularization as the optimizer tried to decrease a beyond the lower limit.

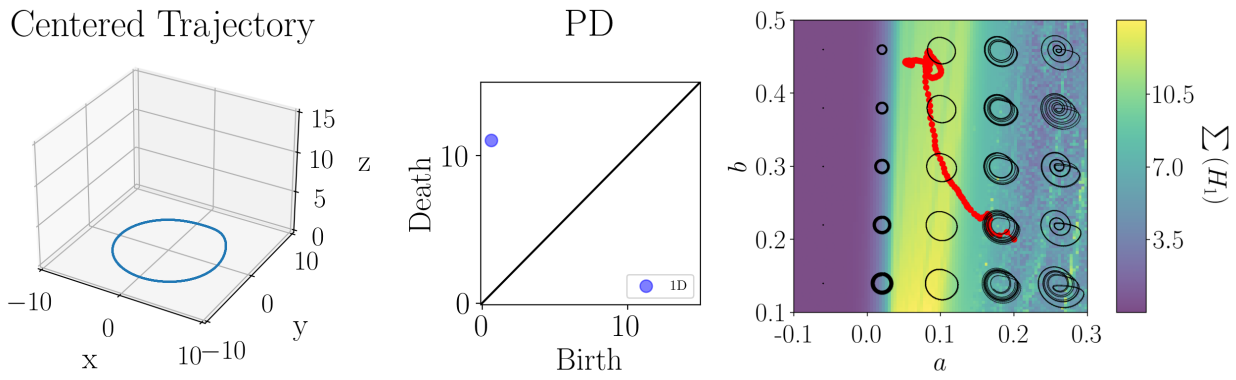


Figure 3.16 Final Rössler trajectory and parameter path after 165 epochs minimizing the maximum persistence using a learning rate of 0.01 and decay rate of 0.999.

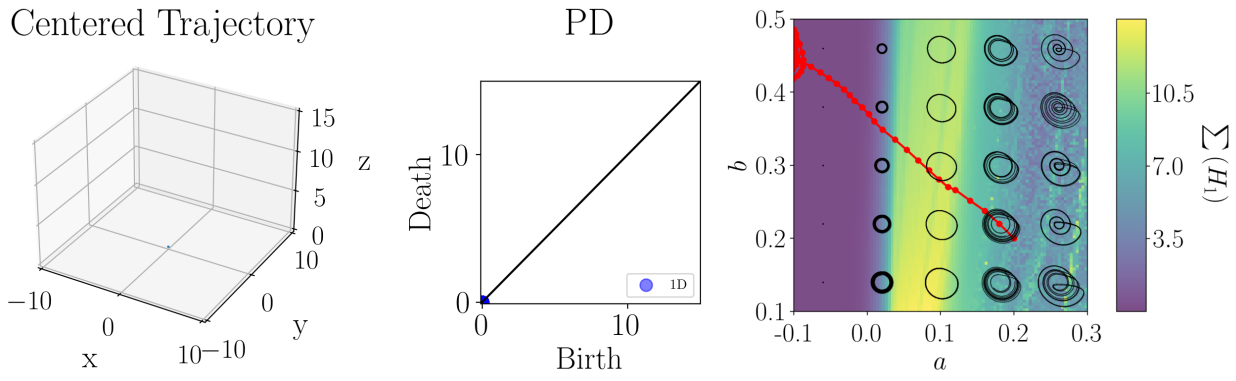


Figure 3.17 Final Rössler trajectory and parameter path after 200 epochs minimizing the maximum persistence using a learning rate of 0.02 and decay rate of 0.999.

3.5.3 Magnetic Pendulum

Next, I chose to study the base excited magnetic pendulum system shown in [101]. This system consists of a normal base excited pendulum system with a magnet on the end of the pendulum arm and another magnet on the base. This creates a magnetic interaction between the pendulum and base that leads to complex behavior. The equation of motion for the pendulum is:

$$(Mr_{\text{cm}}^2 + I_{\text{cm}})\ddot{\theta} + Mgr_{\text{cm}}\sin(\theta) = -\tau_v - \tau_m - Mr_{\text{cm}}\ddot{x}_{\text{base}}\cos\theta. \tag{3.10}$$

where θ is the pendulum angle measured from vertical, $\ddot{x}_{\text{base}} = -A\omega^2\sin(\omega t)$ is the base acceleration, $M = 0.1038$ kg is the total mass, $l = 0.208$ m is the length of the pendulum, $g = 9.81$ m/s^2, $r_{\text{cm}} = 0.18775$ m is the distance to the center of mass from the hinge, $I_{\text{cm}} = 1.919 \times 10^{-5}$ kg·m^2 is the mass moment of inertia, $\mu_v = 0.003$ is the viscous damping coefficient, $m = 1.2$ A·m^2 is the magnetic dipole moment, $\mu_0 = 1.257 \times 10^{-6}$ N/A^2 is the permittivity of free space, and $d = 0.032$ m is the minimum distance between the magnets. In [101], the authors measure system parameters and I use the same parameters in this work. τ_m is the magnetic interaction torque given by,

$$\tau_m = l(F_r\cos(\phi - \theta) - F_\phi\sin(\phi - \theta)), \tag{3.11}$$

where,

$$F_r = \frac{3\mu_0 m^2}{4\pi r^4}\left(2\cos(\phi - a)\cos(\phi - b) - \sin(\phi - a)\sin(\phi - b)\right), \tag{3.12}$$

$$F_\phi = \frac{3\mu_0 m^2}{4\pi r^4}\sin(2\phi - a - b), \tag{3.13}$$

are the radial and tangential magnetic interaction forces, and r and ϕ are the polar coordinate locations of the pendulum measured from the base magnet and they are computed with,

$$r = \sqrt{l^2 + (d+l)^2 - 2l(l+d)\cos\theta}, \tag{3.14}$$

$$\phi = \frac{\pi}{2} - \arcsin\left(\frac{l}{r}\sin\theta\right), \tag{3.15}$$

with $a = \frac{3\pi}{2}$, $b = \frac{\pi}{2} - \theta$. Lastly, the viscous damping torque is,

$$\tau_v = \mu_v\dot{\theta}. \tag{3.16}$$

For this analysis, I chose to vary the base excitation amplitude (A) and frequency (ω) for persistence optimization. I plotted the maximum persistence over a range of amplitudes and frequencies shown in Fig. 3.18. We see that for a range of larger amplitude and lower frequency, the response is periodic and for low amplitude and frequency the system approaches a fixed point. This intuitively makes sense for the system. The goal is to reach a fixed point using persistence optimization now. I set the cost function to minimize the maximum persistence and started the path at $A = 4$ cm and $\omega = 7.5$ rad/s. The system was simulated for 100 seconds sampling every 0.03 seconds and taking the last 500 points for computing persistence. The initial response and persistence diagram are shown in Fig. 3.19. In this case, I wanted to explore a larger region of the parameter space, so I set the learning rate to 0.1. After 85 optimization steps, the path is shown in Fig. 3.20. We see that the path reaches a set of parameters that result in fixed point stability for the system and the regularization term caused the path to reflect off of the lower limit on A. Intuitively, we know that taking A to be as small as possible is the best way to minimize oscillations, but in this case it settles on a set of somewhat nontrivial parameters at $A = 0.57$ cm and $\omega = 6.55$ rad/s.

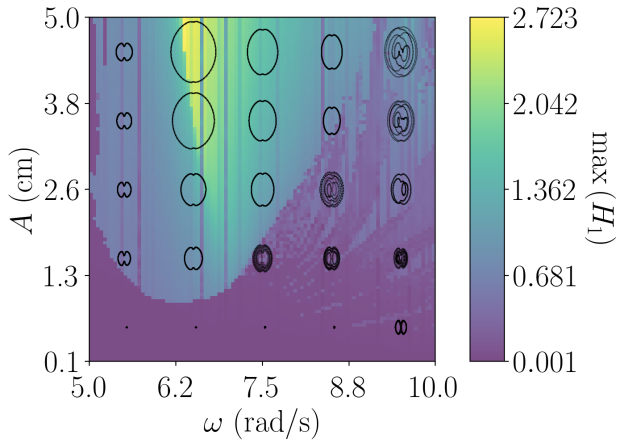


Figure 3.18 Maximum persistence plotted over a range of base excitation amplitude and frequency for the magnetic pendulum system.

3.5.4 Lorenz System

For the final example, I chose to study the Lorenz system to show that this method aligns with the preliminary results from Section 3.3. For this analysis, I chose to vary σ and ρ while

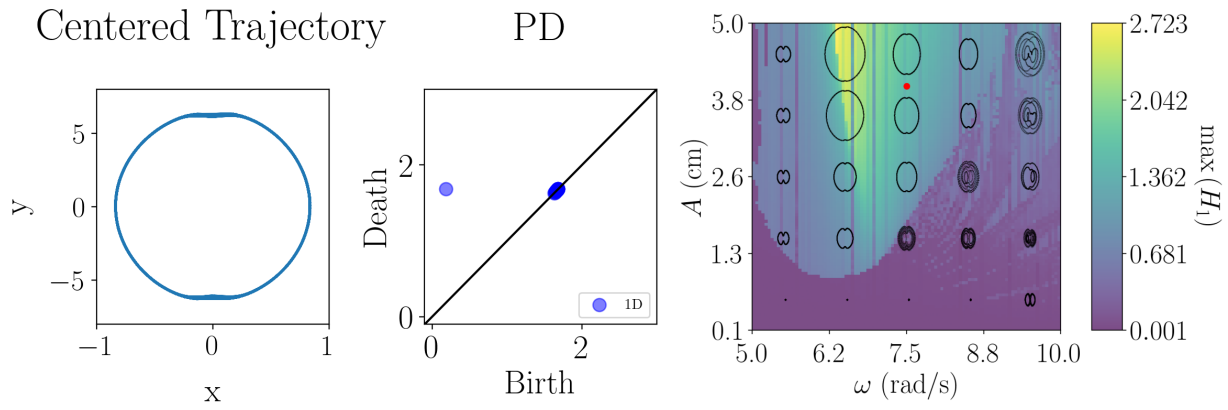


Figure 3.19 Initial response and persistence diagram for the magnetic pendulum system.

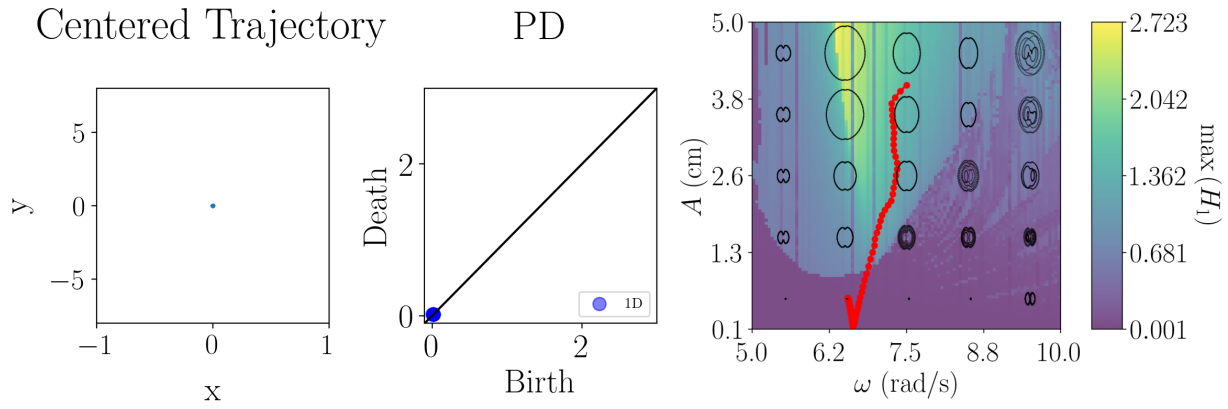
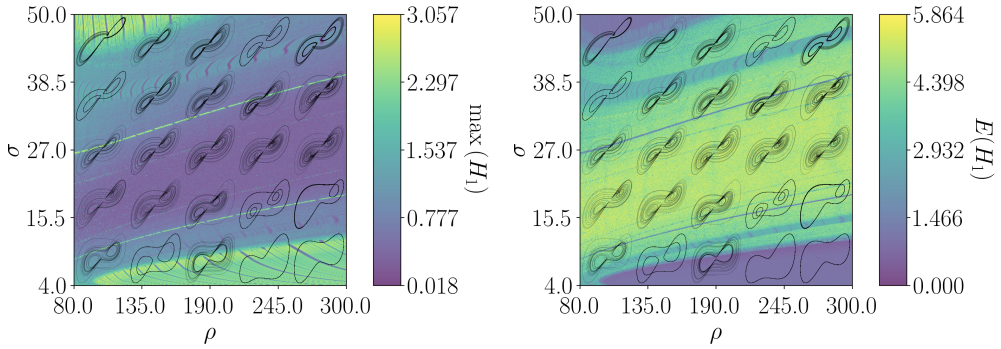


Figure 3.20 Final response and persistence diagram for the magnetic pendulum system with the parameter space path minimizing maximum persistence.

keeping β fixed at the typical value of $8/3$ for visualization purposes. The system was simulated using an initial condition of $(x,y,z) = (1,1,1)$ and integrated over a time span from 0 to 10 time units, sampling every 0.01 time units and taking the last 500 points as the steady state response. The maximum persistence and persistent entropy were plotted over a range of σ and ρ values to identify regions of chaotic and periodic behavior as shown in Fig. 3.21. We see that periodic solutions appear to be most prominent for low values of σ and as this parameter increases, a chaotic region forms. The goal was to start the path in the chaotic region, and use these persistence features to navigate to the periodic region. However, this example is much more challenging than previous examples due to the parameter space being significantly larger. In all cases, the loss function was defined to be the difference between entropy and maximum persistence to minimize entropy and

maximize maximum persistence and the loss function scaling was applied to ensure these terms were on the same scale. Gradient clipping to a norm of one was also applied in this example due to the exploding gradients in chaotic regions of the parameter space. I started by initializing the path at $\rho = 190$ and $\sigma = 20$. The initial trajectory is shown in Fig. 3.22. Note that for the plotting purposes, the trajectories were normalized using the mean and standard deviation, but for the persistence optimization computations the unmodified state space was used.



(a) Maximum Persistence (b) Persistent Entropy

Figure 3.21 Lorenz system persistence features plotted over a range of ρ and σ values.

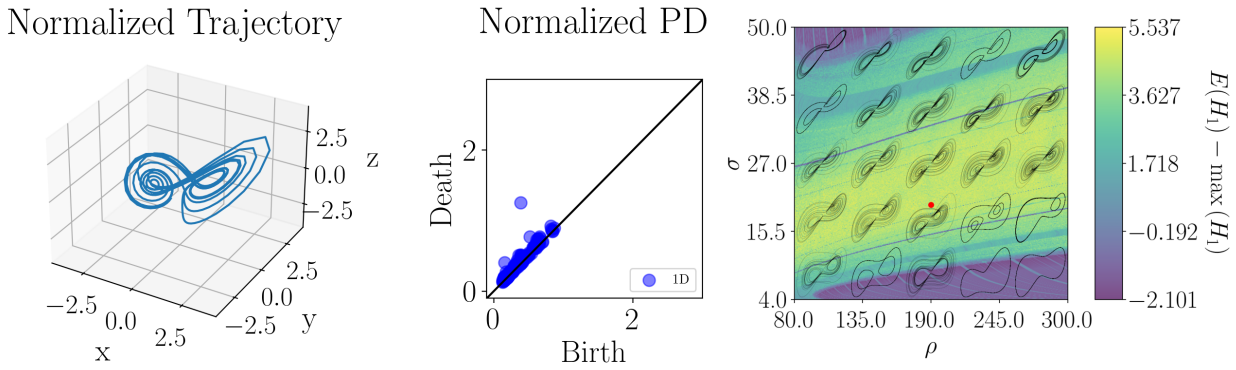


Figure 3.22 Initial Lorenz trajectory.

For the first test, I set the learning rate to 0.1 for the optimization. While this learning rate is too small to reach the periodic region in a reasonable amount of time, it is important to always start small with the learning rate to promote shorter paths. The resulting path without learning rate decay after 385 epochs is shown in Fig. 3.23. We see that the path is very short in this case due to the small learning rate and it was never able to exit the chaotic region. In order to explore a larger region

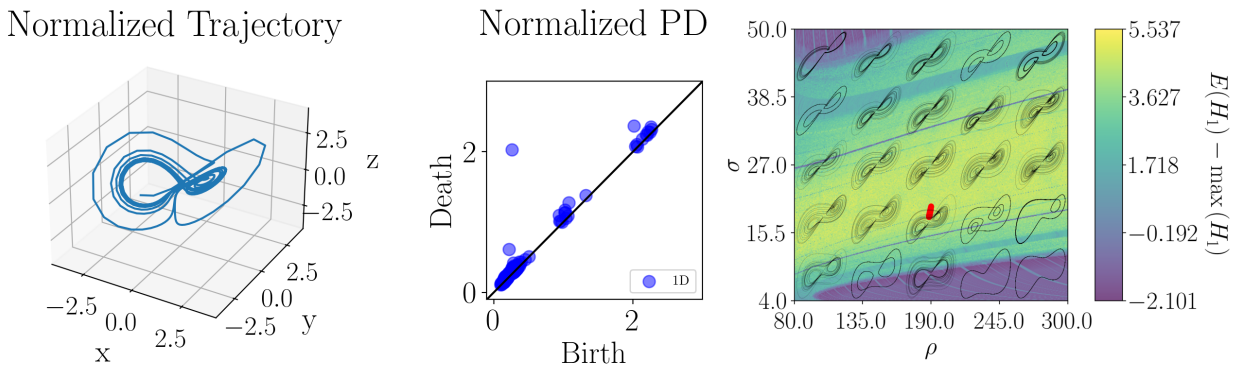


Figure 3.23 Final Lorenz trajectory after 385 epochs with a learning rate of 0.1 and no decay.

of the parameter space, the learning rate was then increased to one which is significantly larger than typical optimization problems, but the learning rate directly corresponds to the step size in the parameter space so it is justified in this problem as long as learning rate decay is used to ensure convergence. With a decay rate of 1% per epoch, the resulting path after 382 epochs is shown in Fig. 3.24. We see that the path was able to escape the chaotic region and reach a periodic solution,

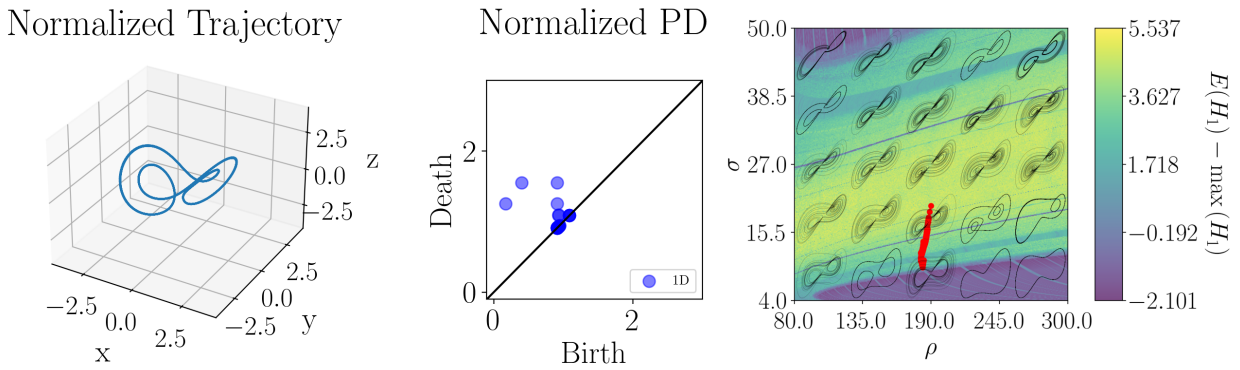


Figure 3.24 Final Lorenz trajectory after 382 epochs with a learning rate of 1.0 and a decay rate of 0.99.

but the optimizer did not have enough momentum to reach the periodic solutions in the region for low values of σ. To allow for more exploration, in the final example the learning rate remained one and the decay rate was reduced to half a percent per epoch. In this case, the optimization was carried out to 2100 epochs to verify convergence, but the periodic solution was found after about 300 epochs. The resulting path and final trajectory are shown in Fig. 3.25 For completeness, the ρ and σ components of the path are also plotted with respect to epoch in Fig. 3.26 where it is clear

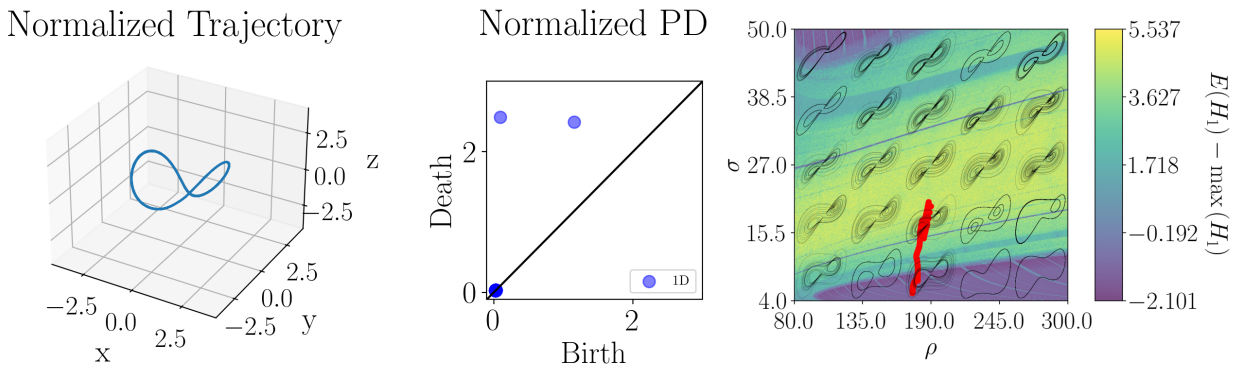


Figure 3.25 Final Lorenz trajectory after 2100 epochs with a learning rate of 1.0 and a decay rate of 0.995.

that the path converges to a single pair of parameters at $\rho = 175.996$ and $\sigma = 6.776$.

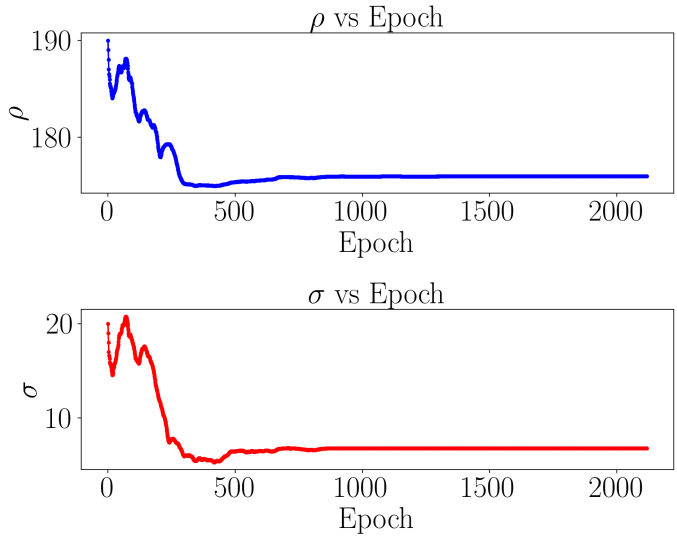


Figure 3.26 Lorenz system path components with respect to optimization epoch to demonstrate convergence.

3.5.5 Numerical Validation Conclusions

Exploring dynamical system parameter spaces is a highly nontrivial task and there are many different approaches to solving the problem. I chose to harness the connection between topology and dynamical systems to navigate parameter spaces by optimizing topological features of a dynamical system trajectory using persistence optimization. This resulted in the creation of a new language for defining topologically driven loss functions that map to different response character-

istics. The loss function dictionary allows for promoting or avoiding limit cycles, fixed points and chaos for a dynamical system. Many choices need to be made by the user for this method to work such as the simulation time and sample frequency for simulations. If the transient behavior is not properly removed from the point cloud the persistence diagrams will be incorrect, but if the simulation time and sample frequency are too high, the computation times quickly become unreasonable. The user also needs to choose a learning rate and decay rate to optimize between exploration and exploitation. Once these factors have been tuned, the loss function can be intuitively designed to promote or avoid different features or regions of the parameter space. It was found that the most important decisions for the success of this method were adding gradient clipping to avoid exploding gradients in chaotic regions of the parameter space and balancing the loss function terms in the multi-objective optimization examples. These choices led to successful demonstrations over many examples from four different dynamical systems. In future work, it will be crucial to add functionality to account for unstable system responses, and avoid them at all costs or search for them for characterizing unsafe bounds on system parameters. In this work the parameter spaces were specifically chosen to not contain any unstable solutions.

CHAPTER 4

TEXTURE ANALYSIS

This chapter outlines the texture analysis techniques using TDA. Specifically, data from a manufacturing process called Piezo Vibration Striking Treatment (PVST) where a texture is intentionally produced on a surface to improve its mechanical properties was analyzed using TDA to quantify consistency in specific features of the texture. Three features of the textures were quantified: depth, roundness, and pattern shape. Depth and roundness methods originally published in [7] are presented in Section 4.1. The pattern shapes are characterized in Section 4.2 and originally published in [8].

4.1 Characterizing Depth and Roundness

Quantifying patterns in visual or tactile textures provides important information about the process or phenomena that generated these patterns. In manufacturing, these patterns can be intentionally introduced as a design feature, or they can be a byproduct of a specific process. Since surface texture has significant impact on the mechanical properties and the longevity of the workpiece, it is important to develop tools for quantifying surface patterns and, when applicable, comparing them to their nominal counterparts. While existing tools may be able to indicate the existence of a pattern, they typically do not provide more information about the pattern structure, or how much it deviates from a nominal pattern. Further, prior works do not provide automatic or algorithmic approaches for quantifying other pattern characteristics such as depths' consistency, and variations in the pattern motifs at different level sets. This paper leverages persistent homology from Topological Data Analysis (TDA) to derive noise-robust scores for quantifying motifs' depth and roundness in a pattern. Specifically, sublevel persistence is used to derive scores that quantify the consistency of indentation depths at any level set in Piezo Vibration Striking Treatment (PVST) surfaces. Moreover, we combine sublevel persistence with the distance transform to quantify the consistency of the indentation radii, and to compare them with the nominal ones. Although the tool in our PVST experiments had a semi-spherical profile, we present a generalization of our approach to tools/motifs of arbitrary shapes thus making our method applicable to other pattern-generating

manufacturing processes.

4.1.1 Introduction

Extracting information from surface images is an important field of research with many applications such as medical imaging [102], remote sensing [103, 104], and metrology. In many instances, the texture on the surface represents a pattern with a tessellation of a repeating, base geometric shape called a motif. These patterns might be intentionally introduced either for functional reasons, e.g., adding friction, or to realize certain aesthetics. Alternatively, surface patterns can be an inevitable side effect of the process that generated the surface, such as machining marks.

Characterizing the resulting patterns can provide valuable information on the surface properties, and it can serve as a useful diagnostic of the production process. The quantification of patterns depends on the involved motifs. For example, a pattern of zero-dimensional motifs (points) is characterized by the lattice formed by the points. One-dimensional motifs (lines) can produce patterns that are characterized by the lines' geometry and the spacing between them (for parallel lines). Patterns can also emerge in two dimensions as a result of the line intersections.

One-dimensional motifs (lines) can produce patterns that are characterized by the lines' geometry and the spacing between them (for parallel lines), or by the two dimensional pattern that can result from line intersections.

Of particular interest is the challenge of characterizing three dimensional patterns imprinted onto nominally planar surfaces. This scenario applies to many scientific domains that use image data to extract information about certain systems or processes. The image can be viewed as a spatial height map that contains information about the motifs. In particular, in this setting quantities of interest include the structure of the two-dimensional projections of the motifs' centroids, the motifs' depths consistency, and the regularity of the shape of the generalized cones produced from intersections of level sets with the motifs. For example, if the motifs are tessellated semi-spheres in the plane, then the quantities of interest are the centers of the circular two-dimensional projections, the depths of the semi-spheres across the surface, and the deviations of the circles' perimeters as a function of the motifs' height.

One specific field where surface texture description plays an important role is at the intersection of manufacturing and metrology. Surface metrology of manufactured parts is directly related to fit, wear, lubrication, and corrosion [105] as well as fatigue resistance [106–108]. In additive manufacturing, surface texture is further used to understand and optimize the process [109–111].

In the field of manufacturing, texture analysis is also a valuable quality control tool that can be used to investigate the effectiveness of a manufacturing process and obtain information about the current state of the machine being used [112]. For example, it has been shown that surface textures can be analyzed to identify the occurrence of chatter in a machining process [113–118]. Surface texture analysis has also been used to monitor and indicate tool wear in a machining process [119–126], detect surface defects such as cracks and scratches [127–131], and for quantifying surface roughness of a part [132–134]. Surface texture can also have a significant effect on the mechanical properties of a part, and as a result, a number of processes have been developed to intentionally introduce surface texture in order to obtain improved mechanical properties. Examples of such a processes include shot peening, elliptical vibration cutting and texturing, and piezo vibration striking treatment (PVST). Shot peening has been shown to improve properties such as the roughness, hardness and wear resistance of a part [135–138] and can increase the ultimate and yield strengths [139, 140]. Elliptical vibration cutting is another process that results in a surface texture left behind on the part by inducing another direction of motion in the cutting process creating an elliptical cutting pattern [141]. These cuts leave a texture behind on the surface of the part that reduces tool wear and burrs, and improved surface properties such as roughness [142]. Models have also been developed to describe the relationships between the system parameters and the resulting textures for this process [143]. Another example of a process that exploits surface texture for improving mechanical properties is piezo vibration striking treatment (PVST) [144], see Section 4.1.1.2. This paper mainly focuses on analyzing results from the PVST process, but avenues are offered for studying textures with differing properties.

Most classical applications of texture analysis involve high resolution gray-scale images that provide depth information of the surface. A variety of different methods have been used for ana-

lyzing these images ranging from statistical techniques to wavelet transform approaches [124, 134, 145]. The classical approaches can be grouped into four categories that are summarized in Fig. 4.1. For statistical methods, the gray level co-occurrence matrix (GLCM) is usually of interest in which

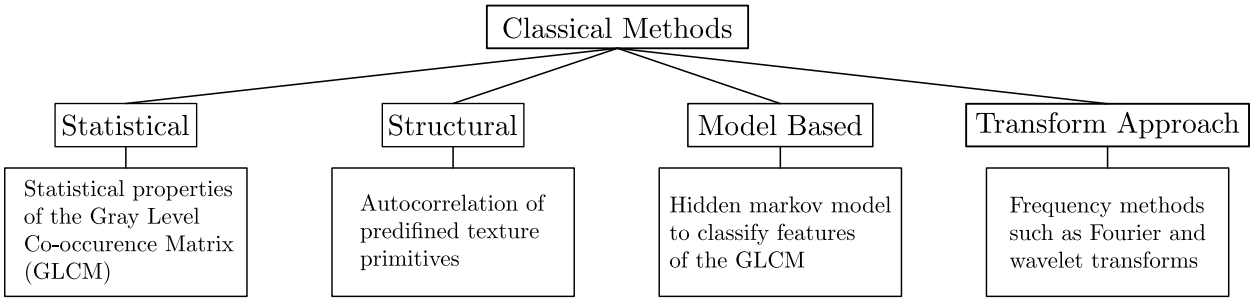


Figure 4.1 Block diagram summarizing the classical texture analysis methods and their basic descriptions.

a matrix is obtained containing information on the probabilities that adjacent pixels would have the same intensity [145]. Statistical measures are then computed on this matrix leading to quantification of broad features such as *smoothness*, *coarseness* and *regularity* of a texture [113, 146].

Another method of texture analysis is referred to as *structural texture analysis*. This method works best for tessellated patterns of predefined fundamental features called primitives [146]. Statistical quantities such as the image autocorrelation function provide information about the sizing of the primitives and a quantification of the texture periodicity [146]. The problem with this method is that the primitives and relative positions need to be manually defined by the user, and the results can vary significantly based on these decisions [146, 147].

The final two methods are model based approaches and transform approaches. Model based methods utilize statistical models such as a Hidden Markov Model [119] to classify texture features from the gray level co-occurrence matrix. Lastly, the transform approach uses frequency methods such as Fourier or wavelet transforms to extract information about feature frequency or relative sizing in the texture [147]. However, with transform methods, relative positioning of the texture features is lost in the process and further analysis is required to obtain this information [148]. With all of the methods discussed so far, expert knowledge of the process/analysis is required for interpreting the results, and it is difficult to target a specific feature in a texture such as the specific

pattern shape or depth of features.

4.1.1.1 Topological Approaches to Texture Analysis

This paper describes a Topological data analysis (TDA) approach for quantifying surface texture and pattern, and it shows the validity of this approach by applying it to PVST surface images. Figure 4.2 shows an overview of the developed pipeline, and the first box in the figure shows an example surface image. While our prior work extended the the TDA approach in [149] to classify surface patterns formed by the indentation centers in PVST processes [8] (second box in Fig. 4.2), quantifying the consistency of indentation depths (third box in Fig. 4.2), and characterizing generalized radii of indentation shapes, e.g., the profile of the indenter at different heights (last box in Fig. 4.2) are two important problems that have not been addressed before.

Specifically, the striking *depth* and *roundness* of semi-spherical PVST indenters are essential for characterizing a PVST surface and they enable predicting the impact forces in the PVST process [144]. Quantifying these properties allows process control and ensures consistent mechanical properties for the part, if the impact forces are constant from strike-to-strike. We provide a framework for automatically characterizing general patterned texture, and apply it to quantitatively describe PVST surfaces. Within this framework, we characterize striking depth and roundness from PVST surface images using sublevel persistent homology (a tool from TDA). Another contribution of this work is locating the specific feature depths to locate a reference height for the surface. This enables not only quantifying the indentation roundness at different heights, but it also allows estimating surface deviations from the theoretical $z = 0$ reference plane, e.g., the surface slope. The developed tools, along with our previously described method for quantifying the patterns of the indentation centers [8], provide a quantitative approach for characterizing surfaces from texture-producing processes such as PVST.

We start by describing the PVST process in 4.1.1.2. We then introduce the relevant TDA background followed by derivations for the theoretical expressions used for quantifying texture features in Section 4.1.2. The results are presented in Section 4.1.3, and the concluding remarks are listed in Section 4.1.5. Finally, CAD simulation of PVST patterns, a feature score noise analysis,

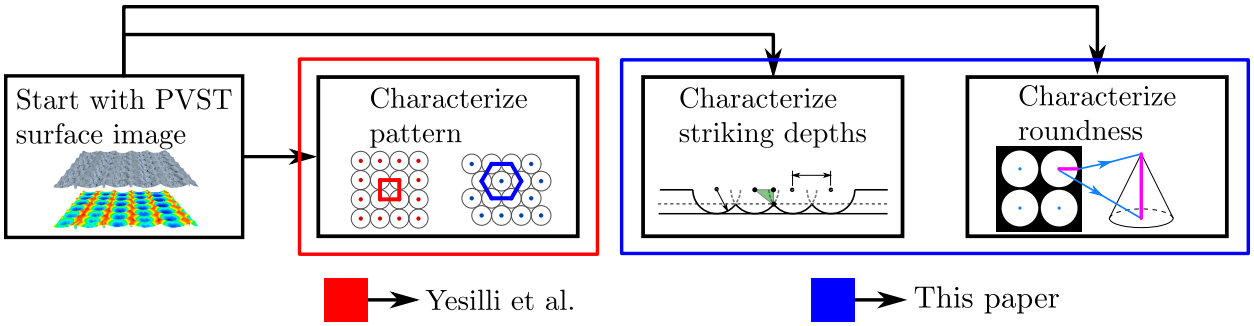


Figure 4.2 An overview flow chart for PVST texture characterization. Starting with a PVST image, three main features can be classified (depth, roundness, and pattern).

and surface slope and angularity estimation are included in the appendices.

4.1.1.2 Piezo Vibration Striking Treatment (PVST)

PVST is a process in which a piezo stack controlled with a CNC machine is used to impact the surface at a specific frequency leaving behind a surface texture on the part. Geometric characteristics of the texture are chosen by varying process parameters such as the shape of the indenter, the impact frequency, and scanning speeds. The diagram shown in Fig. 4.3 demonstrates how the PVST process generates a texture on the surface as a result of the process parameters. We see that

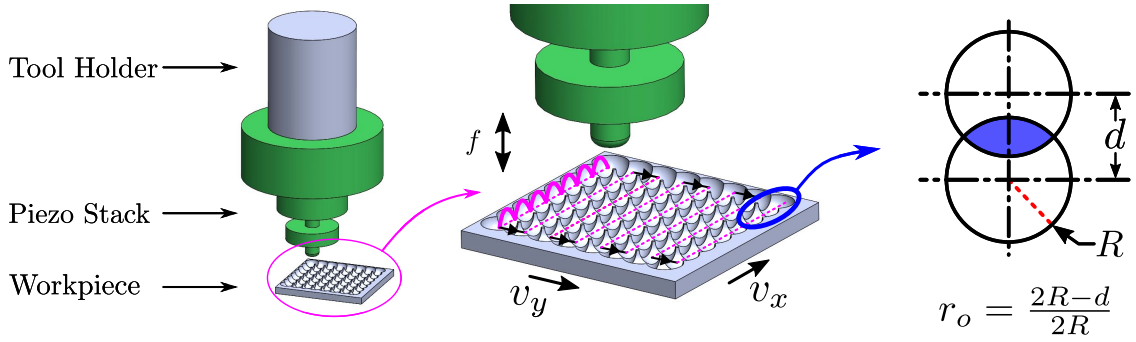


Figure 4.3 PVST diagram showing the mechanics of the PVST process and how the texture can be controlled using the frequency f, the in plane scanning speeds v_x and v_y, and the overlap ratio r_o.

the piezo stack produces oscillations in the impact tool that plastically deform the surface at regular intervals, and the stack is translated in the plane using the CNC machine to produce the texture. Parameters such as the oscillation frequency f, in-plane speeds v_x and v_y, and the overlap ratio r_o can be varied to produce different textures. As a result, it is important to be able to compare the output surface texture to the nominally expected texture based on the input process parameters.

This comparison will allow for quantification of the process effectiveness and ensuring that the mechanical properties are within the expected tolerances compared to the results for the nominal texture.

4.1.2 Background and Theory

Section 4.1.2.1 provides a brief background on persistent homology, the main tool from TDA that we use in this work. Sections 4.1.2.2 and 4.1.2.3 show the derivations for the expressions that will be used to score the strike depths and roundness, respectively, of the PVST surfaces. Section 4.1.2.3 shows how the process knowledge was applied to locate the strike minima and obtain the surface reference height. Section 4.1.2.4 describes how our approach can be generalized to other tool shapes.

4.1.2.1 Persistent Homology Background

Persistent homology (PH) is a tool from topological data analysis (TDA) that allows for quantification of features in a data set by providing information about things like connectivity and loops in the data. We will describe PH through the lens of a PVST image rather than presenting abstract homology constructs, and we refer the reader to [150] for a comprehensive presentation of TDA.

In this work, we use a specific type of PH called sublevel set persistent homology in which a height function is defined on the image. Let I be the $p \times q$ image matrix of interest defined on the interval $[0,1]$. We define a parameter $T \in \mathbb{R}$ to be an arbitrary height in the image and $I_T = f^{-1}[0,T]$ to be a new image that is obtained by taking the sublevel set of I up to a height T. Parameterizing the image sublevel sets allows for the topology to be studied as T is varied using persistent homology. The topology is determined for each sublevel set of the image by only including pixels with gray scale values at or below the threshold T, and the homology is computed at each height [151]. This allows tracking the birth and death of **connected components** in the image, and the formation of **loops** in the process as T is increased.

We illustrate the concept of sublevel PH using a synthetic surface constructed by superimposing 6 Gaussian distributions as shown in Fig. 4.4 (d). This surface can be compared to a PVST grid if each Gaussian distribution is imagined to be a strike in the PVST scan. The example image

shows 6 prominent structures (blue) resulting from the Gaussian distributions and due to the relative positions, we see two loops in the image between the 6 components shown as orange circular structures. We will use sublevel persistent homology on this image to capture the aforementioned features in a quantifiable manner. The image in Fig. 4.4 (d) was thresholded for all $T \in [0,1]$ and persistence was used to determine the image topology at each height and to track the formation of connected components and loops in the image. We note that the 0D homology or H_0 tracks the connectivity of the features and 1D homology or H_1 tracks the loops in the persistence diagram. The example in Figs. 4.4 (e-g) shows three different level sets for the full surface with corresponding binary images in Figs. 4.4 (a-c). Starting with Fig. 4.4 (a), it is clear that 6 components were born in the image at this threshold, but two of them have connected or merged at this height. This connection is indicated in the persistence diagram by plotting the $(birth,\ death)$ coordinate for the younger of the two classes, i.e., the class that appeared at a higher T value. We plot this connection as a red point with coordinate $(0.17, 0.27)$ in Fig. 4.4 (h). Figure 4.4 (b) shows that increasing T to 0.6 causes all 6 classes to connect into one component that persists to ∞. This is shown in the persistence diagram by plotting 4 more points at $(0.105, 0.38)$, $(0.11, 0.49)$, $(0.08, 0.56)$, and $(0.11, 0.59)$. The final red point indicates the infinite lifetime of the overall object on the dashed line. Note also that at a threshold of 0.58, the left loop is born meaning that a closed loop can be formed in the white region around a black region as shown in Fig. 4.4 (b) at $T = 0.6$. A second loop is born at 0.602 shown in Fig. 4.4 (c) at $T = 0.65$. When the threshold height reaches the point where the loops fill in with white in the level set, the loop dies and the point is plotted in the persistence diagram. For this example, the loops are born at 0.58 and 0.602, and die at 0.69 and 0.80 respectively as shown in Fig. 4.4 (h). As T reaches its highest value at 1 (Fig. 4.4 (h)), the full persistence diagram is obtained. The loop on the right side is visually larger than the left one in Fig. 4.4 (d), and this is indicated by the top blue square point having a larger distance to the diagonal in the persistence diagram giving that loop a longer *lifetime*. The distribution of points in the persistence diagram can then be studied to compare to the expected distribution of persistence pairs for a nominal surface.

A major benefit of utilizing sublevel persistence to study various features of a function is that it has been shown to be stable under small perturbations due to noise [152]. Specifically, the bottleneck distance between persistence diagrams is defined as $d_B(X,Y) = \inf_\gamma \sup_x ||x - \gamma(x)||_\infty$ where $x \in X$ and $y \in Y$ are the persistence diagrams (birth and death coordinates) and γ is the set of possible matchings between X and Y. If one diagram contains more persistence pairs, those pairs are matched to the diagonal in γ. The main theorem in [152] states that for two continuous well-behaved functions, f and g, the bottleneck distance satisfies,

$$d_B(D(f), D(g)) \leq ||f - g||_\infty, \tag{4.1}$$

where $D(f)$ and $D(g)$ are the sublevel persistence diagrams for f and g. Assume that f is the nominal texture surface and g is the same texture that contains additive white noise. We represent the textures here as functions $f, g : \mathbb{R}^2 \to \mathbb{R}$ where the output of the functions is a depth map for the texture. Equation (4.1) states that the bottleneck distance between the nominal and noisy surface persistence diagrams will remain bounded by the largest deviation between the surfaces. This result allows for noise robust comparisons between the nominal and experimental texture persistence diagrams.

4.1.2.2 Strike Depth

In order to compare the experimental persistence results with the nominal surface pattern, we need to derive expressions that describe the persistence of nominal patterns as a function of the process parameters. We start with the PVST strike depth, and we consider the scenario of deriving the sublevel persistence of a nominal PVST grid.

Theoretical Expressions: Based on the PVST process inputs, we expect the ideal texture to consist of a square grid of overlapping circular indentations where all strikes have uniform depths. Consider the side views of a single row and column in a perfect PVST lattice with arbitrary overlap ratios in Fig. 4.5, where R is the nominal radius of the circle obtained from a PVST impact, d_x and d_y are the horizontal and vertical distances between centers accounting for overlap ratios. In general, the grid does not have to be square so we derive our expressions assuming a general grid shape and apply the special case for a square grid later. In the horizontal direction, the overlap

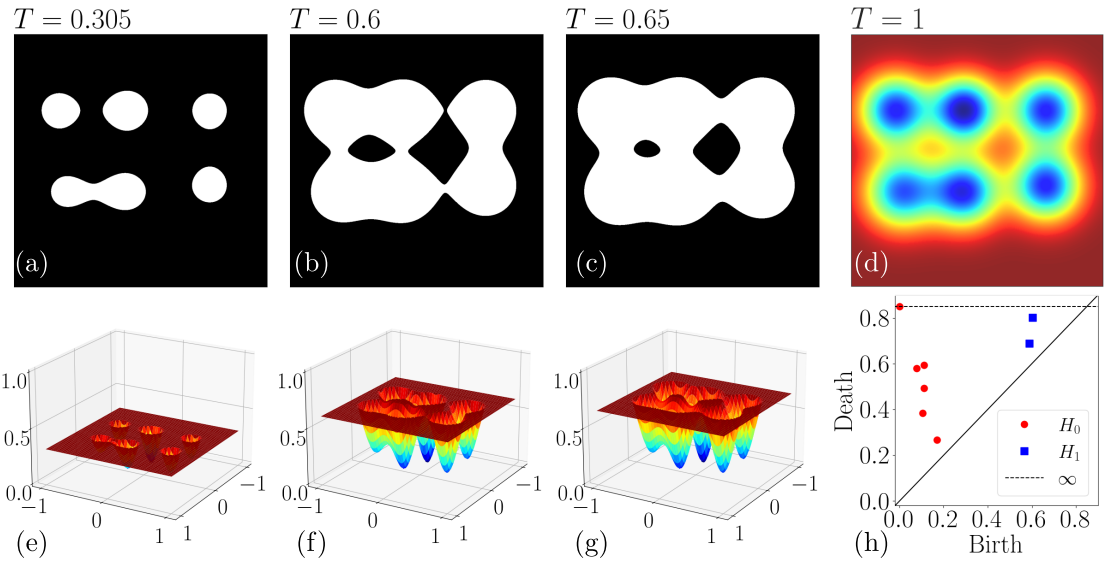
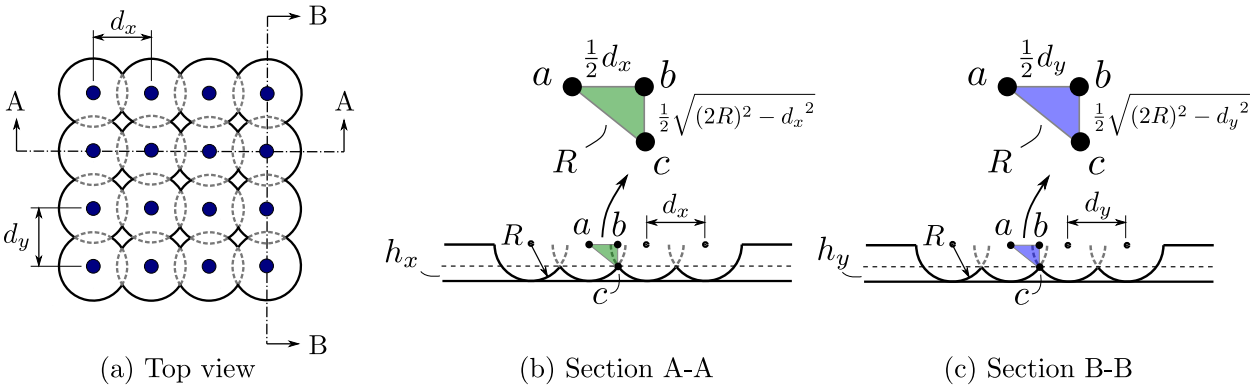


Figure 4.4 Sublevel persistent homology example. (a-c) shows the surface level set represented as a binary image at 3 threshold heights, (d) shows the full surface image (e-g) shows the corresponding 3D surface plots for the thresholded images and (h) shows the full sublevel set persistence diagram for the surface.



(a) Top view (b) Section A-A (c) Section B-B

Figure 4.5 Arbitrary PVST lattice diagram with a grid top view (a), section views for the strike rows (b) and columns (c) to illustrate the geometry of a PVST grid.

ratio is defined by

$$r_x = \frac{2R - d_x}{2R}, \tag{4.2}$$

where r_x is the overlap ratio in the x direction. Using the geometric expressions in Fig. 4.5, h_x measured from the maximum depth of the impact can be computed using

$$h_x = \frac{1}{2}\left(2R - \sqrt{(2R)^2 - d_x^2}\right), \tag{4.3}$$

where h_x is the height at which all of the impact **rows** merge. Combining Eq. (4.2) and Eq. (4.3) to eliminate d_x gives an expression for the height h_x in terms of the impact radius and the overlap ratio

$$h_x = R\left(1 - \sqrt{(2 - r_x)r_x}\right). \tag{4.4}$$

Similar expressions can be obtained for the vertical direction by replacing x with y. In order to normalize Eq. (4.4), we rescale the radius of the PVST strikes at maximum depth to one. This is consistent with the PVST gray scale images used for the experimental analysis. This means that Eq. (4.4) can be normalized by setting $R = 1$ as this makes the connecting height 1 for an overlap ratio of 0. The normalized heights will be denoted by $\overline{h_x}$ and $\overline{h_y}$, respectively, and can be computed using

$$\overline{h} = 1 - \sqrt{(2 - r)r}, \tag{4.5}$$

where $\overline{h}$ is the height in the x or y direction as a function of the overlap ratio r in the x or y direction respectively. Notice that Eq. (4.5) achieves maximum value when r is zero and minimum value when r is one.

Without loss of generality, we assume that $r_x > r_y$. This means that the horizontal rows will connect before the columns because $h_x < h_y$. Therefore, if there are p rows in the grid, p classes will die at h_x and if there are q columns in the grid, q more classes are expected to die at h_y in the 0D persistence diagram when $p \times q$ classes are born at $h = 0$. A theoretical persistence diagram was generated for the scenario when $q > p$ shown in Fig. 4.6, but in general the relative sizes of p and q can vary depending on the number of rows and columns in the grid. For the images of interest, it was expected that the grid would be square ($p = q = n$) and the overlap ratio was constant in both directions ($r_x = r_y = r$). As a result, we expect n^2 classes to be born at 0 and die at a height h. Table 4.1 shows the expected lifetime of the PVST strikes for different overlap ratios using Eq. (4.5). See Section A for CAD-based simulations used to confirm the theoretical derivations.

Depth Score: In this section we develop a score to quantify the uniformity of striking depths thus allowing a comparison between the experimentally measured depths and their nominal coun-

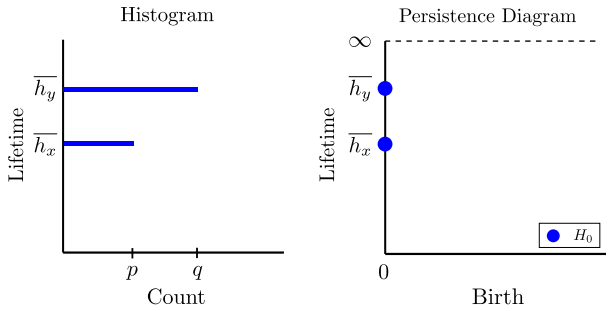


Figure 4.6 Theoretical sublevel persistence diagram and histogram for the striking depths for an arbitrary $p \times q$ grid with critical heights h_x and h_y.

Table 4.1 Expected striking depth lifetimes for different overlap ratios where the grid is square ($n \times n$) strikes, and the heights have been normalized to correspond to a strike radius of 1.

Overlap Ratio	0%	25%	50%
Lifetime ($\bar{h}$)	1	0.339	0.134

terparts. We start by obtaining nominal and experimental histograms to show the sample distributions of the sublevel persistence lifetimes of the strikes. We plot probability density on the x axis, and persistence lifetime on the y axis where the experimental lifetimes come from a direct persistence computation on the image, and the nominal distribution is obtained from the theoretical expressions. Note that the number of histogram bins for the experimental images was determined using Rices Rule which states that the number of bins k is computed using $k = \lceil 2\sqrt[3]{n} \rceil$ where n is the number of persistence pairs in the persistence diagram [153]. Once the two distributions are obtained, we compute the Earth Movers Distance between them to quantify the differences between the distributions [154]. A normalized score was desired to allow for comparison of the earth movers distances for the striking depth distributions. The Earth Movers Distance (EMD) can be analytically analytically computed according to

$$\text{EMD}(u,v) = \inf_{\pi \in \Gamma(u,v)} \mathbb{E}_{(x,y)\sim\pi}[|x-y|], \tag{4.6}$$

where u and v are the two distributions, EMD is the earth movers distance between u and v, and Γ is the set of distributions that exist between u and v. In other words, the EMD computes the minimum amount of work required to transform one distribution into the other [154].

It should be noted that Eq. (4.6) can be used to compare any two distributions u and v, so it

would be straight forward to directly compare the nominal and experimental persistence diagrams to measure the combination of feature depth and surface flatness. While this is a perfectly valid method for quantifying general differences in the textures, it does not directly provide information about a specific texture feature of interest as it considers both birth and death of the features. For this reason, it was chosen to consider the lifetime distributions as probability distributions to isolate the effect of the feature rather than where it is born in the image and provide a path for normalizing scores to quantifying these features in a way that is easy to understand for the user.

For the PVST striking depth distributions, the images have been normalized from 0 to 1. This means that each pixel can only contain a value in the finite interval 0 to 1. As a result, the maximum possible persistence lifetime for a feature occurs when the feature is born at zero and survives for the entire range of the height function. Conversely, the minimum persistence lifetime occurs when the feature is born at zero and survives for an infinitesimal time. The difference between these lifetimes corresponds to the maximal earth movers distance for any two images because Eq. (4.6) is independent of the number of observations. This means that in this case, the earth movers distance has an upper bound where one distribution has all persistence pairs with lifetimes at 1 and the second distribution has all persistence pairs with 0 lifetime. Therefore, the maximum possible earth movers distance in this case is 1, and the distances computed for the different overlap ratios can be directly compared. We define the depth score $0 \leq \bar{D} \leq 1$ according to

$$\bar{D} = 1 - \text{EMD}, \tag{4.7}$$

where $\bar{D} = 1$ when the actual depth distribution is identical to the expected distribution, while $\bar{D} = 0$ when the distributions are the farthest apart. A score between 0 and 1 allows for characterizing the effectiveness of the PVST striking depth distribution as a percentage score where higher percentages indicate improved uniformity in the depth distribution of PVST strikes.

4.1.2.3 Strike Roundness

Since sublevel persistence does not encapsulate spatial information, it cannot be used by itself to characterize roundness of the PVST strikes. Therefore, we needed a tool that can encapsulate that information before using persistence to characterize the shape of the PVST strikes. The tool

we used is the distance transform, which transforms each pixel of the image to display its euclidean distance to the nearest background pixel (black) as a gray scale intensity. Each image needed to be thresholded at a particular height to compute the distance transform, i.e., any pixel below the height is set to black (0) and any pixel above is set to white (1). The distance transform then sets each pixel to a gray scale value encoding that pixels minimum euclidean distance to the nearest black pixel. In other words, the image is transformed to show information about the size of the circles in the third dimension rather than the depths. To obtain theoretical results for quantifying the roundness of the strikes, we first needed to develop a transformation to convert a number of pixels into a physical distance as described by

$$x = \frac{n_p w}{P}, \tag{4.8}$$

where n_p is the number of pixels corresponding to distance x in the image with $P \times P$ pixels and w is the width or height of the image in any desired unit system. We note that x and w must have the same units. Using the nominal process parameters such as the in plane speeds, overlap ratio and frequency, the nominal circle radius can be computed. An example case for computing the nominal radius is as follows: For a frequency of f, a speed v_x in mm/min, image width w in mm, the nominal radius in mm can be computed using

$$R = \frac{v_x}{120f}. \tag{4.9}$$

The factor of 120 is an artifact of the unit conversions from minutes to seconds and division by two to obtain the radius instead of the diameter. The speed v_x is dependent on the overlap ratio with the relationship

$$v_x = 3000(1-r), \tag{4.10}$$

where 3000 mm/min is the speed that results in a 0% overlap pattern at a frequency of 100 Hz.

Substituting the frequency and speed expression into (4.9), we obtain an expression for the nominal circle radius in terms of the overlap ratio,

$$R = \frac{1}{4}(1-r), \tag{4.11}$$

where r is the overlap ratio and R is the nominal circle radius in mm at a PVST frequency of 100 Hz. We then threshold the texture at a height T and compute the circle radius at the given height using the geometry shown in Fig. 4.7 (a). It is clear that as the image threshold height changes, the circle radius also varies due to the geometry of the strikes. Using the Pythagorean theorem we can obtain a relationship between σ, h and R as follows

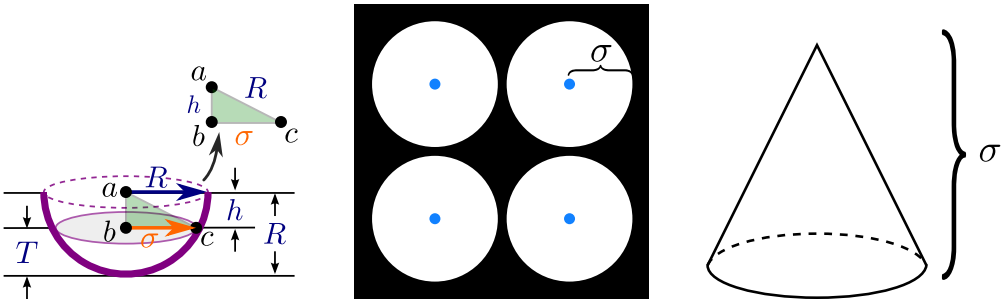
$$h^2 + \sigma^2 = R^2. \tag{4.12}$$

Solving for σ and setting $h = R - T$ yields the following expression for the nominal radius at a given threshold height:

$$\sigma = \sqrt{(2R - T)T}, \tag{4.13}$$

where T is the threshold height from the bottom of the strike in mm. Using this information, we can threshold the image at various heights and apply the distance transform to allow for sublevel persistence to be used for measuring the strike roundness. Basically, the distance transform is used to encode spatial information as height information, thus allowing us to leverage sublevel persistence for scoring strike roundness as described in the following sections.

Sublevel Persistence for no overlapping strikes ($T < \overline{h}$): Consider the PVST grid with no overlap shown in Fig. 4.7 (b). When the distance transform is applied to the thresholded grid, spatial information about the size of the circles is encoded as height information in the shape of cones (Fig. 4.7 (c)). As the distance from the edge of the circle increases, so does the height of the cones which can be understood from Fig. 4.7 (b). Applying sublevel set persistence to this grid of cones allows quantifying the roundness of the circles. We see that as the height of the connectivity parameter is varied starting at the bottom of the cones, 1 0D class is born at time 0 and remains to ∞. Applying one-dimensional persistence to the $n \times n$ grid of cones we expect n^2 1D classes to be born at 0 and die at σ. 1D persistence was chosen for the roundness application because we are interested in the lifetimes of cycles in the images as they will provide information about the roundness of the strike. This was not necessary for the depth measurements because we only needed to know the depth at which the strikes connected.



(a) Impact Geometry (b) Binarized Image (c) Distance Transform

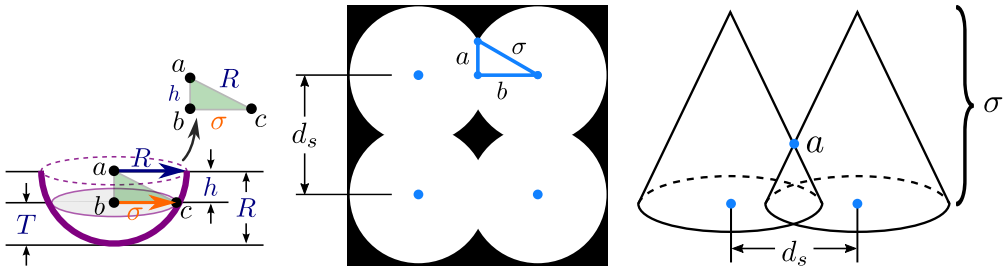
Figure 4.7 Converting between a binarized image and its corresponding distance transform geometry. (a) shows the strike geometry used for converting threshold heights to the radius of the strike at that height, (b) shows the thresholded image at a height below the critical height $T < \bar{h}$ (no overlap), (c) shows the cone geometry resulting from the distance transform of the binarized image and how the strike radius σ appears in each form.

Sublevel Persistence for overlapping strikes ($T \geq \bar{h}$): We now generalize the result from the case with no overlap by thresholding the image above the critical height ($\bar{h}$) where we obtain an image with overlapping circles shown in Fig. 4.8 (b). The critical height is the depth at which water would overflow from the strike into the other strikes and it can be computed using,

$$\bar{h} = R(1 - \sqrt{(2-r)r}), \tag{4.14}$$

where $\bar{h}$ is the critical height, R is the nominal circle radius, and r is the overlap ratio. We define a new parameter ε to indicate the threshold height T in terms of the critical height $\bar{h}$ using $T = \varepsilon\bar{h}$ where ε defines the threshold height relative to the critical height. If $\varepsilon < 1$, the circles in the thresholded image do not overlap and the no overlap case is used, whereas if $\varepsilon \geq 1$, the circles will overlap and a more general relationship needs to be considered. In Fig. 4.8 (a) we show a binarized image where $\varepsilon > 1$ with strike overlap. When this thresholded image is distance transformed, a result similar to Fig. 4.8 (c) is obtained where a critical distance a needs to be considered. The distance a is the height in the distance transformed image where the gap between the cones connects to the surrounding object. Above a, the circles also disconnect in the filtration so we have a formation of cycles that can be considered when performing sublevel persistence. To obtain an expression for a, we consider the triangle shown in Fig. 4.8 (b) and apply the Pythagorean theorem

$$a = \sqrt{\sigma^2 - b^2}. \tag{4.15}$$



(a) Impact Geometry (b) Binarized Image (c) Distance Transform

Figure 4.8 Converting between a binarized image and its corresponding distance transform geometry. (a) shows the thresholded image at a height below the critical height $T \geq \overline{h}$ **(overlap present)**, (b) shows the cone geometry resulting from the distance transform of the binarized image and how the strike radius σ appears in each form, and (c) shows the geometry used to obtain expressions for the cone intersection height a.

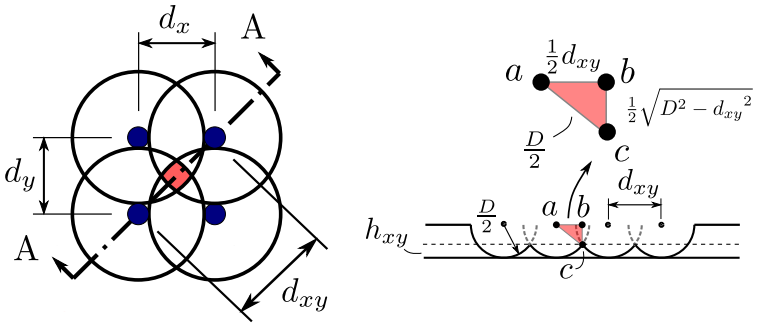
An expression was needed for the side length b in terms of other known parameters. For this, the center-to-center distance d_s of the circles was used because we know that $d_s = 2R(1-r)$ from the definition of the overlap ratio. At this point it is important to note that this expression depends on the full nominal radius R and should not be written in terms of σ because d_s remains invariant for all threshold heights. Observe that, $d_s = 2b$ from Fig. 4.8 (b) due to the circle position remaining constant. Applying the definition of d_s to the result for b we obtain an expression for b in terms of known parameters

$$b = R(1-r). \tag{4.16}$$

Substituting b into Eq. (4.15) gives the critical distance

$$a = \sqrt{\sigma^2 - R^2(1-r)^2}. \tag{4.17}$$

Effect of High Threshold: Lastly, we consider the gaps between the cones at higher overlap ratios. For low overlap ratio, the gap heights will span the entire depth of the strike, but as the overlap ratio increases, the gap height eventually begins to decrease causing the cycles to have lower lifetimes. To quantify this result, we needed to compute the height of the gaps as a function of the overlap ratio. Consider the grid diagonal cross section shown in Fig. 4.9. We see that this section view results in the same triangle that was used to determine the closing height when categorizing the striking depths with the difference being the addition of the value d_{xy}. This value



(a) Top view (b) Section A-A

Figure 4.9 Nominal PVST grid with a high overlap ratio to demonstrate diagonal cross section overlap height.

can be computed using

$$d_{xy} = \sqrt{d_x^2 + d_y^2}, \tag{4.18}$$

or if the grid is square

$$d_{xy} = 2\sqrt{2}R(1-r), \tag{4.19}$$

where R is the nominal strike radius. If we apply the Pythagorean theorem in the same way as the depth results, we obtain an expression for h_{xy},

$$h_{xy} = R(1 - \sqrt{1 - 2(1-r)^2}). \tag{4.20}$$

Substituting for an overlap ratio of 0.5, and the nominal radius of 1 due to the normalized depths, we obtain a value of $h_{xy} = 0.29289$. It should be noted that h_{xy} will be equal to the nominal radius R as long as the following inequality is satisfied,

$$d_{xy} > 2R. \tag{4.21}$$

Here, if we assume equality, and substitute Eq. (4.19), we find that this corresponds to an overlap ratio of $r = 0.29289$. Because the 50% overlap ratio case is larger than 29.28% overlap, we needed to consider that all of the 1D persistence loops merge into a single loop above the height h_{xy} whereas this phenomena was not present in the lower overlap ratio cases. This single component will have zero lifetime if the grid continues on forever.

Roundness Expected Results Summary: We summarize the PVST roundness expected 1D persistence results for the distance transformed images as follows:

(a) Binarized Image (b) Distance Transform

Figure 4.10 50% overlap ratio image thresholded at the reference height ($T = 0.24$) found by taking the first threshold height that contained $19^2 = 361$ features in the persistence diagram. The binarized image is shown on the left and distance transformed image on the right.

1. If $\varepsilon < 1$, we expect n^2 classes to be born at 0 and die at σ.

2. For $\varepsilon > 1$, we expect 1 class to be born at 0 and die at σ, and $n^2 - 1$ classes to be born at a and die at σ.

3. If $r > \frac{2-\sqrt{2}}{2} \approx 0.29$ and $T > h_{xy}$, we expect one object to be born at time 0 and die at 0.

See Appendix A for CAD-based simulations that confirm the theoretical results.

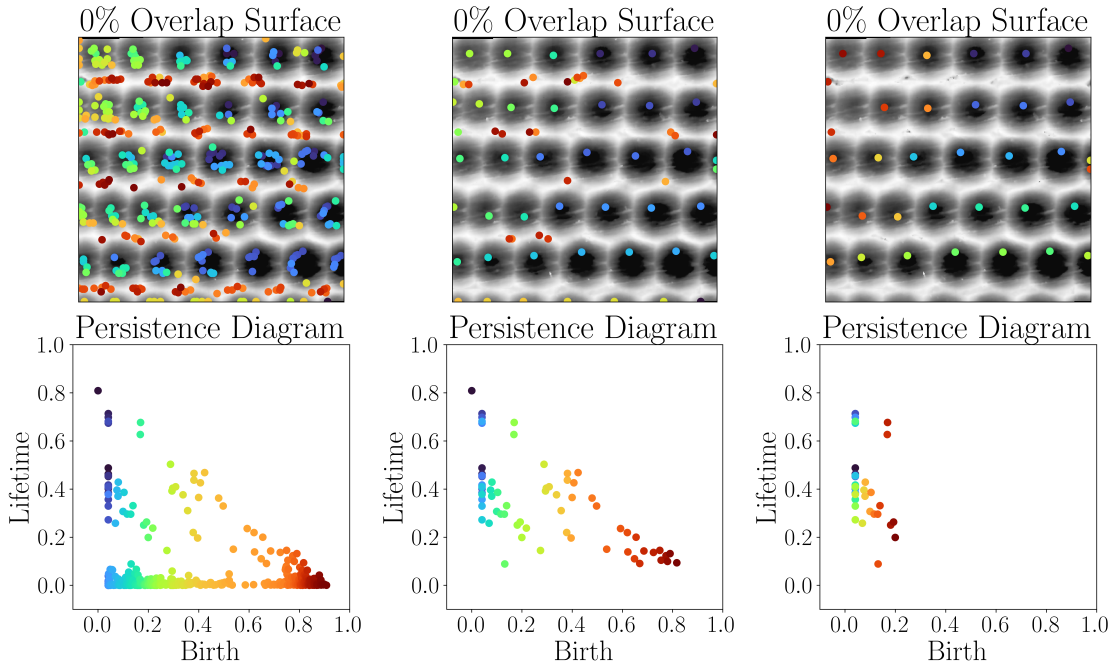
Finding Reference Heights: Due to variations in strike forces, initial surface heights, and artifacts in the images the strike minima do not lie uniformly at a height of 0 in practice, so a reference plane is required to determine what height to compare the roundness results with using the theoretical model. If the reference height is not used, then a shift would be present in the results that would skew the final roundness measurements. The first attempt at locating a reference height was to use the first height at which the persistence diagram contained a number of pairs equal to the number of strikes in the image. The problem with this approach is that the features that were obtained were due to noise in the image and very few of the strike minima were present in the image as shown in Fig. 4.10 where we see the first threshold height in the 50% overlap image that contains $19^2 = 361$ features. It appears that no strikes have been located in the top left corner of the image so this would be a poor estimate of the reference height for this image.

In order to obtain a better reference height, we needed to first utilize knowledge of the surface to filter the persistence diagram down to the features that corresponded to the strike minima of the

surface.

We locate the strike minima by computing sublevel persistence on the surface and taking the birth times to be the minima of each feature. These points are plotted as shown in Fig. 4.11 (a). The critical points have been matched to their location in the persistence diagrams by color. From the color coding, it was clear that the blue/purple/green features corresponded to the strikes and the orange/yellow/red features corresponded to locations between the strikes. This observation allowed for the persistence diagram to be filtered in order to obtain the features of interest. Note that these images have been down sampled to 300×300 down from 6000×6000 to reduce the number of features in the image. The process begins by observing that there were approximately 35 features in this image, so the goal was to algorithmically filter the persistence diagram such that the resulting 35 features correspond to the actual strike minima. We start by filtering out low lifetime persistence pairs by computing a histogram of the lifetimes, and thresholding the lifetime above any point that contained a bar height larger than the number of desired features. This threshold resulted in the persistence diagram shown in Fig. 4.11 (b). It is clear that the features removed up to this point are attributed to noise as we see that each strike still retained at least one critical point after this step. We also observe that the features born at exactly time zero are due to the artifacts in the image, so the birth times were restricted to be larger than 0. The final step is to remove critical points from the right of the persistence diagram (red region) until only the desired number of features remain in the image; the result of this step is shown in Fig. 4.11 (c). The remaining features in the final filtered persistence diagram are taken to be the strike minima and the average height of these points is used as the reference height. Applying this process to the 25% and 50% overlap images yielded the results in Fig. 4.12. We see that the located features in the filtered persistence diagrams are exceedingly close to the true strike minima and taking the average height of these points provided a good estimate of the reference zero height. A byproduct of this process is to enable estimating the surface slope/angularity by computing a regression plane to using the strike minima, as shown in Appendix C.

Roundness Score: To quantify the feature roundness, the images needed to be thresholded at

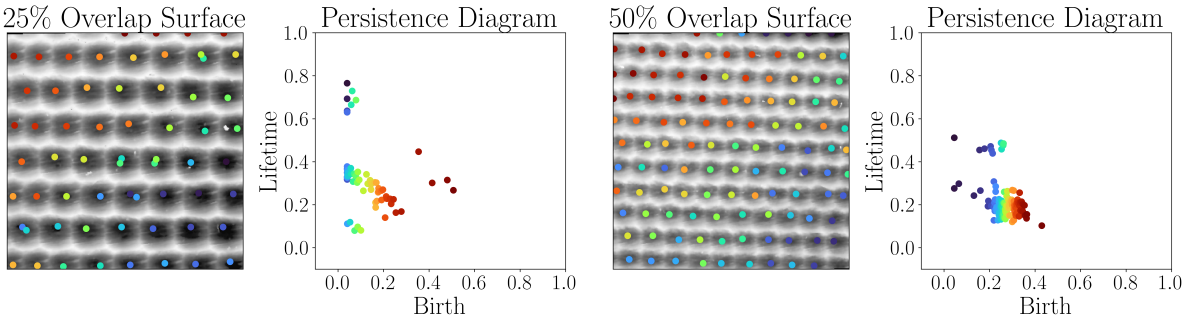


(a) Original Simplified Surface Persistence Diagram.

(b) Low Lifetime Filtered Persistence Diagram.

(c) Final Filtered Persistence Diagram.

Figure 4.11 Persistence diagram (PD) filtering on the simplified surface to locate strike minima. (a) The original PD of the simplified surface, (b) shows the persistence features after removing low lifetime features, and (c) shows the final filtered PD.



(a)

(b)

Figure 4.12 Persistence diagram (PD) filtering on the simplified surface to locate strike minima. (a) 25% Overlap filtered PD, (b) 50% overlap filtered PD.

many different heights to compare the shapes to the nominal distribution over the entire feature. The output of this process is a curve for the earth movers distance as a function of the threshold height of the image. The overall feature roundness is then summarized by computing the area under this curve and diving by the interval width to remove the effects of different reference heights. For

a general impact geometry, the area under this curve can be computed using Eq. (4.22),

$$\bar{R}_G = \frac{1}{1-h_r}\int_0^1 \mathrm{EMD}(T)\,\mathrm{d}T, \tag{4.22}$$

where h_r is the reference height of the image and $\bar{R}_G$ is the generalized roundness score for the texture. We note that this score, by definition, results in a larger score meaning that the texture shape is further from nominal and a lower score is closer to nominal.

In order to obtain a roundness relationship similar to the percentage based depth score, we need to define a roundness score that is specific to the spherical impact by normalizing the area with an upper bound earth movers distance. Similar to the depth score, the earth movers distance at any threshold height is bounded above by two images with all pixels differing by the maximal distance between gray scale intensities. However, the distance transform operation makes it difficult to determine the maximum possible difference in pixel intensities because it is not possible to have all distances at the same value if at least one background pixel exists in the image.

To mitigate this issue, we assume for a reasonably generated physical texture, that the features will be generally close in size to the nominal features. To quantify this assumption, we will say that the experimental feature sizes will have a radius that is at most one nominal radius larger or smaller than the nominal feature size. By assuming that the experimental features are reasonably close in size to the nominal texture, it allows for the earth movers distance to be bounded by the radius at each threshold height and permits the definition of a percentage based score for this feature. For each threshold height of the image, the nominal radius is defined by σ. The σ curve for a spherical feature geometry is defined by Eq. (4.13) as a function of the height T ranging from 0 to R where R is the maximum strike radius. We then introduce the change of variables $T = Rt$ where $t \in [0,1]$ is the image threshold height. This change of variables results in a quarter elliptical curve describing the maximum earth movers distance as a function of threshold height shown in Fig. 4.13.

The area under this quarter ellipse is computed as $\frac{\pi}{4}R$. A roundness score is then defined by normalizing the area under the experimental EMD curve by the quarter ellipse area and subtracting the result from unity to provide a percentage based score similar to the depth score. Equation (4.23)

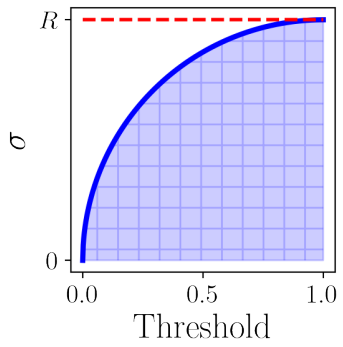


Figure 4.13 Plot of σ as a function of the image threshold height to demonstrate the worst case earth movers distance plot.

shows the spherical impact roundness score as a percentage where a higher score corresponds to the feature roundness being closer to nominal.

$$\bar{R} = 1 - \frac{4\bar{R}_G}{\pi R} \tag{4.23}$$

The resulting score is specific to the spherical impact shape, and if a score is desired for a different impact shape, the σ curve specific to that geometry needs to be obtained that bounds the earth movers distance and the score can be computed in a similar fashion. Note also that if the input experimental texture contains features that differ significantly from nominal this score will be less than zero so it should only be used on textures with feature sizes close in size to nominal. However, the generalized roundness score $\bar{R}_G$ can be used for any such texture, but a lower score means that the texture is closer to nominal in this case. See Appendix B for a quantification of the noise robustness of the depth and roundness scores.

4.1.2.4 Generalizing for Other Textures

While the methods used in this paper were designed to account for features in a PVST image created using a semi-spherical tool, the process can be modified to account for any tool shape. One such example of a generalization of this process arises when a 5-axis milling machine is used to generate a dimple texture on a part. This process leaves behind elliptical dimples which result in improved texture properties [155]. It is clear that the methods used for analyzing a PVST texture will not work for this case. Generalizing the expressions used may introduce significant complex-

ities in the analysis, but we provide two potential avenues for doing this. The first method offered is to apply the techniques in Appendix A where a CAD model is created for the nominal texture and the nominal persistence diagrams can be computed directly from the images for comparison with experimental results. This method is the most straight forward and has been shown to provide results within 5% of the true values for the examples considered in this paper. The second method is to derive expressions for the theoretical persistence lifetimes using a generalized conic section to define the cross section shape. Pattern and depth can apply to any texture being analyzed, but roundness may not be a valid descriptor of the impact shape if it is not spherical. We adopt a *generalized radius* feature that applies to a significantly larger set of indenter geometries to be the generalized conic section [156] described by

$$\rho(x,y) = \sum_{i=1}^{n} \alpha_i ||\vec{x} - \vec{b}_i||_p, \tag{4.24}$$

where $\rho(x,y)$ is the generalized radius as a function of x and y, α_i is the ith weight coefficient, $\vec{x}$ is a vector of coordinates ((x,y) in this case), $\vec{b}_i$ is the ith focal point of the curve, and p is the corresponding p-norm of the vector. For the special case of $n = 1$, $\alpha = 1$, $p = 2$, and b is the center point, we get the equation of a cone which has cross-sections of circles at various heights. Varying the weights and adding more focal points allows for arbitrary shapes to be formed such as the curves shown in Fig. 4.14.

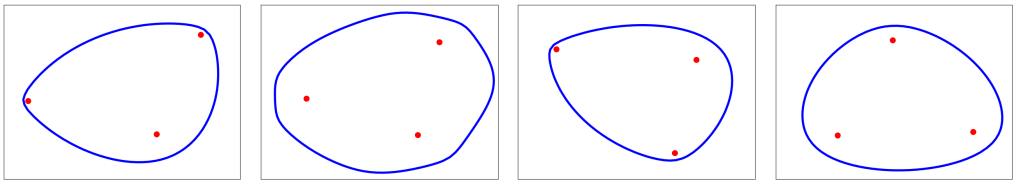


Figure 4.14 Example plots of generalized conic sections to demonstrate different tool shape configurations for generalizing the results in this paper. Red points are the focal points of the conic, and the blue curve represents the cross section of the impact tool.

4.1.3 Results

The theoretical approaches were implemented on three PVST scans at varying overlap ratios being 0%, 25% and 50% to quantify the strike depth and roundness in comparison to the respective

Table 4.2 Striking depth scores for each overlap ratio. Higher is closer to nominal.

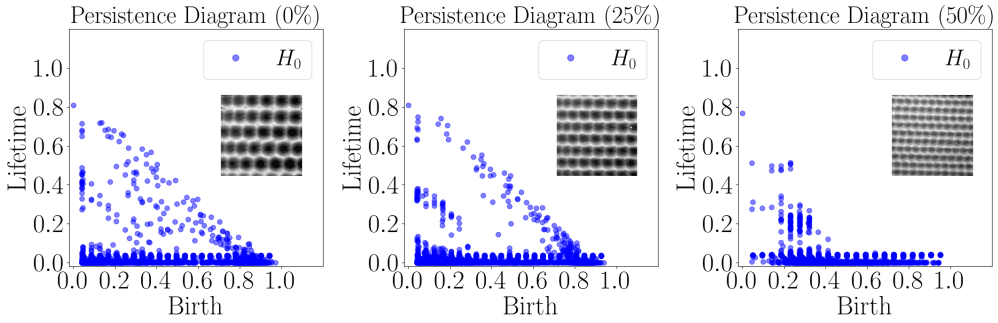
Overlap Ratio	0%	25%	50%
$\bar{D}$	41.04%	86.31%	88.63%

nominal textures. We begin by measuring the strike depths for each image.

4.1.3.1 Strike Depth Results

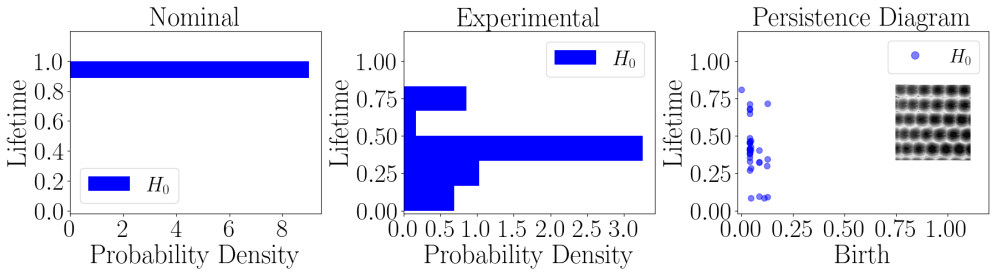
Sublevel set persistence was applied to the PVST images shown in Fig. 4.15 with the corresponding persistence diagrams adjacent to each image. We see a significant portion of the persistence pairs have negligible lifetime and are likely a result of noise in the images. The noise was removed from these persistence diagrams by generating histograms for the pairs and increasing the persistence lifetime threshold if any of the histogram bars had a count larger than the number of strikes in the image. This method is reliant on the observation that a large number of points are present in the low lifetime region of the persistence diagrams. We also filter by the birth times of the features by removing features with the largest birth times until the desired number remain similarly to Fig. 4.11.

Applying these operations to the diagrams in Fig. 4.15, the filtered persistence diagrams in Fig. 4.16 were obtained. The corresponding computed depth scores are shown in Table 4.2, and it was clear that the 25% and 50% overlap images had significantly higher depth scores which could be a result of the strikes being closer together.

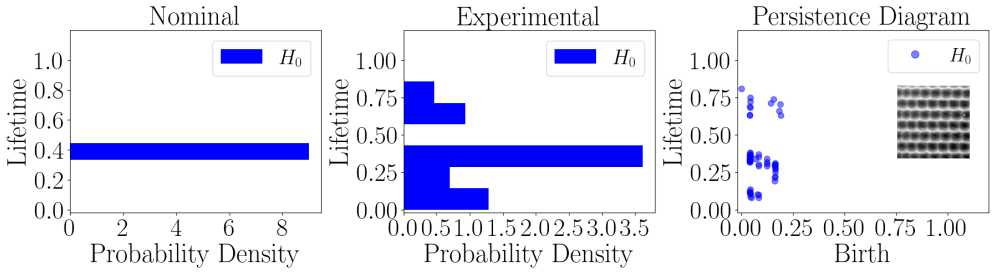


(a) 0% Overlap (b) 25% Overlap (c) 50% Overlap

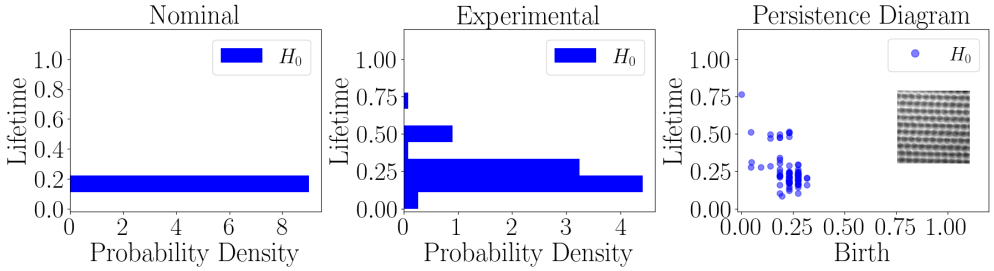
Figure 4.15 Unaltered striking **depth** persistence diagrams (a) 0% Overlap PVST Image, (b) 25% Overlap PVST Image, (c) 50% Overlap PVST Image.



(a) 0% Overlap



(b) 25% Overlap



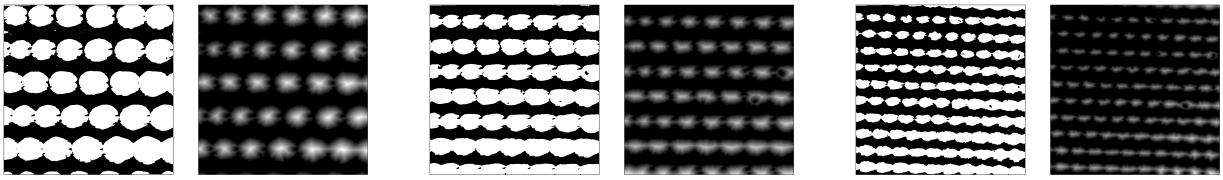
(c) 50% Overlap

Figure 4.16 Noise filtered **depth** persistence diagrams and histograms for each overlap ratio.

4.1.3.2 Strike Roundness Results

Experimental images were thresholded and distance transformed at 50 heights ranging from 0 to 1 in the image and sublevel persistence was computed at each height. Figure 4.17 shows the thresholded and distance transformed images at various heights as an example.

The persistence lifetime histograms were used to compute the earth movers distance between the nominal and experimental distributions which provided a score at each height in the image and therefore information about the roundness over the entire depth of the strikes. Thresholding was started at the reference point ($T = 0$) found from the filtered persistence diagrams. Histograms such as the ones in Fig. 4.18 were generated at each height to visually compare the theoretical distribution of persistence lifetimes to the experimental distribution. This process resulted in an



(a) 0% Overlap (b) 25% Overlap (c) 50% Overlap

Figure 4.17 Example binarized and distance transformed images for each overlap ratio. (a) shows the 0% overlap image thresholded at a height of 0.47, (b) is the 25% overlap image thresholded at 0.47 and (c) shows the 50% overlap image thresholded at 0.51. Note: Binarized images are shown on the left and distance transformed images are shown on the right.

earth movers distance distribution with respect to threshold height as shown in Fig. 4.19. We expect the experimental distributions to be identical to the theoretical distributions and therefore have an earth movers distance of 0 at each height. Any deviation from 0 indicates a change in the uniformity of the roundness. The generalized roundness score was computed for each image by taking the area under the curves in Fig. 4.19. Qualitatively, we see that the 0% image has the most deviation in the roundness when compared to the theoretical model due to its larger area under the earth movers distance curve. Similarly, the 50% overlap image has the most consistent roundness due to its smaller area. To truly compare these plots, the scores need to be computed because the domain for each overlap ratio was different. The roundness scores were computed using Eq. (4.23) because the strikes were nominally spherical. The computed scores are shown in Table 4.3 where a higher score corresponds to the roundness distribution being closer to nominal.

Table 4.3 Computed roundness scores for each overlap ratio. Note that a higher score corresponds to the texture being closer to nominal.

Overlap Ratio	0%	25%	50%
$\bar{R}$	30.82%	70.02%	74.26%

As expected, the 50% overlap image showed the highest roundness score indicating that this image had a more uniform roundness distribution, and the 0% image had the lowest roundness score meaning it had the most deviation from nominal.

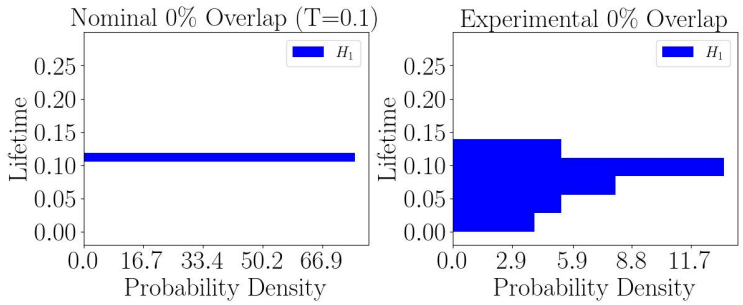


Figure 4.18 Roundness lifetime histogram example at a threshold height of 0.1 from the 0% overlap image.

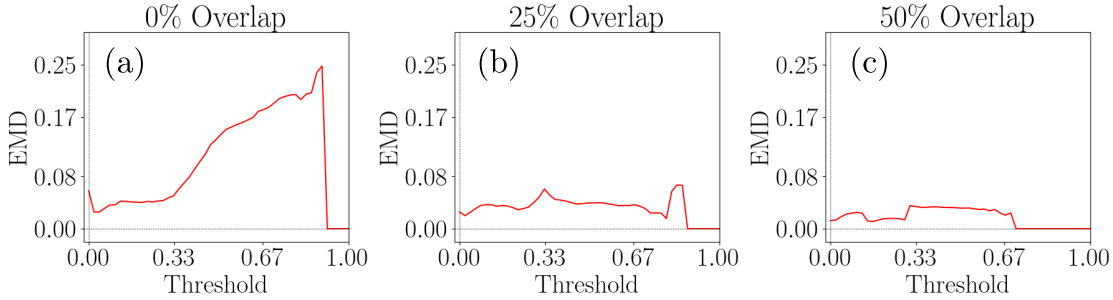


Figure 4.19 Earth movers distance between the experimental and theoretical roundness lifetime distributions as a function of threshold height for each overlap ratio.

4.1.4 Analysis Software

The software used for this analysis was implemented as a texture analysis module in the *teaspoon* python library for topological signal processing [157]. This code provides functions for computing the depth and roundness scores between two input images and the user is responsible for supplying the nominal and experimental image arrays. The teaspoon functions utilize cubical ripser for sublevel persistence computations [151].

To generate the nominal images, the process presented in Section A was followed using SolidWorks to model the surfaces and the algorithm in [158] to convert the CAD model to a point cloud that could be converted to an image.

4.1.5 Conclusion

A novel approach to texture analysis was developed to describe specified features in a texture using topological data analysis. The tools were presented as an application to the surface treat-

ment process piezo vibration striking treatment (PVST) in which a metal surface is impacted in a regular pattern by a tool on a CNC machine leaving a texture on the surface. Strike depth and roundness were successfully characterized using sublevel persistent homology, and scores were devised to quantify the features in the textural images relative to the nominal texture. In general, the higher overlap ratio images were found to provide more consistent strikes which could be due to the higher density of impacts on the surface. Two methods were also presented for generalizing the application of PVST of which the authors recommend using the CAD model method for an arbitrary tool shape. These tools allow for engineers to quantify specific features in a texture, a process which has typically been conducted qualitatively by manual inspection in the past. The scores obtained for depth and roundness features can be used to measure the effectiveness of the process that produced the pattern, and in future work, we plan to utilize these scores for extracting information on the material properties of the work piece.

4.2 Characterizing Pattern Shape

Surface texture influences wear and tribological properties of manufactured parts, and it plays a critical role in end-user products. Therefore, quantifying the order or structure of a manufactured surface provides important information on the quality and life expectancy of the product. Although texture can be intentionally introduced to enhance aesthetics or to satisfy a design function, sometimes it is an inevitable byproduct of surface treatment processes such as Piezo Vibration Striking Treatment (PVST). Measures of order for surfaces have been characterized using statistical, spectral, and geometric approaches. For nearly hexagonal lattices, topological tools have also been used to measure the surface order. This paper utilizes tools from Topological Data Analysis for quantifying the impact centers' pattern in PVST. We compute measures of order based on optical digital microscope images of surfaces treated using PVST. These measures are applied to the grid obtained from estimating the centers of tool impacts, and they quantify the grid's deviations from the nominal one. Our results show that TDA provides a convenient framework for the characterization of pattern type that bypasses some limitations of existing tools such as difficult manual processing of the data and the need for an expert user to analyze and interpret the surface images.

4.2.1 Introduction

One of the main objectives of the manufacturing enterprise is to achieve products that satisfy a preset quality under the constraints of time, cost, and available machines [159]. A key quality in manufacturing is the surface texture which is directly related to surface roughness [160–162] and to the tactile feel of the resulting products which is quantified by tactile roughness [163,164]. Surface texture can be either intentionally introduced to satisfy functional or aesthetic surface properties, or it can be a byproduct of a specific manufacturing setting. However, the importance of surface texture goes beyond merely the aesthetics since the resulting surface properties have a strong influence on ease of assembly, wear, lubrication, corrosion [105], and fatigue resistance [106–108].

An example of a process where surface texture is introduced in order to both enhance the mechanical properties of the part and as an inevitable byproduct is the Piezo Vibration Striking Treatment (PVST) [144]. In PVST, a tool is used to impact the surface and a scanning strategy is applied to treat the whole surface, see Fig. 4.20. Depending on the impact depth and speeds set for the process, various grid sizes and diameters can be obtained. By varying parameters such as the scanning speed and the overlap of the impacts, a texture inevitably is left behind on the surface. While this can be leveraged to both treat and texture the surface, the resulting pattern can also provide invaluable information about the success of the treatment such as quantifying missed or misplaced impact events. Further, the texture can also be utilized to assess the quality of the machined surface through comparing the resulting pattern with the nominal or desired pattern. Specifically, if the resulting pattern is missing too many features (indentations), this can be an indication of large deviation of material distribution on the surface. Detecting such events can signal the need for further finishing, or for adjusting the manufacturing process to enhance the resulting surfaces.

While there are several classical tools for quantifying surface texture [165], one limitation of these methods is their strong reliance on the user for tuning the needed parameters. For example, a common pre-requisite for these tools is knowing the relative pixel intensity in the image and which threshold size for removing objects from the image will result in a successful texture seg-

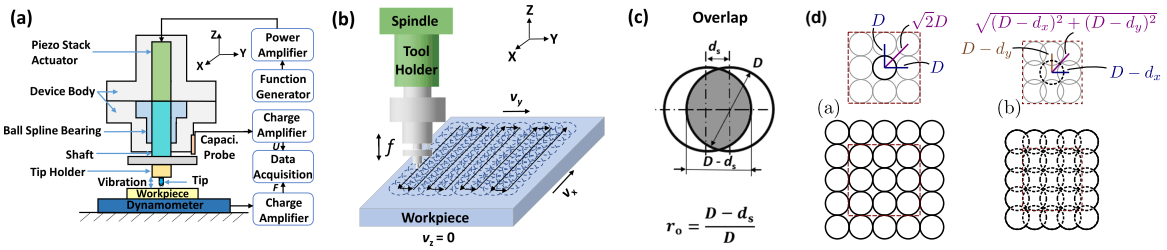


Figure 4.20 Schematics of PVST: (a) PVST; (b) Striking process in PVST; (c) Overlap of indentations; (d) Grid geometry. PVST nominal grid for (left) x and y overlap $d_x = d_y = 0$, and (right) for x overlap $d_x \geq 0$ and y overlap $d_y \geq 0$. The striking tool diameter is D, and the figure shows a scanning strategy where the x and y scanning speeds are set to obtain a certain overlap ratio. When the overlap in x and y is the same, we write $d_x = d_y = d_s$.

mentation. An emerging tool that has shown promise for quantifying texture is from the field of Topological Data Analysis (TDA). More specifically, topological measures were used to quantify order of nearly hexagonal lattices [149] which represent nanoscale pattern formation on a solid surface that can result from broad ion beam erosion [166]. The input data in [149] was the location of the nanodots on the surface, which is an input data type often referred to as a point cloud. Topological measures were shown to be more sensitive than traditional tools, and they can provide insight into the generating manufacturing process by examining the resulting surfaces.

This paper explores utilizing topological measures for quantifying the surface texture produced by PVST. Specifically, the goal of this paper is to use topological methods to quantify lattice types in a PVST image, which can provide insight into the effectiveness of the PVST process. While we use measures inspired by those utilized on point clouds, i.e., point locations in the plane, in [149], the data in our work are images of the resulting surface obtained using KEYENCE Digital Microscope. Therefore, in our setting we need to process the data to extract the PVST indentation centers in order to quantify the resulting pattern. Our exploratory results show possible advantages to our approach including automation potential in contrast to standard tools where intensive user-input is required.

The paper is organized as follows. Section 4.2.2 provides background for PVST. Section 4.2.3 explains the experimental setup and how the experimental data is collected. Section 4.2.4 outlines the processing performed on each image to obtain the point cloud data. Section 4.2.5 discusses the

TDA-based approach proposed in this study. Section 4.2.6 compares the results of the analysis to a perfect square lattice. Section 4.2.9 includes the concluding remarks.

4.2.2 Piezo Vibration Striking Treatment

Mechanical surface treatment uses plastic deformation to improve surface attributes of metal components, such as surface finish, hardness, and residual stress, which is an effective and economical way of enhancing the mechanical properties of engineering components. Among various mechanical surface treatment processes, i.e., Shot Peening [167], Surface Mechanical Attrition Treatment [168], High-frequency Mechanical Impact Treatment [169], and Ultrasonic Nanocrystal Surface Modification [170], PVST is a novel mechanical surface treatment process that is realized by a piezo stack actuated vibration device integrated onto a computer numerical control (CNC) machine to impose tool strikes on the surface. Different from those processes, the non-resonant mode piezo vibration in PVST and the integration with CNC machine enable PVST to control the process more conveniently and precisely as demonstrated in previous applications in modulation-assisted turning and drilling processes [171, 172]. The schematics of PVST are shown in Fig. 4.20. The device is connected to the spindle of a CNC mill through a tool holder. The spindle can only move along the Z direction to control the distance between striking tool and workpiece surface. The motion of the machine table along the X or Y directions defines specific striking locations on the workpiece mounted on the table. As shown in Fig. 4.20a, the piezo stack actuator is connected with a spline shaft that is part of the ball spline bearing, both of which are fixed in the device body. The actuator drives the shaft to move along the Z direction, but no bending or rotation is allowed. The striking tool is rigidly connected to the shaft through a holder. The power generator and amplifier produce amplified driving voltage to extend and contract the actuator and hence actuate the tool to oscillate along the axial direction. A capacitance probe clamped onto the device body and a dynamometer plate mounted on the machine table are used to measure the displacement of the tool and the force during the treatment. Both the force and displacement are recorded synchronously in a data acquisition system. The lower bound and upper bound of the driving voltage are set as zero and peak-to-peak amplitude V_{pp} of the voltage oscillation, which can control the frequency

and amplitude of the tool vibration to generate different surface textures. The initial position of the tool Z can be used to control the distance between the tool and the workpiece surface and hence change the striking depth to produce different surface textures as well. As shown in Fig. 4.20b-d, the successive strikes controlled by scan speed v_s will be imposed on different locations of the surface along the tool scan path. The offset distance d_s between two successive strikes and the diameter of the indentation can be utilized to compute the overlap ratio. Different overlap ratios can generate various surface textures, namely higher overlap ratio leads to a denser distribution of the indentations. This paper focuses on the low overlap ratio images produced using PVST to detect the center points of the circles in the image and quantitatively determine the lattice type present in the texture with minimal user input.

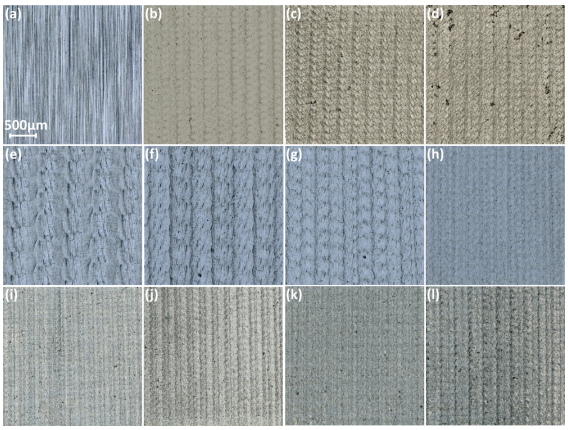


Figure 4.21 Various surface textures under different PVST conditions: (a) initial workpiece surface; (b) — (d) different Z values; (e) — (h) different overlap ratios; (i) — (l) different driving voltages.

4.2.3 Experimental Procedure

A mild steel ASTM A572GR50 workpiece with a dimension of 120 mm × 40 mm × 20 mm is used for surface texture data collection under various PVST conditions (see Tab. 4.4).

Table 4.4 Various PVST conditions. The first column represents the samples in Fig. 4.21.

No.	f (Hz)	V_{pp} (V)	d (mm)	r_o	Z (μm)
b-d	100	120	3	0.75	0,10,20
e-h	100	120	3	0, 0.25, 0.5, 0.75	0
i-l	100	60, 90, 120, 150	3	0.75	0

The treated area for each condition is 5 mm × 5 mm. Only a size of 2.5 mm × 2.5 mm is used for surface texture data collection due to the duplicate characteristic of the surface texture throughout the treated area and the computation efficiency for data processing. The selected area is fixed at the upper left corner of the treated area for consistent data collection. The workpiece is placed on a free-angle XYZ motorized observation system (VHX-S650E), and 3D surface profiles are characterized using KEYENCE Digital Microscope (VHX6000), as shown in Fig. 4.22a. A real zoom lens (KEYENCE VH-Z500R, RZ x500 — x5000) and x1000 magnification are utilized to achieve sufficient spatial resolution (0.21 μm). Since each capture under this magnification can only cover a small area, the stitching technique (22×22 scans in horizontal and vertical directions) is employed to achieve sufficient capture field. Fig. 4.22b shows one of the captured 3D profiles under two different types of illustrations (texture and surface height map). The scanned surface textures after different PVST conditions are shown in Fig. 4.21. Note that for this paper, images e, f, and g were the main focus because they allow extracting the indentation centers. The centers in the remaining images are not easily identifiable, and they require tools from image analysis that are beyond the scope of this paper.

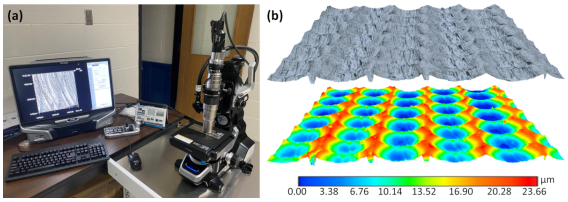


Figure 4.22 (a) KEYENCE digital microscope (b) illustrations of scanned surface texture.

4.2.4 Data Preprocessing

In this section, we explain how we preprocess the data set. The raw images obtained from the microscope have dimension of nearly 12000×12000. However, there are black pixels around the edges of each image. We removed these pixels such that all raw images have a final dimension of 12000×12000 taken from the center of the corresponding raw image. This significantly reduced the number of black pixels in the retained images. Each image was then converted to grayscale as shown in Fig. 4.23.

4.2.4.1 Image Cropping:

Because the methods used to detect the lattice type depend on the number of points in each image, the images needed to be cropped in such a way that nominally, the same number of centers were obtained each time [149]. This was accomplished by using the speed and frequency from the PVST process to compute the expected number of pixels per circle. The highest speed was used to set an upper bound on the number of points per image, and this was held constant among the images. This information was then used to compute the pixel dimensions required to obtain a specified grid in each image and the images were cropped accordingly.

4.2.4.2 Nominal Grid:

The PVST process parameters that were used to generate the surface in each image were then used to plot a nominal (expected) grid on top of the actual image to visualize the difference between the nominal and the resulting grids. The nominal grid was created by first placing a datum point on top of the experimentally found center at the upper left center in the image, then computing the locations of the other points based on the nominal process parameters. An example plot of the nominal grid for the 0% overlap ratio image is shown in Fig. 4.24 as the red triangles. It is clear that the nominal grid is not aligned with the true grid as evidenced by the slight shifts in the rows which causes a change in the lattice type.

4.2.4.3 True Grid:

To quantify the grid resulting from the PVST treatment, the center points of the tool indentations need to be located. This was accomplished by applying a region growing algorithm starting from a manually selected point near the center of the circle to detect a bounding polygon. The centroid of the generated bounding polygon was then used as an estimate of the center location as shown in Fig. 4.24. The reason for manually choosing a guess for the centers' locations, instead of using the nominal locations, is shown in Fig. 4.22b. The figure shows that although the overlap was selected to be equal in the horizontal and the vertical direction, the columns are more widely spaced in the horizontal direction. We hypothesize that this is the result of the CNC motion whose acceleration is more smoothly varied in the direction of the scan (the vertical direction in the

Fig. 4.22b) thus producing more precise impact locations in that direction. In contrast, we suspect that the rapid positioning motions of the CNC when moving to the next column in Fig. 4.22b are creating a drift in the center locations along that column that propagates every time a new column is treated which leads to incrementally shifting the whole pattern in the horizontal direction.

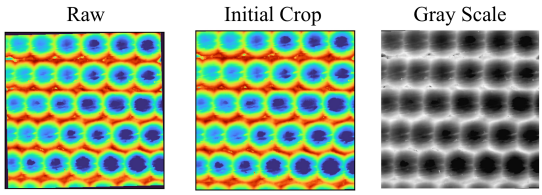


Figure 4.23 Preprocessing of a sample raw image.

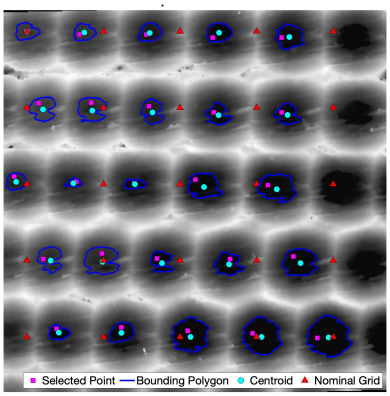


Figure 4.24 Locating Nominal and True Lattice Centers.

4.2.5 Topological Data Analysis Based Approach

In this section, we give a brief description of persistent homology, a tool from Topological Data Analysis (TDA), specifically as it applies to point clouds since that is what we use to characterize the PVST grid. We refer the interested reader to more in-depth treatment of TDA in [29, 31, 33, 34, 173, 174].

We then summarize the information extracted from images in persistence diagrams, and we use the latter to score the PVST-treated surfaces.

4.2.5.1 Persistent homology

Persistent homology, or persistence, is a tool from TDA for extracting geometric features of a point cloud such as the connectivity or the number of holes in the space as a function of a connectivity parameter. Specifically, consider the point cloud shown in Fig. 4.25a which represent points in an almost prefect square lattice but with three perturbed points. Suppose that we start expanding disks of diameter d around each of the points, and we monitor changes in the connectivity of the components as d is increased.

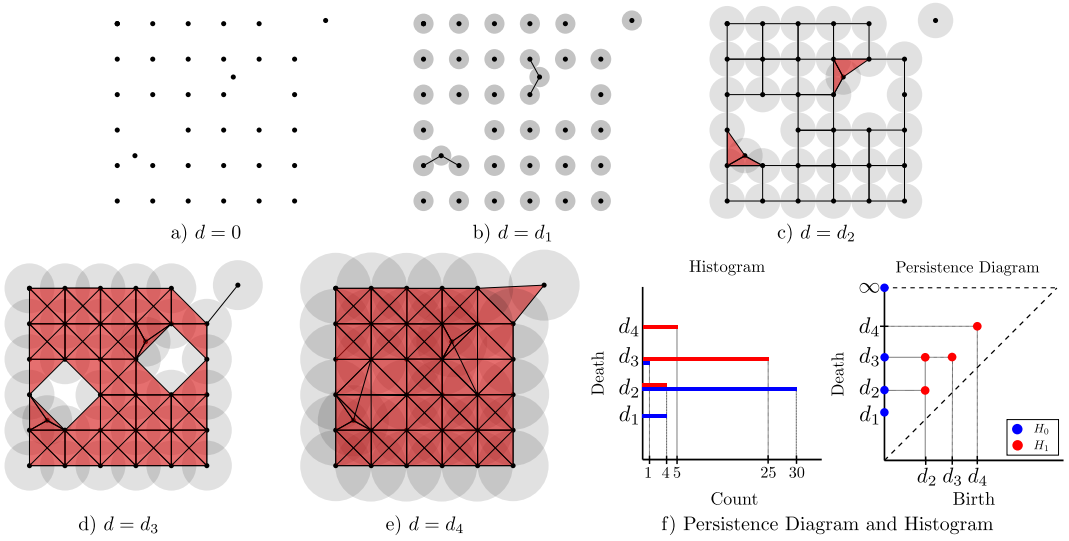


Figure 4.25 Point Cloud Persistence Example.

We define connectivity using the Euclidean distance in the plane between each point in the set. Specifically, we connect two vertices via an edge if their Euclidean distance is at least equal to the connectivity parameter d. While Fig. 4.25a shows that for $d = 0$ we have 36 distinct components that emerge or are *born*, Fig. 4.25b demonstrates that some of these components begin to merge or *die* at $d = d_1$. When $d = d_2$, the square edges connect which creates holes at the center of each square and two larger holes where the perturbations are present. Note that the third perturbation in the corner remains disconnected at this point. Once the connectivity parameter reaches d_3, all of the components remain connected as $d \to \infty$. The difference between the death value and the birth value for any component is called its *the lifetime*.

We can summarize the connectivity information using the zero-dimensional (0D) persistence diagram shown in Fig. 4.25f (blue). The coordinates of points in this diagram are the (birth, death) values of the connectivity parameter where the number of connected components changes. This diagram shows that four of the components die when $d = d_1$ and 30 more connect at $d = d_2$ with the remaining point joining at d_3. These quantities are represented in the histogram shown with the persistence diagram.

Moreover, Fig. 4.25 shows that the grid has interesting geometric features characterized by the emergence and disappearance of holes in the plane as d is varied. Persistence can also track the birth and death of these holes via one-dimensional (1D) persistence. In order to track holes, for each value of d we construct a geometric object that includes the vertices themselves, pairs of points (edges) that are connected at d, and the 3-tuples that represent faces (geometrically, they are triangles) which get added whenever all their bounding edges are connected. The geometric objects constructed this way are called simplicial complexes, and varying d leads to a growing sequence of them indexed by the single parameter d. For example, Fig. 4.25f (red) shows that at $d = 0$ only the vertices are included in the corresponding simplicial complex, and we have no holes. Increasing d to d_1 in Fig. 4.25b, edges are included between points whose Euclidean distance is less than or equal to d. We also fill in or include in the simplicial complex any triangles whose bounding edges all are included. This is shown in Fig. 4.25c where four triangles are added. At this point, when $d = d_2$, 18 loops were born that were not filled in immediately. As d is increased to d_4, more holes emerge and are filled at the same time. This process is continued until all of the components are connected as one and no loops persist in the simplicial complex. The information related to the birth and death of holes is summarized in the 1D persistence diagram shown in Fig. 4.25f. Notice that two of the holes born at d_2 are larger than the others causing them to have a longer lifetime as d is varied. These larger holes are apparent in the persistence diagram with the largest vertical distance to the diagonal.

The combination of the 0D and 1D persistence allows us to quantify the deviation of a given grid from its nominal, perfectly ordered lattice. For example, in a PVST process we can compute

the 0D and 1D persistence of the resulting, actual grid by locating the centers of the tool on the treated surface. We can then compare the actual persistence values to their nominal counterparts where the latter are obtained by assuming a perfectly produced lattice with no defects or center deviations. This allows us to quantify the proximity of the resulting grid to the commanded grid, and gives us a tool to characterize the defects during the PVST treatment process such as misplaced or missed strikes that alter the expected pattern.

4.2.5.2 TDA-based scores:

We focus on 0D (H_0) and 1D (H_1) persistence to obtain scores for texture analysis. In order to quantify the type of the pattern on a PVST-treated surface, a method is needed for scoring different lattices. To achieve this, point cloud persistent homology was applied to a perfect square lattice, and expressions were obtained to be used for comparison with the persistence outputs from the actual surfaces. Comparing the results from a perfect lattice to the true surface of interest allows for conclusions to be drawn about the type of lattice present due to PVST. To correctly compare the results from multiple images, the same number of points must be chosen in each case and the square regions containing these points must be similarly scaled, e.g., to [-1,1]×[-1,1]. Otherwise, the comparisons become less meaningful because mismatched sample scaling or different number of points in each sample will strongly influence the resulting scores. **0-D Persistence:** To compute the H_0 persistence of a perfect square lattice, a Vietoris–Rips complex was applied to a perfect square lattice with an $n \times n$ grid of points in Fig. 4.26 and the connectivity parameter d was varied until the all of the rectangles were included in the complex.

For the H_0 persistence, points were born at $d = 0$ and died at $d = \frac{2}{n-1}$ where d is the diameter of the expanding balls. For 0D persistence of this lattice, all of the elements are born and die at the same time. It was shown in [149] that for both perfect square and perfect hexagonal lattices, the variance in the 0D persistence (lifetimes) is zero, which we express as

$$Var(H_0) = 0. \tag{4.25}$$

This expression can be used for measuring the deviation from a square/hexagonal lattice in the presence of a non-zero variance [149]. Note that the overlap ratios for the PVST images are

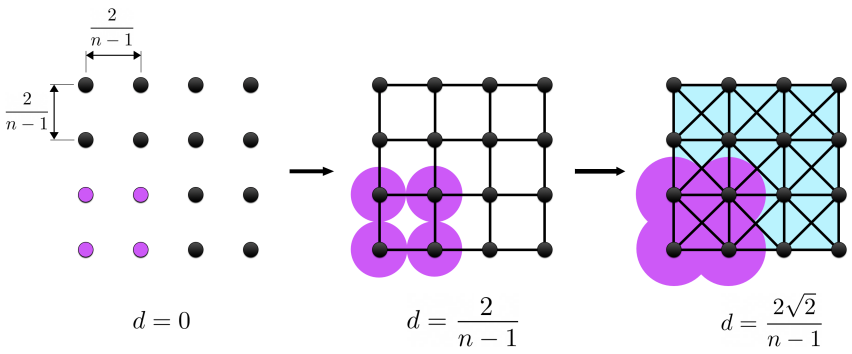


Figure 4.26 An illustration of the key values of the diameter d at which features are born and die for a perfect Square Lattice. The expanding disks (purple) are only shown for four points to better visualize the edges and the triangles that get added at each shown d value.

accounted for when the nominal distance between points is computed using the in-plane speeds and frequency to locate the centroids of the circles. The theoretical persistence diagram for the 0-D persistence was generated as shown in Fig. 4.27.

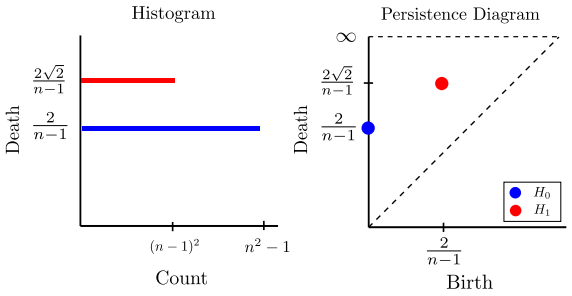


Figure 4.27 Theoretical Persistence Diagram for a perfect square lattice.

Where n^2-1 components are born at 0 and die at $\frac{2}{n-1}$, and one object survives for all time. The H_0 variance was determined to have a maximum value of $\frac{1}{4}$ over all possible lattice types [149] so to normalize this measure, it was multiplied by a factor of 4. **1-D Persistence:** For the 1-D persistence in a Vietoris–Rips complex, the presence of loops is of interest. Figure 4.26 shows that all of the loops are born at $d = \frac{2}{n-1}$ and die at $d = \frac{2\sqrt{2}}{n-1}$ giving the following lifetime for each individual loop

$$H_1 = \frac{2\sqrt{2}}{n-1} - \frac{2}{n-1}. \tag{4.26}$$

If the grid of interest is $n \times n$ points, the total number of loops k present for a perfect rectangular

lattice is $k = (n-1)^2$. The loop lifetimes provide insight into the density of the lattice in the image as a higher point density would have smaller loops. To incorporate all of the loop lifetimes into one measure, the sum of all individual lifetimes is computed. For a perfect lattice, the sum can be multiplied by the number of loops because in a perfect lattice all of the loops have the same lifetime. Therefore, Eq. (4.27) can be used to quantify the type of lattice in the image where it is a maximum value for a square lattice and 0 for a hexagonal lattice [149].

$$\sum H_1 = 2(\sqrt{2}-1)(n-1). \tag{4.27}$$

The 1-D persistence diagram for the rectangular lattice is shown in Fig. 4.27. For the perfect lattice, k loops are born at $\frac{2}{n-1}$ and die at $\frac{2\sqrt{2}}{n-1}$. If the H_1 lifetimes are smaller than the perfect lattice lifetime, this would indicate the presence of shifts in the lattice type (i.e., some of the points are shifted allowing some of the loops to prematurely close). It has been shown that this measure is equal to zero for a perfect hexagonal lattice and achieves a maximum value for a square lattice [149]. In our case, the normalization factor was a function of the number of points in a row or column of the grid as shown previously. For this reason, the sum of 1D persistence lifetimes was divided by the expression shown in Eq. (4.27). Because the H_1 sum is nonzero for a square lattice, the CPH score approach presented in [149] for hexagonal lattices cannot be used due to the underlying nominal lattice being square, and a square lattice cannot be detected with a single measure of order. Together, the H_0 and H_1 measures were used to provide scores for classifying the type of lattice. For example, if both scores are close to zero the lattice is mostly hexagonal. If the H_1 sum is close to 1 and the H_0 variance is small, the lattice is mostly square and if both measures are close to one the lattice is neither square nor hexagonal. Expressions for the computed scores used with the PVST centers point clouds are given by

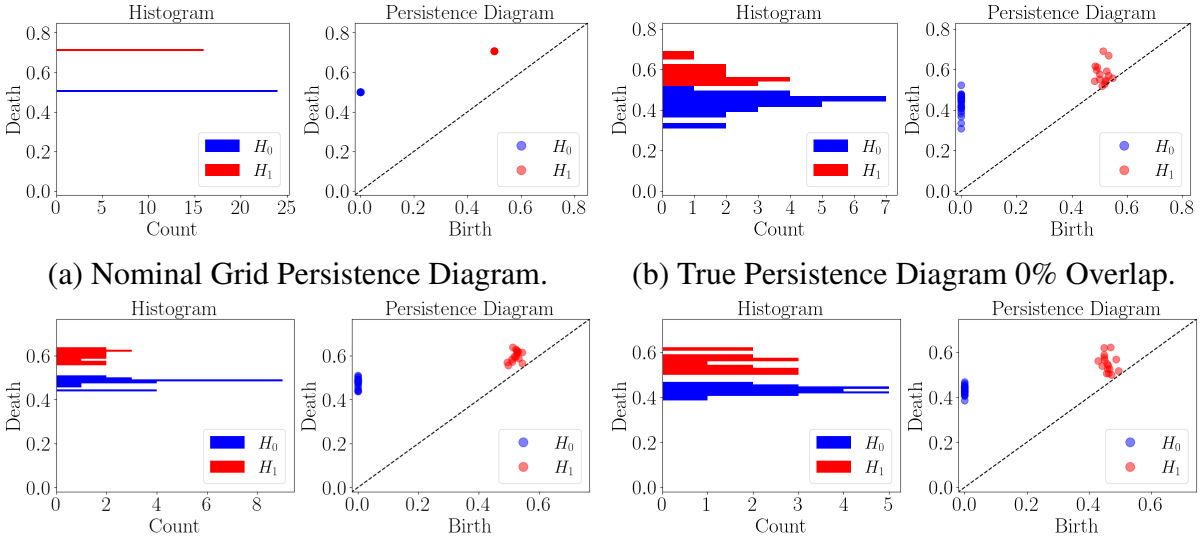
$$\overline{H_0} = 4Var(H_0), \tag{4.28a}$$

$$\overline{H_1} = \frac{1}{2(\sqrt{2}-1)(n-1)}\Sigma H_1, \tag{4.28b}$$

where $\overline{H_0}$ is the normalized 0-D persistence score, and $\overline{H_1}$ is the normalized 1-D persistence score. These scores were used to detect the presence of square and hexagonal lattices in the PVST images and quantify the relative magnitudes of each type.

4.2.6 Results

Point cloud persistence was applied to the nominal and actual grids from the PVST images and the corresponding persistence diagrams and histograms were generated. The scores from Section 4.2.5.2 were then used with the computed statistics from the persistence output to quantify the lattice types.



(a) Nominal Grid Persistence Diagram.

(b) True Persistence Diagram 0% Overlap.

(c) True Persistence Diagram 25% Overlap.

(d) True Persistence Diagram 50% Overlap.

Figure 4.28 The resulting persistence diagrams and the corresponding histograms for (a) a nominal grid, (b) PVST with 0% overlap, (c) PVST with 25% overlap, (d) PVST with 50% overlap.

4.2.7 Persistence Diagrams

The generated point clouds from Section 4.2.4 were passed into Ripser, a python library used to generate persistence diagrams from point cloud data with a Vietoris-Rips Complex [175]. First, nominally generated grid point clouds were passed to Ripser to verify the expressions obtained in Section 4.2.5.2. The same persistence diagram resulted for all three of the grids shown in Fig. 4.28a, because the data was scaled to the same region. Histograms were plotted beside the

persistence diagrams to illustrate repeated points in the diagram. We see that the 0D persistence pairs have a birth time of 0 and a death time at 0.5 which is consistent with the predicted results from Fig. 4.26 when $n = 5$. We then plotted the persistence diagrams for the actual grids obtained from the region growing algorithm. These persistence diagrams are shown in Fig. 4.28b—d. It was clear from the 0% overlap ratio image, the point density was larger than expected which was captured by the 1D persistence diagram loops having a lower lifetime as a result of the centers being closer together in this image. The 0D persistence pairs in the 0% overlap image appear to have a significantly larger spread when compared to the other overlap ratio results. This was likely due to some of the rows shifting into a hexagonal lattice causing some of the components to connect before they would if the lattice were square.

4.2.8 Measures of Order

Equations (4.28a) and (4.28b) were used to compute measures of order based on the experimental persistence information generated from the point clouds. The computed measures of order for each image are shown in Table 4.5. The $\overline{H_0}$ scores were all significantly smaller than 1 indicating that all of the PVST tests considered in this paper produced lattices that were somewhere between square and hexagonal. With the information from the 0D persistence score, the $\overline{H_1}$ score allows for the lattice type to be located on the lattice figures in [149] showing that the 0% overlap ratio image was the closest to a hexagonal lattice with the larger overlap images corresponding to images closer to square relative to the first image. It should be noted that all of the $\overline{H_1}$ scores were below 0.5 which meant that the resulting lattices were predominantly closer to hexagonal than square. Another way to interpret this measure is to treat it as a percentage of square lattice present in the image for $\overline{H_0} \approx 0$. With this interpretation, the images had 31.4%, 37.9% and 44.1% square lattice respectively with the remaining proportion being hexagonal.

Table 4.5 Measures of Order for the experimental data.

Image	0% Overlap	25% Overlap	50% Overlap
$\overline{H_0}$	2.29e−3	0.405e−3	0.337e−3
$\overline{H_1}$	0.314	0.379	0.441

4.2.9 Conclusion

Topological approaches were applied to characterizing texture of images obtained from PVST-treated surfaces to determine the underlying lattice type. Our exploratory results show that using TDA for surface texture characterization can be beneficial for quantifying the lattice shape obtained from a PVST sample. Scores used in this paper allowed for direct quantification of the proportions of different lattice types present in a PVST surface which agreed with the qualitative examination of the images. The information gained from applying this analysis removes ambiguity in determining the true lattice shape and can be used for gaining insight into process control, and gauging improvement in the regularity of the PVST process.

The authors plan to continue work on this topic in the future to further automate the process for detecting the true PVST centers. Future work also includes expanding the analysis of PVST surfaces through quantifying the roundness of the resulting tool indentations at different level sets, and examining the consistency of the the striking depths.

CHAPTER 5

MATHEMATICAL MODELING

This chapter presents my work on a time delay model for metabolic oscillations in yeast cells published in [9], reproduced with permission from Springer Nature. When cells are starved of resources, it has been observed experimentally that the protein production rates will oscillate in approximately 40 minute intervals. I developed a time delay framework for modeling metabolic oscillations in yeast cells and analyzed the model using three numerical approaches to find parameters that resulted in a limit cycle. I also extended the model to include three coupled proteins and used the same methods for analysis.

5.1 A Nonlinear Delay Model for Metabolic Oscillations in Yeast Cells

We introduce two time-delay models of metabolic oscillations in yeast cells. Our model tests a hypothesis that the oscillations occur as multiple pathways share a limited resource which we equate to the number of available ribosomes. We initially explore a single-protein model with a constraint equation governing the total resource available to the cell. The model is then extended to include three proteins that share a resource pool. Three approaches are considered at constant delay to numerically detect oscillations. First, we use a spectral element method to approximate the system as a discrete map and evaluate the stability of the linearized system about its equilibria by examining its eigenvalues. For the second method, we plot amplitudes of the simulation trajectories in 2D projections of the parameter space. We use a history function that is consistent with published experimental results to obtain metabolic oscillations. Finally, the spectral element method is used to convert the system to a boundary value problem whose solutions correspond to approximate periodic solutions of the system. Our results show that certain combinations of total resource available and the time delay, lead to oscillations. We observe that an oscillation region in the parameter space is between regions admitting steady states that correspond to zero and constant production. Similar behavior is found with the three-protein model where all proteins require the same production time. However, a shift in the protein production rates peaks occurs for low available resource suggesting that our model captures the shared resource pool dynamics.

5.1.1 Introduction

Cellular processes often exhibit non-trivial temporal dynamics in the absence of the external stimulus. Most common is the cell division cycle. However, as observed more than 50 years ago [176], yeast populations in low growth conditions exhibit metabolic cycling (MC) [177, 178] also known as respiratory cycling [179]. While traditionally described as a result of carbon limitation, limitations by other essential nutrients like phosphate [180] or ammonium, ethanol, glucose, and sulfur [181, 182] can lead to MC also known as metabolic oscillations. Under the growth conditions commonly used in this system, the population doubling time and thus the length of the cell division cycle is about 8 h, and the metabolic oscillations have period $40 - 44$ min [183].

The oscillations were first observed as periodic oscillations in the oxygen consumption of continuous, glucose-limited cultures growing in a chemostat, but were later also observed in batch cultures [184]. The MC has two distinct phases: low oxygen consumption (LOC) phase when dissolved oxygen in the medium is high and high oxygen consumption phase (HOC) when the oxygen in the medium drops to low levels [177, 178, 183]. Using experimental techniques ranging from micorarray analysis [177, 183] to short-life luciferase fluorescent reporters [179], researchers were able to assign transcription of particular genes to these phases. During the LOC phase the yeast culture performs oxidative metabolism focused on amino-acid and ribosome synthesis, while during the HOC phase reductive reactions including DNA replication and proteosome related reactions occur [177]. Cellular metabolism during the reductive HOC phase seems to be devoted to the production of acetyl-CoA, preparing cells for the upcoming oxidative phase, during which metabolism shifts to respiration as accumulated acetyl-CoA units for ATP production via the TCA cycle and the electron transport chain [177].

This compartmentalization of cellular processes in time is thought to be related to help assembly of macro-molecular complexes from units that, at low growth rates, are expressed at very low levels. Expressing them at the same time helps ensure timely synthesis and avoids waste of limited resources [178].

There were many different hypotheses centered on chemical signals that may mediate metabolic

synchrony. In particular, Murray et. al. [185] proposed acetaldehyde and sulphate, Henson [186] and Sohn and Kuriyama [187] hydrogen sulphide, while Adams et. al. [188]found that Gts1 protein plays a key stabilizing role. Finally, Muller at. al. [189] suggest a signalling agent, cAMP, plays a major role in mediating the integration of energy metabolism and cell cycle progression.

Several mathematical models that do not specify the synchronizing chemical agent, but explore a general idea that cells in one phase of a cell cycle can slow down, or speed up progression of other cells through a different phase, have been suggested [190, 191]. Finally, paper [192] explores synchronization which is a result of criticality of necessary cellular resources combined with the engagement of a cell cycle checkpoint, when these resources dip below the required level.

In this paper we explore the hypothesis that oscillations may arise spontaneously when several cellular processes share a limited resource. This does not explain why the processes separate into oxidative and reductive phase but argues that compartmentalization in time may help utilize limited resources more efficiently. Ribosomes are essential cellular resources as they produce enzymes used in all metabolic processes as well as all other proteins including those used to assemble ribosomes themselves. Yeast ribosomes are large molecular machines consisting of 79 proteins [193] and therefore they require substantial investment of cellular resources. This is reflected in the observation that the ratio of ribosomal proteins to all proteins scales linearly with cell growth rate across metabolic conditions [194]. For this reason in our model we equate the limited shared cellular resource to the number of available ribosomes.

In many biological systems, time delays are often incorporated into the models because many of these processes have nontrivial time spans that dictate the overall system behavior [195–202]. For this reason, the time delay framework is ideal for modeling a metabolic process where the protein production times can take upwards of 40 minutes. We aim to model the protein synthesis process in yeast cells using time delays and explore under what conditions oscillations are present in the responses. This paper is structured as follows. In Section 5.1.2 we introduce a single protein model and extract theoretical results such as the fixed points and its linear stability behavior. Section 5.1.3 presents the three protein extension to the single protein model and the equilibrium conditions are

derived along with the system linearization. We then show the numerical methods that are utilized on the models in Section 5.1.4 where we describe the spectral element linear stability method, response feature analysis of system simulations under low growth conditions, and boundary value problem computation of periodic solutions to the nonlinear systems from simulation data. Results for the single protein system are then presented in Section 5.1.5 where the numerical methods are applied and the stability of the system is characterized in a subset of the overall parameter space. We then apply the same methods to the three protein system in Section 5.1.6. Finally, we give concluding remarks in Section 5.1.7.

5.1.2 Theory — Single Protein

5.1.2.1 Model Derivation

Both transcription and translation involve processing molecules (RNAP, ribosomes) that are sequestered during the time of processing. These processing molecules constitute cellular resources that need to be shared by all necessary protein production processes. We will concentrate here on ribosomes, as their concentration is known to be tightly correlated with the microbial growth rate [194]. The rate of production of protein $p(t)$ is proportional to the rate of initiation μ at some time $t-\tau(t)$ in the past when the processing started

$$\dot{p}(t) = B\mu(t-\tau) - Dp(t), \tag{5.1}$$

with the maximal growth rate B and the decay rate D. The rate of initiation $\mu(t)$ is a product of the activator (which we assume for simplicity is the protein p itself) and the ribosome R:

$$\mu(t) = f(p(t))R(t), \tag{5.2}$$

with Hill function

$$f(t) = \frac{p^n(t)}{\kappa^n + p^n(t)}.$$

A suggestion for the sequestration equation based on [203] is given by

$$R(t) = R_T - A\int_{t-\tau}^{t} \mu(s)ds, \tag{5.3}$$

where R_T is the total resource (ribosomes) and the integral is the resource which is currently being sequestered to produce a protein. Differentiation of Eq. (5.3) leads to

$$\dot{R}(t) = A\left(\mu(t-\tau) - \mu(t)\right). \tag{5.4}$$

We note that this differentiation step is only valid for constant delays and if variable delays are used, Eq. (5.3) must be used for analysis. Putting the equations together, we have the model

$$\begin{aligned} \dot{p}(t) &= Bf(p(t-\tau))R(t-\tau) - Dp(t), \\ \dot{R}(t) &= A\left(f(p(t-\tau))R(t-\tau) - f(p(t))R(t)\right). \end{aligned} \tag{5.5}$$

The constant total resource R_T is the sum of $R(t)$ and the integral over the history of $\mu(s)$ from $s = t-\tau(t)$ to $s = t$. This means that the initial functions for $p(\theta)$ and $R(\theta)$ with $\theta \in [-\tau(0), 0]$ specify the value R_T.

5.1.2.2 Equilibrium Points

Equilibrium conditions of the system can be obtained by setting $p(t) = p(t-\tau) = p^*$, $R(t) = R(t-\tau) = R^*$, $\dot{p}(t) = 0$, $\dot{R}(t) = 0$ in Eq. (5.5). This process yields one equilibrium condition (Eq. (5.6)) because the equation for $\dot{R}(t)$ in Eq. (5.5) is satisfied for all constant p and R.

$$Dp^* = Bf(p^*)R^* \tag{5.6}$$

The other equilibrium conditions are obtained from Eq. (5.3) by assuming that the integrand is constant when equilibrium has been reached. This yields Eq. (5.7).

$$R^* = \frac{R_T}{1 + A\tau f(p^*)} \tag{5.7}$$

We then substitute Eq. (5.7) into Eq. (5.6) to obtain,

$$p^* = \frac{Bf(p^*)R_T}{D(1 + A\tau f(p^*))}. \tag{5.8}$$

Finally, inserting the definition of $f(p^*)$, we get a polynomial expression that must be satisfied for the equilibrium points as shown in Eq. (5.9).

$$(1 + A\tau)p^{*n+1} - \frac{BR_T}{D}p^{*n} + \kappa^n p^* = 0 \tag{5.9}$$

Any solution to Eq. (5.9) can then be plugged in to Eq. (5.7) to obtain the equilibrium solutions. Note that this polynomial does not have analytical solutions for all values of n, but there is always one trivial equilibrium solution at $(p^*, R^*) = (0, R_T)$. The nontrivial equilibrium points are then obtained from solving the following system for (p^*, R^*).

$$(1 + A\tau)p^{*n} - \frac{BR_T}{D}p^{*n-1} + \kappa^n = 0, \tag{5.10a}$$

$$R^* = \frac{R_T}{1 + A\tau f(p^*)}. \tag{5.10b}$$

Once the trivial solution is removed from the conditions, the remaining polynomial in p^* has either 0 or 2 positive real roots by Descartes' rule of signs for every $n \in \mathbb{Z}^+$ assuming only positive parameters are chosen. In conclusion, there is at least 1 trivial root at $(0, R_T)$ and at most 3 equilibrium points including the trivial point where the other two points are the nontrivial equilibria. We choose to restrict the analysis to $n = 2$ so that we can obtain analytical solutions for the equilibrium points. Solving Eq. (5.9) for $n = 2$ yielded three equilibrium points:

$$(p^*, R^*) = (p_{\text{trivial}}, R_{\text{trivial}}),$$

$$(p^*, R^*) = (p_{\text{middle}}, R_{\text{middle}}),$$

$$(p^*, R^*) = (p_{\text{top}}, R_{\text{top}}),$$

where,

$$p_{\text{trivial}} = 0$$

$$R_{\text{trivial}} = R_T$$

$$p_{\text{middle}} = \frac{BR_T - \sqrt{B^2R_T^2 - 4D^2\kappa^2(A\tau + 1)}}{2D(A\tau + 1)}$$

$$R_{\text{middle}} = \frac{BR_T\sqrt{B^2R_T^2 - 4D^2\kappa^2(A\tau + 1)} - 2AD^2\kappa^2\tau(A\tau + 1) - B^2R_T^2}{B(A\tau + 1)\left(\sqrt{B^2R_T^2 - 4D^2\kappa^2(A\tau + 1)} - BR_T\right)}$$

$$p_{\text{top}} = \frac{BR_T + \sqrt{B^2R_T^2 - 4D^2\kappa^2(A\tau + 1)}}{2D(A\tau + 1)}$$

$$R_{\text{top}} = \frac{BR_T\sqrt{B^2R_T^2 - 4D^2\kappa^2(A\tau+1)} + 2AD^2\kappa^2\tau(A\tau+1) + B^2R_T^2}{B(A\tau+1)\left(\sqrt{B^2R_T^2 - 4D^2\kappa^2(A\tau+1)} + BR_T\right)}$$

$A, B, D, \tau, \kappa,$ and R_T are system parameters. Note that the second equilibrium point has a protein production rate that is between the protein production rates of the other two equilibrium points. Therefore, we refer to this equilibrium point as the "middle" equilibrium point and the point with the largest p^* as the "top" equilibrium point. By studying the stability of these three fixed points, the stability of the system can be characterized for certain parameters.

To understand the role that each equilibrium point plays in the system, we need to understand which equilibrium points are present in different regions of the parameter space. For this system, we can compute the saddle node bifurcation by studying the curve where the argument in the square root term becomes negative indicating that the top and middle equilibria are no longer real numbers. This curve forms a boundary that separates the region with three equilibrium points and the region with only the trivial equilibrium point. It can be computed analytically for this system and is defined in terms of τ and R_T as:

$$R_T < \sqrt{\frac{4D^2\kappa^2(1+A\tau)}{B^2}}. \tag{5.11}$$

This boundary given by the equality of Eq. (5.11) is plotted in the stability diagrams in Sec. 5.1.5 as a red curve with triangles. So any parameters that satisfy (5.11) only have a single equilibrium at the trivial point, and if the parameters do not satisfy this inequality, the top and middle equilibrium points are valid equilibrium solutions along with the trivial point.

5.1.2.3 System Linearization

In the analysis of nonlinear dynamical systems it is useful to study the associated linearized system about the fixed points. The Hartman-Grobman theorem states that near a hyperbolic equilibrium point, the linearized system exhibits the same behavior as the nonlinear system [77]. We linearize (5.5) by computing the Jacobian matrices of the present and delayed states about an equi-

librium point $\vec{q} = \begin{bmatrix} p^* & R^* \end{bmatrix}^T$. We start by defining two state space vectors, $\vec{x}$ and $\vec{x}_\tau$ where,

$$\vec{x} = \begin{bmatrix} p(t) \\ R(t) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$
$$\vec{x}_\tau = \begin{bmatrix} p(t-\tau) \\ R(t-\tau) \end{bmatrix} = \begin{bmatrix} x_{1\tau} \\ x_{2\tau} \end{bmatrix}.$$

The nonlinear delay system can then be written in the form,

$$\dot{\vec{x}}(t) = \vec{g}(\vec{x}, \vec{x}_\tau), \tag{5.12}$$

where, $\vec{g} = \vec{g}_1(\vec{x}) + \vec{g}_2(\vec{x}_\tau)$, $\vec{g}_1 = \begin{bmatrix} -Dx_1 \\ -Af(x_1)x_2 \end{bmatrix}$, $\vec{g}_2 = \begin{bmatrix} Bf(x_{1\tau})x_{2\tau} \\ Af(x_{1\tau})x_{2\tau} \end{bmatrix}$. In this form, the system is written as a sum of a nonlinear component as a function of t and a nonlinear delay component as a function of $t-\tau$. We linearize each piece of g by computing the Jacobian matrix of the vector functions.

$$\mathbf{G}_1 = \frac{\partial \vec{g}_1}{\partial x} = \begin{bmatrix} \frac{\partial \vec{g}_{11}}{\partial x_1} & \frac{\partial \vec{g}_{11}}{\partial x_2} \\ \frac{\partial \vec{g}_{12}}{\partial x_1} & \frac{\partial \vec{g}_{12}}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -D & 0 \\ -Af'(x_1)x_2 & -Af(x_1) \end{bmatrix}, \tag{5.13}$$

and

$$\mathbf{G}_2 = \frac{\partial \vec{g}_2}{\partial x_\tau} = \begin{bmatrix} \frac{\partial \vec{g}_{21}}{\partial x_{1\tau}} & \frac{\partial \vec{g}_{21}}{\partial x_{2\tau}} \\ \frac{\partial \vec{g}_{22}}{\partial x_{1\tau}} & \frac{\partial \vec{g}_{22}}{\partial x_{2\tau}} \end{bmatrix} = \begin{bmatrix} Bf'(x_{1\tau})x_{2\tau} & Bf(x_{1\tau}) \\ Af'(x_{1\tau})x_{2\tau} & Af(x_{1\tau}) \end{bmatrix}, \tag{5.14}$$

where $f'(x_{1\tau}) = \frac{n\kappa^n x_{1\tau}^{n-1}}{(\kappa^n + x_{1\tau}^n)^2}$. The linearized system about an equilibrium q is then written as

$$\dot{\vec{x}} \approx \begin{bmatrix} -D & 0 \\ -Af'(x_1)x_2 & -Af(x_1) \end{bmatrix}\Big|_q (\vec{x} - \vec{q}) + \begin{bmatrix} Bf'(x_{1\tau})x_{2\tau} & Bf(x_{1\tau}) \\ Af'(x_{1\tau})x_{2\tau} & Af(x_{1\tau}) \end{bmatrix}\Big|_q (\vec{x}_\tau - \vec{q}). \tag{5.15}$$

If a change of variables, $\vec{y} = \vec{x} - \vec{q}$ is implemented, with $\vec{y}_\tau = \vec{x}_\tau - \vec{q}$, Eq. (5.15) simplifies to Eq. (5.16) effectively moving the equilibrium point to the origin.

$$\dot{\vec{y}} \approx \begin{bmatrix} -D & 0 \\ -Af'(p^*)R^* & -Af(p^*) \end{bmatrix} \vec{y} + \begin{bmatrix} Bf'(p^*)R^* & Bf(p^*) \\ Af'(p^*)R^* & Af(p^*) \end{bmatrix} \vec{y}_\tau. \tag{5.16}$$

We can write Eq. (5.16) in a simplified form as,

$$\dot{\vec{y}} \approx \mathbf{G}_1(q)\vec{y} + \mathbf{G}_2(q)\vec{y}_\tau. \tag{5.17}$$

We will use this linearized system to evaluate the stability of the equilibrium points of the system. Note that Eq. (5.17) was derived only from the DDE system Eq. (5.5). In addition, the constraint Eq. (5.3) must hold also for the perturbations. It is straightforward to directly compute the linearized system about the trivial equilibrium. We do this by inserting $(p^*, R^*) = (0, R_T)$ into (5.17), which leads to the elementary ODE system

$$\dot{\vec{y}} \approx \begin{bmatrix} -D & 0 \\ 0 & 0 \end{bmatrix} \vec{y}, \tag{5.18}$$

because all elements of $\mathbf{G}_2$ become zero. This system has only two characteristic exponents that are directly obtained from the diagonal of $\mathbf{G}_1$ as $-D$ and 0. Since the original nonlinear DDE system is infinite dimensional, there are infinitely many other characteristic exponents, which all tend to $-\infty$ as $\vec{q} \to \begin{bmatrix} 0 & R_T \end{bmatrix}^T$. Moreover, the characteristic exponent equal to zero corresponds to perturbations of the resources $R(t)$, which changes the value of the overall resources R_T. However, such perturbations do not fulfill the additional constraint equation (5.3), which means that this eigenvalue corresponds to the eigenvector along a one dimensional family of equilibria parameterized by the total resource R_T. As such this eigenvalue does not reflect the stability of the equilibrium within a phase space with R_T fixed. As a result, the trivial equilibrium point is a locally stable node as long as the decay rate is positive $(D > 0)$. We emphasize that using the Jacobian methods for linearization at this step is valid for constant delays. For state-dependent delays or time-dependent delays the system can be linearized using methods from [204].

5.1.3 Theory — Three Protein Model

5.1.3.1 Model

We extend the single protein model in Eq. (5.5) to incorporate production of three proteins with shared resource i.e. a shared ribosomal pool. If the resources are shared, we expect that oscillations may occur if there are not enough resources to produce all three proteins simultaneously. The

extended model is shown in Eq. (5.19).

$$\begin{aligned}
\dot{p_1}(t) &= B_1 f(p_2(t-\tau_1)) f(p_3(t-\tau_1)) R(t-\tau_1) - D_1 p_1, \\
\dot{p_2}(t) &= B_2 f(p_1(t-\tau_2)) R(t-\tau_2) - D_2 p_2, \\
\dot{p_3}(t) &= B_3 f(p_1(t-\tau_3)) R(t-\tau_3) - D_3 p_3, \\
\dot{R}(t) &= A(\mu_1(t-\tau_1) + \mu_2(t-\tau_2) + \mu_2(t-\tau_3) - \mu_1(t) - 2\mu_2(t)),
\end{aligned} \tag{5.19}$$

where A, B_1, B_2, B_3, τ_1, τ_2, τ_3, D_1, D_2, D_3, are system parameters, $\mu_1(t) = f(p_2(t))f(p_3(t))R(t)$, and $\mu_2(t) = f(p_1(t))R(t)$ and $f(x) = \frac{x^n}{\kappa^n + x^n}$. This means that production of the first protein is activated when the other two protein production rates are nonzero and production of the second and third proteins is activated by p_1. Analogously to the single protein system, the total resource (R_T) is computed using,

$$\begin{aligned}
R_T = R(t) + A \Bigg(&\int_{t-\tau_1}^{t} f(p_2(s)) f(p_3(s)) R(s) ds + \int_{t-\tau_2}^{t} f(p_1(s)) R(s) ds + \\
&\int_{t-\tau_3}^{t} f(p_1(s)) R(s) ds \Bigg).
\end{aligned} \tag{5.20}$$

5.1.3.2 Equilibrium Points

The equilibrium conditions are found by first setting $\dot{p_1} = \dot{p_2} = \dot{p_3} = 0$ yielding the following conditions:

$$\begin{aligned}
D_1 p_1^* &= B_1 f(p_2^*) f(p_3^*) R^*, \\
D_2 p_2^* &= B_2 f(p_1^*) R^*, \\
D_3 p_3^* &= B_3 f(p_1^*) R^*,
\end{aligned} \tag{5.21}$$

where (p_1^*, p_2^*, p_3^*,R^*) is the equilibrium point. Similarly to the single protein system, the $\dot{R}$ expression in Eq. (5.19) is always satisfied at equilibrium, but Eq. (5.20) yields the fourth and final equilibrium condition:

$$R_T = R^* + A\left[f(p_2^*) f(p_3^*) R^* \tau_1 + f(p_1^*) R^* (\tau_2 + \tau_3)\right]. \tag{5.22}$$

This system has a trivial equilibrium at $p_1^* = p_2^* = p_3^* = 0, R^* = R_T$. For finding the other equilibria, we need to solve this system of equations. We see that the relations in Eq. (5.21) all depend directly

on R^*. We eliminate R^* by solving Eq. (5.22) for R^*,

$$R^* = \frac{R_T}{1 + A\left(f(p_2^*)f(p_3^*)\tau_1 + f(p_1^*)(\tau_2+\tau_3)\right)}, \tag{5.23}$$

and substituting R^* in the three Eqs. (5.21), along with using the definition of $f(x)$. These steps result in three multivariate polynomial equilibrium equations shown in Eqs. (5.24),(5.25) and (5.26).

$$\begin{aligned} &D_1 p_1^*(\kappa^n + p_1^{*n})(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}) + AD_1 p_1^* p_2^{*n} p_3^{*n}(\kappa^n + p_1^{*n})\tau_1 \\ &+ AD_1 p_1^{*(n+1)}(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n})(\tau_2+\tau_3) = B_1 R_T p_2^{*n} p_3^{*n}(\kappa^n + p_1^{*n}), \end{aligned} \tag{5.24}$$

$$\begin{aligned} &D_2 p_2^*(\kappa^n + p_1^{*n})(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}) + AD_2 p_2^{*(n+1)} p_3^{*n}(\kappa^n + p_1^{*n})\tau_1 \\ &+ AD_2 p_1^{*n} p_2^*(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n})(\tau_2+\tau_3) = B_2 R_T p_1^{*n}(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}), \end{aligned} \tag{5.25}$$

$$\begin{aligned} &D_3 p_3^*(\kappa^n + p_1^{*n})(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}) + AD_3 p_2^{*n} p_3^{*(n+1)}(\kappa^n + p_1^{*n})\tau_1 \\ &+ AD_3 p_1^{*n} p_3^*(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n})(\tau_2+\tau_3) = B_3 R_T p_1^{*n}(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}). \end{aligned} \tag{5.26}$$

The solutions (p_1^*, p_2^*, p_3^*) to these three equations correspond to the equilibrium point of the system and the resource equilibrium is obtained by substituting these values in to Eq. (5.23). Due to the complexity of these equations, we solve them numerically using the variable precision (VPA) solver in Matlab [205]. Details for how these equations were solved are outlined in Sec. 5.1.6.

5.1.3.3 Three Protein System Linearization

The three protein system was also linearized about its equilibrium points for stability analysis using the spectral element linear stability method described in section 5.1.4.1. We will linearize the system about the equilibrium point, $\vec{q} = \begin{bmatrix} p_1^* & p_2^* & p_3^* & R^* \end{bmatrix}^T$ from the solution to Eqs. (5.24),(5.25), and (5.26). Similarly to the single protein model, we define state vectors for the current states and delayed states, but in this case, three delayed states are present due to the system having multiple time delays. The system states are,

$$\vec{x} = \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \\ R(t) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \; \vec{x}_{\tau_i} = \begin{bmatrix} p_1(t-\tau_i) \\ p_2(t-\tau_i) \\ p_3(t-\tau_i) \\ R(t-\tau_i) \end{bmatrix} = \begin{bmatrix} x_{1\tau_i} \\ x_{2\tau_i} \\ x_{3\tau_i} \\ x_{4\tau_i} \end{bmatrix}, \tag{5.27}$$

where $i \in [1,2,3]$ represents the system state at delay τ_i. We then write the system as:

$$\dot{\vec{x}} = \vec{g}(\vec{x}, \vec{x}_{\tau_1}, \vec{x}_{\tau_2}, \vec{x}_{\tau_3}), \tag{5.28}$$

and separate $\vec{g}$ as a sum of terms only dependent on one of the system states.

$$\vec{g}(\vec{x}, \vec{x}_{\tau_1}, \vec{x}_{\tau_2}, \vec{x}_{\tau_3}) = \vec{g}_1(\vec{x}) + \vec{g}_2(\vec{x}_{\tau_1}) + \vec{g}_3(\vec{x}_{\tau_2}) + \vec{g}_4(\vec{x}_{\tau_3}),$$

where,

$$\vec{g}_1(\vec{x}) = \begin{bmatrix} -D_1 x_1 \\ -D_2 x_2 \\ -D_3 x_3 \\ -Af(x_1)x_4 - 2Af(x_2)x_4 \end{bmatrix}, \qquad \vec{g}_2(\vec{x}_{\tau_1}) = \begin{bmatrix} B_1 f(x_{2\tau}) f(x_{3\tau}) x_{4\tau} \\ 0 \\ 0 \\ Af(x_{1\tau})x_{4\tau} \end{bmatrix},$$

$$\vec{g}_3(\vec{x}_{\tau_2}) = \begin{bmatrix} 0 \\ B_2 f(x_{1\tau})x_{4\tau} \\ 0 \\ Af(x_{2\tau})x_{4\tau} \end{bmatrix}, \qquad \vec{g}_4(\vec{x}_{\tau_3}) = \begin{bmatrix} 0 \\ 0 \\ B_3 f(x_{1\tau})x_{4\tau} \\ Af(x_{2\tau})x_{4\tau} \end{bmatrix},$$

where $f(x) = \frac{x^n}{\kappa^n + x^n}$. Now, the system can be linearized about q as,

$$\dot{\vec{x}} \approx \mathbf{G}_1(\vec{q})(\vec{x} - \vec{q}) + \sum_{i=2}^{4} \mathbf{G}_i(q)(\vec{x}_{\tau_i} - \vec{q}), \tag{5.29}$$

where $\mathbf{G}_i$ is the Jacobian matrix of $\vec{g}_i(\vec{x}_{\tau_{i-1}})$. The Jacobian matrices for this system are analytically computed with the matrix of partial derivatives. For example, $\mathbf{G}_1$ is computed as,

$$\mathbf{G}_1 = \begin{bmatrix} \frac{\partial \vec{g}_{11}}{\partial x_1} & \frac{\partial \vec{g}_{11}}{\partial x_2} & \frac{\partial \vec{g}_{11}}{\partial x_3} & \frac{\partial \vec{g}_{11}}{\partial x_4} \\ \frac{\partial \vec{g}_{12}}{\partial x_1} & \frac{\partial \vec{g}_{12}}{\partial x_2} & \frac{\partial \vec{g}_{12}}{\partial x_3} & \frac{\partial \vec{g}_{12}}{\partial x_4} \\ \frac{\partial \vec{g}_{13}}{\partial x_1} & \frac{\partial \vec{g}_{13}}{\partial x_2} & \frac{\partial \vec{g}_{13}}{\partial x_3} & \frac{\partial \vec{g}_{13}}{\partial x_4} \\ \frac{\partial \vec{g}_{14}}{\partial x_1} & \frac{\partial \vec{g}_{14}}{\partial x_2} & \frac{\partial \vec{g}_{14}}{\partial x_3} & \frac{\partial \vec{g}_{14}}{\partial x_4} \end{bmatrix},$$

where $\frac{\partial g_{1j}}{\partial x_k}$ is the partial derivative of the j-th component of the g_1 vector with respect to x_k. A similar process is used for $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ with the main difference being that the derivatives are

computed with respect to delayed states at the corresponding delay τ_i. Carrying out this procedure yields the following expressions for the Jacobian matrices.

$$\mathbf{G}_1(\vec{q}) = \begin{bmatrix} -D_1 & 0 & 0 & 0 \\ 0 & -D_2 & 0 & 0 \\ 0 & 0 & -D_3 & 0 \\ -2Af'(p_1^*)R^* & -Af'(p_2^*)f(p_3^*)R^* & -Af(p_2^*)f'(p_3^*)R^* & -2Af(p_1^*) - Af(p_2^*)f(p_3^*) \end{bmatrix},$$

$$\mathbf{G}_2(\vec{q}) = \begin{bmatrix} 0 & B_1f'(p_2^*)f(p_3^*)R^* & B_1f(p_2^*)f'(p_3^*)R^* & B_1f(p_2^*)f(p_3^*) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & Af'(p_2^*)f(p_3^*)R^* & Af(p_2^*)f'(p_3^*)R^* & Af(p_2^*)f(p_3^*) \end{bmatrix},$$

$$\mathbf{G}_3(\vec{q}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ B_2f'(p_1^*)R^* & 0 & 0 & B_2f(p_1^*) \\ 0 & 0 & 0 & 0 \\ Af'(p_1^*)R^* & 0 & 0 & Af(p_1^*) \end{bmatrix},$$

$$\mathbf{G}_4(\vec{q}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ B_3f'(p_1^*)R^* & 0 & 0 & B_3f(p_1^*) \\ Af'(p_1^*)R^* & 0 & 0 & Af(p_1^*) \end{bmatrix},$$

where $f'(x) = \frac{n\kappa^n x^{n-1}}{(\kappa^n + x^n)^2}$. Finally, we introduce the change of variables $\vec{y} = \vec{x} - \vec{q}$ to Eq. (5.29) resulting in the linearized system:

$$\dot{y} \approx \mathbf{G}_1(\vec{q})\vec{y}(t) + \mathbf{G}_2(\vec{q})\vec{y}(t-\tau_1) + \mathbf{G}_3(\vec{q})\vec{y}(t-\tau_2) + \mathbf{G}_4(\vec{q})\vec{y}(t-\tau_3). \tag{5.30}$$

For the trivial equilibrium $(0,0,0,R_T)$, we have $f(0) = 0$ and $f'(0) = 0$, and the matrices $\mathbf{G}_2$, $\mathbf{G}_3$ and $\mathbf{G}_4$ vanish. Thus, similarly to the single protein system, the linearization at the trivial

equilibrium becomes a simple ODE system

$$\dot{y} \approx \begin{bmatrix} -D_1 & 0 & 0 & 0 \\ 0 & -D_2 & 0 & 0 \\ 0 & 0 & -D_3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \vec{y}.$$

This system has four eigenvalues $-D_1$, $-D_2$, $-D_3$, and zero. Again, the characteristic exponent equal to zero corresponds to family of equilibria parameterized by R_T and can be discarded. We conclude that the trivial equilibrium point is locally stable for all positive decay rates ($D_1 > 0$, $D_2 > 0$, $D_3 > 0$).

5.1.4 Methods

5.1.4.1 Spectral Element Approach — Linear Stability Analysis

For analyzing the dynamic behavior of the system and identify regions of bistablity in parameter space, we study the linear stability of the equilibria. We use the spectral element method, which is an advanced numerical method for the stability analysis of DDE systems [206]. In particular, the linear variational systems Eq. (5.17) and Eq. (5.30) for perturbations around the equilibrium points are converted to a dynamic map, which describes the evolution of the system state $\vec{z}_{n-1}$ at time step $n-1$ to the system state $\vec{z}_n$ at time step n

$$\vec{z}_n = \mathbf{U}\vec{z}_{n-1}, \tag{5.31}$$

where $\mathbf{U}$ is the monodromy matrix. The state vector $\vec{z}_n$ is a discrete representation of the DDE state, which contains information of the system variable $\vec{y}$ for the time interval $[t-\tau_{\max}, t]$, where $\tau_{\max}$ is the maximum delay. The matrix $\mathbf{U}$ is a high dimensional approximation of the monodromy operator which is an operator that allows for mapping dynamic states forward in time by one period [206]. The full operator is infinite dimensional, but this method utilizes finite approximations of the operator to permit computing approximate solutions to the characteristic equation of the system. Specifically for the spectral element method, because the system is transformed into a discrete map, if the eigenvalues of $\mathbf{U}$ have a magnitude less than unity, that equilibrium point is stable and

larger than one makes it unstable. If the magnitude is exactly one the equilibrium point is said to be marginally stable and this also is indicative that the equilibrium is non-hyperbolic [207]. Note that the monodromy matrix for an autonomous system always has a trivial eigenvalue $\lambda = 1$ which does not dictate the stability of the equilibrium point [208]. In the case where the trivial eigenvalue is the furthest from the origin we take the second largest eigenvalue of the system to characterize the stability.

This problem falls under the broader classification of *pseudospectral differencing methods* where in general an approximation of an infinite dimensional operator is computed resulting in a matrix where the eigenvalues approach solutions to the characteristic equation of the linear delay differential equations [209]. Further, Breda et al. proved many useful convergence results for such methods such as the fact that none of the eigenvalues of the approximation matrix are "ghost roots". This means that all of the computed eigenvalues will eventually converge to a true root of the full infinite dimensional system if sufficient nodes are used in the discretization [209]. It is also known that roots closer to the origin in the complex plane are approximated first with these differencing methods so it is important to use sufficient discretization meshes to find the unstable eigenvalues [209]. It has been shown that for a DDE system the number of characteristic equation roots in the right half of the complex plane is finite [210] meaning that if enough eigenvalues are approximated for the system about its equilibrium, eventually the right most eigenvalue will be computed which allows for characterizing the stability of the equilibrium point. For a discrete system this means that the number of eigenvalues outside of the unit circle is finite. In further sections, methods described in [206] are applied for discretizing and computing dominant eigenvalues for the metabolic systems to evaluate the system stability at different parameters.

5.1.4.2 Numerical Simulations

For the second method, we chose to perform many numerical simulations to demonstrate the behavior of the system and connect the results to experimental observations. This was done by brute force simulation of the system using the Julia differential equations library. The goal was to study specific features of the system trajectories in the parameter space to locate regions with

different types of solutions. A diagram is obtained from the simulations by computing scalar features of the asymptotic solutions from time domain simulations and plotting the result as an image as a 2D projection of the overall parameter space. The features can be used to distinguish periodic solutions from equilibria. If the simulation times are long enough such that the system behaviour is characteristic of its long run behavior, we refer to this as the *steady state response* as this is when the transient response has dissipated. The method for computing these response feature diagrams for this system was inspired by the *AttractionsViaFeaturizing* function of the dynamical systems library in Julia [211]. This method computes a feature $M : \mathbb{R}^n \to \mathbb{R}$ on the system trajectory with n system variables that indicate different features of the system response at each point in the parameter space. For this analysis, we needed to fix the history function for our systems. Specifically for this paper, we focus on a feature based on the amplitude of the time series signals. However, many other features can be used to study system behavior such as the mean response and standard deviation. We compute the amplitude feature A of a response $x_i(t)$ as:

$$A_i = \frac{1}{2}\left(\max\left(x_i(t)\right) - \min\left(x_i(t)\right)\right).$$

The amplitude feature is then consolidated into a scalar value by summing over the variables in i as:

$$M_A = \sum_{i=1}^{n} A_i. \tag{5.32}$$

If the trajectory is stable, we expect M_A to be close to zero and if the response contains oscillations M_A should be nonzero and finite.

5.1.4.3 Low Growth History Functions

To compute features of the system response, sufficient information is required to simulate the system such as the start time, end time and initial conditions. One critical difference between time delay differential equation systems (DDE) and ordinary differential equation systems (ODE) is that a DDE system requires the solution to be defined over the interval $[-\tau_{\max}, 0]$ rather than just supplying a single point initial condition for an ODE system. A history function was chosen based on the experimental process for achieving these metabolic oscillations in practice [177]. In this

paper, the authors starve the cells of all resource prior to the oscillations. Consequently, the protein production rate is also zero during this time.

Single Protein History Function and Initial Conditions: To define the history function for the single protein model, we assume that the cell was operating at zero protein production on $[-\tau,0)$. In other words, $p(\theta)=0$ and $R(\theta)=0$ for $\theta\in[-\tau,0)$, whereas at time $t=0$ we set $p(0)=p_0$. We obtain the initial conditions of the system by letting $t=0$ in Eq. (5.3) yielding:

$$R_T = R_0 + A\int_{-\tau}^{0} f(p(s))R(s)ds, \tag{5.33}$$

where $R_0=R(0)$ is the resource value at time $t=0$. For a response of this system to be valid, Eq. (5.33) must hold for the value of R_T used for the simulation. Since p and R are zero for the history function, the integral in Eq. (5.33) vanishes and we have $R_0=R_T$. As a result, for each simulation R_T can be specified and any positive value of p_0 can be chosen. This system can then be studied by varying the parameters p_0, τ and R_T and holding the remaining parameters constant to determine which parameter values result in periodic solutions.

Three Protein History Function and Initial Conditions: The solutions for the three protein system are also studied using the same zero resource and zero protein production assumption prior to $t=0$. So $p_1(\theta)=p_2(\theta)=p_3(\theta)=0$ and $R(\theta)=0$ for $\theta\in[-\tau_{\max},0)$. Applying this to the total resource equation from Eq. (5.20) again yields $R_0=R_T$. Similar to the single protein system, the initial protein production rates p_{10}, p_{20} and p_{30} can be varied in the system. The dimension of the parameter space is reduced by limiting this system to the case where $\tau_1=\tau_2=\tau_3=\tau$ or in other words, all three proteins require the same production time. We then vary this delay and the total resource to determine which parameter combinations yield oscillations in the system response.

5.1.4.4 Boundary Value Calculation of Periodic Solutions to Nonlinear DDE Systems

As an alternative to detecting periodic orbits by simulation and amplitude computation, we will find them directly by solving a boundary value problem (BVP). This method comes from [212] where the authors describe how a nonlinear DDE system can be converted to a BVP where the solutions to this problem correspond to periodic solutions of the original system. Specifically, the system is converted to the boundary value problem in Eq. (5.34) and the spectral element

method is used to discretize the DDE system to approximate the infinite dimensional BVP as a finite dimensional problem that can be solved numerically to obtain a single period of the periodic solution to the system [212].

$$\begin{aligned} \vec{f} &= \frac{d\vec{x}}{dt} - T\vec{g}(\vec{x}(t), \vec{x}(t-\tau/T)) = 0,\ t \in [0,1], \\ &\vec{x}(s) - \vec{x}(s+1) = 0,\ s \in [-\tau/T, 0], \\ &p(\vec{x}) = 0, \end{aligned} \tag{5.34}$$

where T is the system period, $\vec{x}$ is the vector of system variables, and the first line corresponds to the specific DDE system being studied. In this case we take $\vec{g}$ to be Eq. (5.12) for the single protein system, and Eq. (5.28) for the three protein system. The second line imposes a periodicity condition on the system and the last line imposes a phase condition to yield a unique periodic solution by setting $p(\vec{x})$ to be the inner product of the initial state ($\vec{x}_0$) and the time derivative $\dot{\vec{x}}(t)$ [212]. An initial guess is provided by simulating the system in Julia using the differential equations library and this simulation is provided to the boundary value problem solver in Matlab to perform Newton-Raphson iteration and converge on the periodic solution. This process also yields an approximation to the period T of the system. This method has been shown to compute accurate periodic solutions for nonlinear DDE systems with exponential convergence rates as the number of mesh points increases [212] making it ideal for verifying parameters of the metabolic system that result in oscillating solutions to the system.

5.1.5 Results — Single Protein

The single protein system is analyzed in this section by applying the three methods outlined in Sec. 5.1.4.

5.1.5.1 Spectral Element Linear Stability

The eigenvalues of the linearized single protein system about its nontrivial equilibrium points were approximated at each set of parameters in a 400×400 grid in the (τ, R_T) plane varying each parameter from zero to 50 using the monodromy matrix from the spectral element method [206]. We hold the remaining parameters constant at $\kappa = 0.5$, $A = 1.0$, $B = 2.0$, $D = 10.0$, $n = 2$. The monodromy matrix requires an oscillation period to map the system states to the next period. It

has been shown that for systems with one delay the period can be set as the delay for stability computations [206]. For this reason, we set the period to τ for this stability analysis. We examine the stability of the equilibrium points by plotting the magnitude of the eigenvalue furthest from the origin. Stability diagrams were plotted for the nontrivial equilibrium points. Note that below the curve in Eq. (5.11), only the trivial equilibrium is present $(0, R_T)$, but above this curve three equilibrium points exist in the system. We only consider the stability of the nontrivial equilibria in this section as the stability of the trivial solution is computed analytically in Section 5.1.2.3. As a result, we color points in the stability diagram as white if only the trivial equilibrium is present in that region.

We start by computing the stability of the middle equilibrium point as shown in Fig. 5.1 where we plot the dominant eigenvalue of the middle equilibrium point for combinations of τ and R_T between 0 and 50. We see that for all parameters shown, this equilibrium point is unstable because its largest eigenvalue is outside of the unit circle in the complex plane.

Next, we plot the largest magnitude eigenvalue of the top equilibrium point in Fig. 5.2 where we see that for small delay and sufficient resource, the top equilibrium point is stable with $|\lambda| < 1$. As the delay increases for a given total resource, a pair of complex conjugate eigenvalues leave the unit circle that govern the stability of this equilibrium point making it an unstable focus [207].

Therefore, a Hopf bifurcation occurs from the top equilibrium point along this line. We plot the Hopf bifurcation curve in subsequent stability diagrams as a green line with dots. This line was found to be approximately,

$$R_T = 2.6449\tau + 4.6323, \tag{5.35}$$

for $\tau \geq 0.75$ by computing a linear regression along the boundary where the eigenvalue exits the unit circle. The model had a coefficient of determination of 0.9999 indicating that this boundary is well approximated by a linear model.

5.1.5.2 Response Features

Response feature diagrams were generated for the single protein system (Eq. (5.5)) for $\kappa = 0.5$, $A = 1.0$, $B = 2.0$, $D = 10.0$, $n = 2$ with varying τ, R_T and p_0. The results for these simulations are

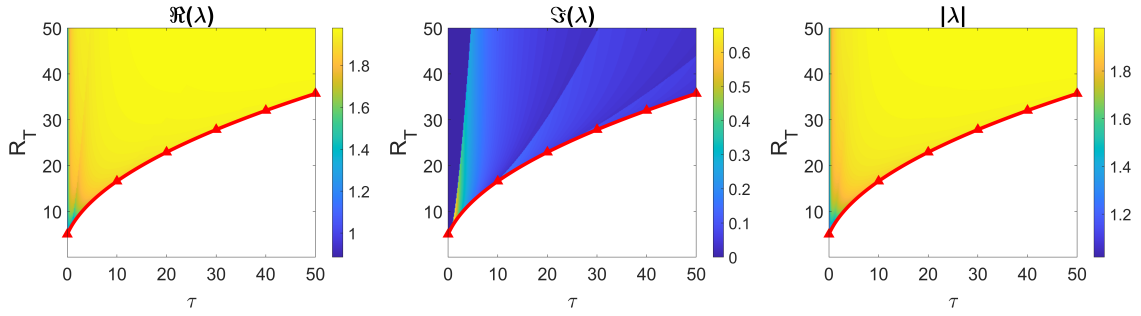


Figure 5.1 Single protein middle equilibrium point stability diagrams. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant eigenvalue, (right) the modulus of the eigenvalue. The red curve with triangles is the saddle node boundary that separates regions with 1 and 3 equilibria. Above the red curve all three equilibrium points exist and below the curve only the trivial point is present.

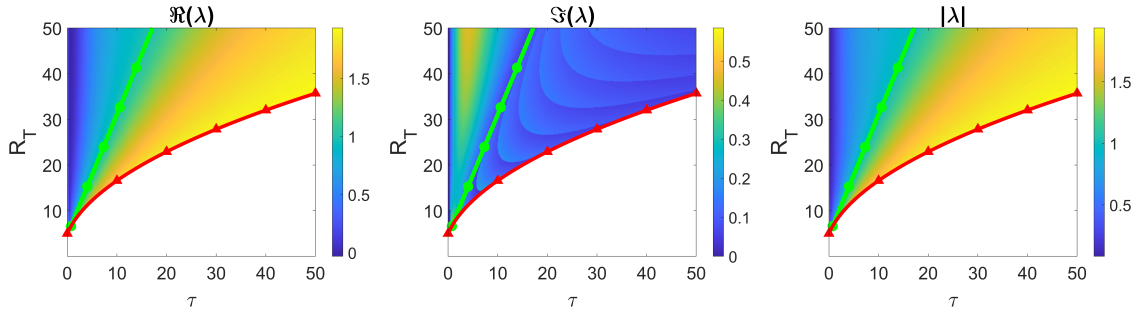


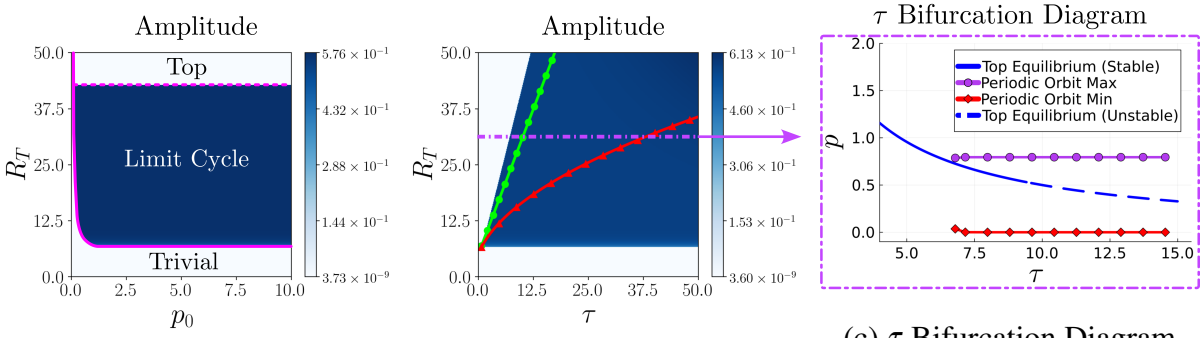
Figure 5.2 Single protein top equilibrium point stability diagrams. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant eigenvalue, (right) the modulus of the eigenvalue. The red curve with triangles is the saddle node boundary that separates regions with 1 and 3 equilibria. Above the red curve all three equilibrium points exist and below the curve only the trivial point is present. The Hopf bifurcation curve is shown as a line with green dots.

shown in Fig. 5.3 where we color pixels in the parameter space according to the response amplitude feature using Eq. 5.32. We used the starving cell history function from Section 5.1.4.3 and each simulation was taken between 10000–11000 time units to ensure that the transient response had dissipated. The authors acknowledge the arbitrarily chosen parameters for this system and that these parameters may not be in biologically significant range. However, our model is conceptual and the purpose of this paper is to demonstrate that certain parameters yield oscillations in the protein production when the cell is starved of resource prior to $t = 0$. This is also the reason why

we use "time units" instead of seconds for the simulations.

First we study the dependence on the initial protein production rate p_0 by fixing the delay at $\tau = 10$ and plotting the amplitude feature over the region $(p_0, R_T) \in [0,10] \times [0,50]$. We see in Fig. 5.3 (a) that for nontrivial p_0, the response is essentially independent of the initial condition so any large enough initial protein production rate was sufficient. For small p_0 the response approaches the trivial equilibrium point. While this diagram is only shown for a single delay, we observed a trend where as the delay varies, the only change is in the width of the limit cycle region for large enough p_0. For this reason, we arbitrarily choose $p_0 = 10$ for our initial p_0.

Next, we keep $p_0 = 10$ and vary the parameters $(\tau, R_T) \in [0,50] \times [0,50]$ and plot the amplitude feature in this region of the parameter space in Fig. 5.3 (b) along with the Hopf and saddle node bifurcation boundaries obtained from the linear stability analysis. We see that periodic solutions were found above the Hopf curve for this particular history function indicating that the Hopf bifurcation is subcritical. So slightly above the green curve we have a bistability between the top equilibrium, trivial equilibrium, and the limit cycle. Below the Hopf curve we have a bistability between the trivial equilibrium point and the limit cycle and below $R_T \approx 7$ we did not observe any oscillations and the trajectory approached the trivial equilibrium. Note that the pink curves in Fig. 5.3 (a) are specific to $\tau = 10$ and will increase as the delay is increased according to the bounds of the periodic region in Fig. 5.3 (b). In other words, at a delay of 10 if we draw a vertical line in Fig. 5.3 (b), we should expect it to intersect the blue region at $R_T \approx 7$ and $R_T \approx 42$ which correspond to the pink curves in Fig. 5.3 (a) for nontrivial p_0. We can also show a horizontal slice of Fig. 5.3 (b) which produces a bifurcation diagram in τ as shown in Fig. 5.3 (c). The stable periodic orbit was generated by setting $R_T = 30$ and using simulations with the initial conditions from Fig. 5.3 (b). The stability region for the top equilibrium point was computed using analytical expressions. This region ends in subcritical Hopf bifurcation at $\tau \approx 9.59$. Importantly, the region between $\tau \approx 6.78$ and $\tau \approx 9.59$ exhibits bistability since the stable equilibrium and a stable periodic orbits coexist. Since the branch of periodic orbits connecting the Hopf bifurcation to the stable periodic orbit at $\tau \approx 9.59$ is unstable, we are unable to find it using simulations.



(a) p_0 Dependence (b) τ, R_T Dependence (c) τ Bifurcation Diagram ($R_T = 30$)

Figure 5.3 Single protein model response feature diagrams by varying system parameters p_0, τ and R_T and simulating the system at for each parameter combination between 0 and 50. The solid pink curve corresponds to a point on the horizontal boundary at $\tau = 25$ in the middle image and the dashed pink curve corresponds to a point on the boundary above the green curve at $\tau = 25$ in the middle image. The red curve with triangles is the saddle node boundary that separates regions with 1 and 3 equilibria, the line with green dots is the Hopf bifurcation boundary. The right image corresponds to a horizontal slice of the middle plot at $R_T = 30$ to show the bifurcation diagram as τ is varied.

5.1.5.3 Periodic Solutions

Next we utilize the spectral element approach to solve the boundary value problem in Eq. (5.34). This process was performed on the three points in the (τ, R_T) parameter space where oscillations were expected and the two points where we expect fixed point responses. The first point considered was $\tau = 12$ and $R_T = 50$. We see that this point corresponds to a response with nonzero amplitude indicating that oscillations should be expected and is in the subcritical region of the Hopf bifurcation. The system was simulated and sampled between 15,988–16,000 time units for the period. Because the system is autonomous, we can take the period to be equal to the delay. Solving the boundary value problem in Eq.(5.34) for the periodic solution, we obtain the response shown in Fig. 5.4. We see that the periodic solution from the boundary value problem closely matches the simulation result with the period matching the delay. Further, the protein production rate is nearly constant and close to the top equilibrium point $(p^*, R^*) = (0.7434, 5.3982)$ for most of the period in this case with a drop in the production rate emerging yielding the metabolic oscillations.

The next parameters that were considered were $\tau = 10$ and $R_T = 20$. The system was simulated at these parameters from 15,990–16,000 time units and the period was set to 10. Passing this

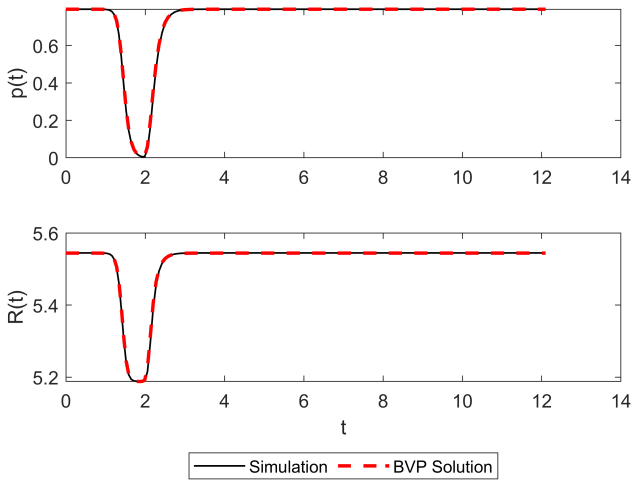


Figure 5.4 $\tau = 12$ and $R_T = 50$ single protein periodic solution results from solving the relevant boundary value problem.

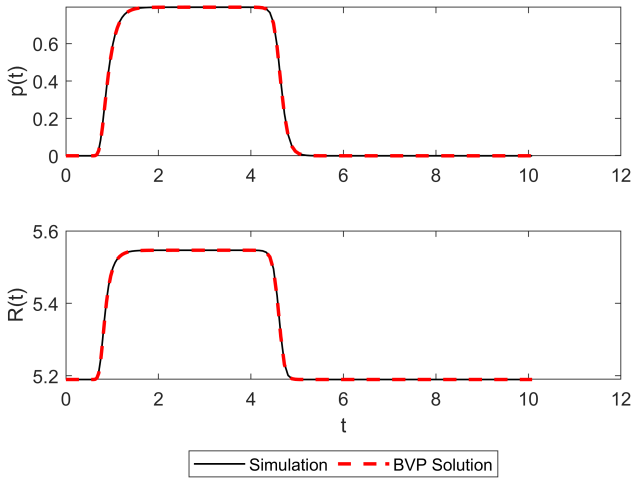


Figure 5.5 $\tau = 10$ and $R_T = 20$ single protein periodic solution results from solving the relevant boundary value problem.

initial guess into the boundary value problem, the obtained periodic solution is shown in Fig. 5.5. Interestingly, we see that the time that the protein production rate spends at 0 is much longer compared to Fig. 5.4. As the Hopf bifurcation curve is crossed, the drop in the protein production rate appears to spend more time at zero during the oscillation period. The third set of parameters considered was $\tau = 45$ and $R_T = 15$. The system was simulated at these parameters from 15,955–16,000 time units and the period was set to 45 resulting in the periodic solution shown in Fig. 5.6. We see that the trajectory starts to spend more time near a protein production rate of zero as the

total resource approaches the horizontal line $R_T \approx 7$ in Fig. 5.3 (b). The periodic solutions shown demonstrate the transition from the fixed point stability at the top equilibrium to fixed point stability at the trivial equilibrium.

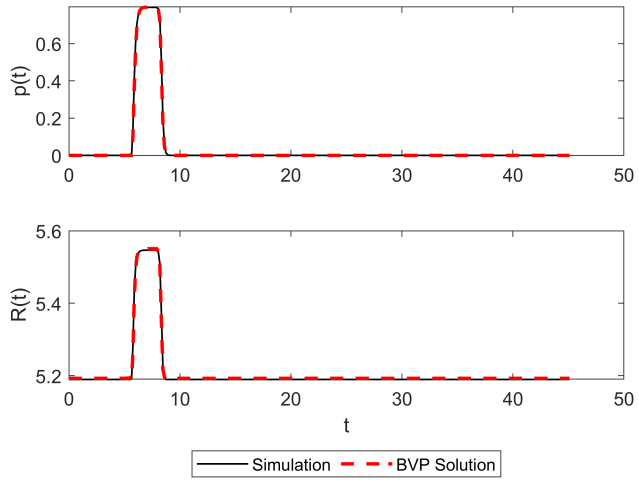


Figure 5.6 $\tau = 45$ and $R_T = 15$ single protein periodic solution results from solving the relevant boundary value problem.

5.1.5.4 Steady State Solutions

We also explore the steady state solutions of the system by examining two parameter conditions. The first case is where the total resource is too low to sustain protein production (low growth conditions). In this case, we found a trajectory that approaches the trivial equilibrium point. This was verified by simulating the system at $\tau = 45$ and $R_T = 5$. The resulting response is shown in Fig. 5.7 where we see the system approach $(p,R) = (0,R_T)$. We also consider the case where the

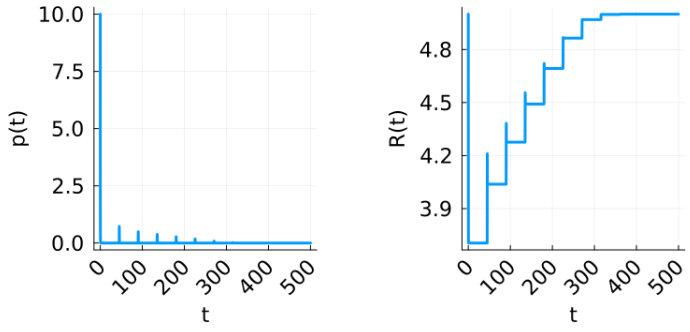


Figure 5.7 Approach to trivial fixed point for low growth conditions in the cell ($\tau = 45$, $R_T = 5$).

cell has access to plentiful resources and can synthesize proteins at a constant rate (high growth conditions). To examine this case, we simulated the system at $\tau = 5$ and $R_T = 50$. The response for these parameters is shown in Fig. 5.8. We see that as time progresses, the protein production rate approaches a steady state value because the cell is able to produce proteins at a constant rate. Further, the point that this trajectory approaches corresponds to the top equilibrium point of the system which for these parameters works out to be $(p^*, R^*) \approx (1.6412, 8.968)$.

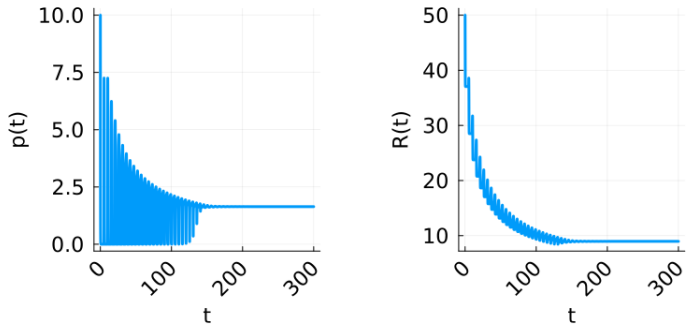


Figure 5.8 Fixed point response for high growth conditions in the cell ($\tau = 5$, $R_T = 50$).

5.1.5.5 Single Protein Summary

Three distinct behaviors were observed in the single protein time delay model. First, if the resources are not sufficient to sustain metabolic activity, the system will approach the trivial equilibrium point with zero protein production rate. If the resources are plentiful, they can sustain constant protein production at the top equilibrium point. Between these two cases, the cell initially has enough resource to synthesize proteins, but as the protein production rate increases, resources are used and the metabolic activity decreases. This balance between constant production and no production seems to lead to oscillations in the system response. Slightly above the Hopf bifurcation curve, we observe oscillations that are close to a constant solution at the top equilibrium and as the parameters cross the Hopf curve and approach the line $R_T \approx 7$, the solution continues to oscillate but with a solution that is closer to a constant solution at the trivial equilibrium. The oscillation region represents a transition between the top and trivial equilibria and the middle solution remains unstable for all parameters in this region. For $\tau < 0.75$, the solution can switch

directly between the top and trivial equilibrium with no oscillations, but after this bifurcation point at $(\tau, R_T) \approx (0.75, 6.6)$, the periodic solution emerges to transition from constant to zero protein production.

5.1.6 Results — Three Proteins

The three protein system is analyzed in this section by applying the three methods outlined in Sec. 5.1.4.

5.1.6.1 Spectral Element Linear Stability

Our main goal with the three protein system was to find parameters where the protein production rates peak at different times in the period. This phenomena would be indicative of the cell prioritizing its resources to produce proteins in a way that could be more efficient. To begin exploring this systems parameter space, we use the spectral element method to study the linear stability of the three protein system with arbitrarily chosen parameters $\kappa = 0.5$, $A = 1.0$, $B_1 = 2.0$, $B_2 = 2.0$, $B_3 = 2.0$, $D_1 = 10.0$, $D_2 = 10.0$, $D_3 = 10.0$, $n = 2$. We set $\tau_1 = \tau_2 = \tau_3 = \tau$ such that all three proteins require the same amount of production time, and vary τ and R_T just as was done with the single protein system.

However, for this system we do not have analytical expressions for the equilibrium solutions and we only have the coupled polynomial system in Eqs. (5.24), (5.25), and (5.26). Solving this system of equations is a nontrivial task, but if we make some assumptions based on our observations from the single protein system we can still generate stability diagrams using this method. Namely, we will assume that this system also exhibits three possible equilibrium points (top, middle, and trivial). Using the variable precision accuracy (VPA) solver in Matlab, we can solve this system of equations numerically in our parameter space and approximate the dominant eigenvalues to characterize the stability of each point. The documentation for the VPA solver used states that for polynomial systems, all solutions in a region will be returned by the function [205]. This solver was applied to a 400×400 grid of parameters in $(\tau, R_T) \in [0, 50] \times [0, 100]$, and the assumption was found to be correct where one region of the space contained 3 equilibrium points and the other region only contained the trivial point.

With the single protein system, we defined the top and middle equilibrium points based on the magnitude of the equilibrium protein production rate p^*. However, in the three protein system we have multiple equilibrium protein production rates. To modify this approach for the three protein system, we form the following vector of equilibrium coordinates,

$$\vec{p} = \begin{bmatrix} p_1^* & p_2^* & p_3^* \end{bmatrix}^T.$$

Thus, the top and middle equilibria are defined by the l_2 norm of $\vec{p}$ where the top equilibrium has the largest l_2 norm and the middle solution has a norm between the top and trivial. We then use Eq. (5.23) to obtain R^* for a given set of parameters at each equilibrium point. This system can then be linearized about each equilibrium point using Eq. (5.30) and the dominant eigenvalue of the linearized system at a given set of parameters can be approximated with the spectral element method described in Section 5.1.4.1. We note that because a single delay is present, we use this delay for the system period when computing the monodromy matrix for the system.

Similar to the single protein model, we start by plotting the stability of the middle equilibrium point in Fig. 5.9. Note that we color the region as white if only the trivial equilibrium is present. We see that there are two distinct regions in the plot of the magnitude of the eigenvalue. The curve that separates these regions is the saddle node bifurcation curve for this system. Using a third order polynomial fit, the saddle node curve was found to be,

$$R_T = 0.0016\tau^3 - 0.1118\tau^2 + 4.8855\tau + 10.3749, \tag{5.36}$$

for $\tau \geq 0.625$. This curve had a coefficient of determination of 0.9997. We plot the saddle node boundary as a red curve with triangles in the stability diagrams.

Because the dominant eigenvalue for the middle equilibrium always has a modulus greater than one for these parameters, this equilibrium point will not govern the stability of the system if another point is stable or marginally stable. This was also the case with the single protein system.

Lastly, we plot the stability of the top equilibrium point of this system in Fig. 5.10. These diagrams show that for small delay and large resource, this point is stable. As the delay increases for a given resource, this point becomes an unstable focus by way of a Hopf bifurcation. We plot

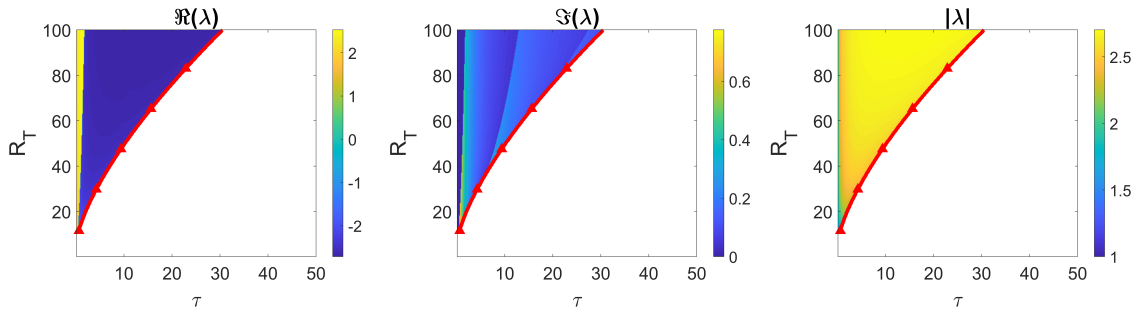


Figure 5.9 Three protein middle equilibrium point stability diagrams at equal delay. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant eigenvalue, (right) the modulus of the eigenvalue. The red curve with triangles is the saddle node boundary that separates regions with 1 and 3 equilibria. Above the red curve all three equilibrium points exist and below the curve only the trivial point is present.

the Hopf bifurcation curve as a green line with dots. This curve was approximated by locating points in the parameter space with unit length eigenvalues. Using linear regression, the Hopf bifurcation curve was approximated to be,

$$R_T = 12.0948\tau + 4.7910, \tag{5.37}$$

for $\tau \geq 0.75$. This model had a coefficient of determination of 0.9987 suggesting that it is a good approximation. Note that while there is interesting behavior that occurs between the green (dots) and red (triangles) curves in these stability diagrams, all of the eigenvalues plotted are outside of the unit circle and are therefore unstable so this is not a bifurcation it just means that another eigenvalue moved further from the origin.

5.1.6.2 Response Features

Holding the remaining parameters constant at the values from Section 5.1.6.1, an amplitude feature diagram was generated with equal delays for all three proteins and varying the delay τ with the total resource R_T. The three protein system was simulated between 10,000–11,000 time units using the zero history function described in Sec. 5.1.4.3, and the amplitude feature was plotted in the $\tau - R_T$ space where τ is the same for all three proteins. The results are shown in Fig. 5.11. We see that the amplitude diagram has a similar structure to the single protein system, but there is a change in the amplitude feature for a small region at low total resource. System responses

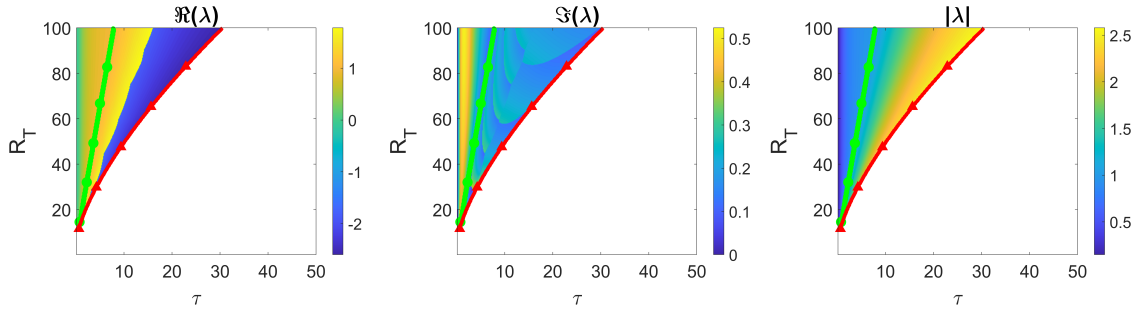


Figure 5.10 Three protein top equilibrium point stability diagrams at equal delay. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant eigenvalue, (right) the modulus of the eigenvalue. The red curve with triangles is the saddle node boundary that separates regions with 1 and 3 equilibria. Above the red curve all three equilibrium points exist and below the curve only the trivial point is present. The Hopf bifurcation curve is shown as a line with green dots.

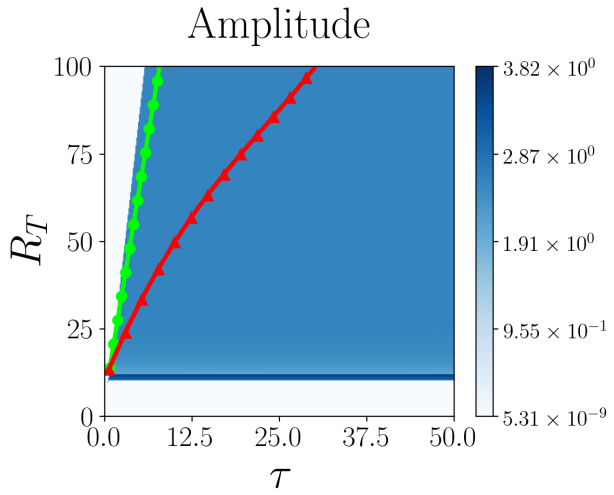


Figure 5.11 Three protein system response amplitude diagram in the $\tau - R_T$ parameter space where τ is the same delay for all three proteins. The low growth history function was used for all simulations in this diagram.

in these regions are explored further in Section 5.1.6.3. We plot the Hopf bifurcation curve from Section 5.1.6.1 in Fig. 5.11 as the green line with dots. The red curve with triangles corresponds to the saddle node bifurcation curve from the approximations in Section 5.1.6.1.

5.1.6.3 Periodic Solutions

Three points were considered within the region of the parameter space with nontrivial amplitude in Fig. 5.11. Namely, we choose $(\tau, R_T) = (5.7, 100)$, $(25, 50)$ and $(25, 11.8)$. The first

point is in the nonzero amplitude region to the left of the Hopf bifurcation curve. This point was chosen to verify the subcriticality of the Hopf bifurcation. The second point was chosen to show the response near the middle of the oscillation region. We chose the third point in the region of differing amplitude at low total resource to observe the changes that occur when the cell has limited resources available. The steady state regions or regions with near zero amplitude were found to exhibit similar behaviors to the single protein model outside of the oscillation region so these responses will not be considered.

Starting with $\tau = 5.7$ and $R_T = 100$, the system was simulated between 15,000 and 15,005.7 time units to capture a single period of the response. This solution was then verified by solving the nonlinear DDE boundary value problem to obtain the periodic solution in Fig. 5.12. We see that the obtained solution is nearly identical to the simulation and appears to be close to a constant solution at the top equilibrium point which for these parameters is at $(p_1^*, p_2^*, p_3^*, R^*) \approx (0.9942, 1.1328, 1.1328, 7.0965)$. We also note that all of the protein production rates here appear to oscillate in-phase.

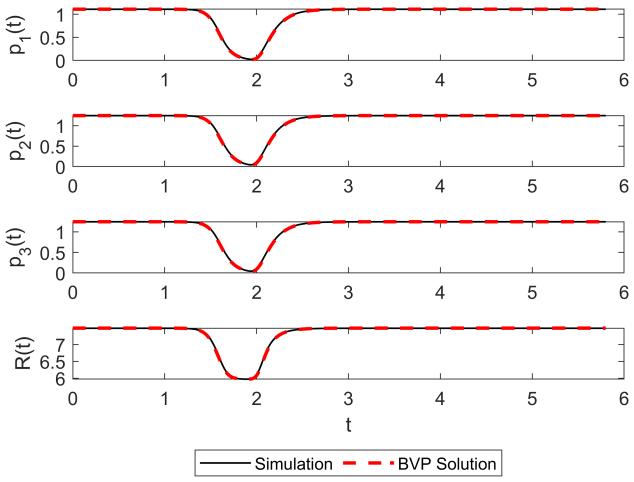


Figure 5.12 Boundary value problem solution for the three protein system with $\tau = 5.7$ and $R_T = 100$.

Next, we study the solution when $\tau = 25$ at the same resource $R_T = 50$. This point corresponds to a region in the response feature diagram in Fig. 5.11 with the same amplitude as the solution in Fig. 5.12. Plotting the response for these parameters between 15,000 and 15,025 time units

yielded Fig. 5.13. We see that as the total resource has decreased, the periodic solution is at zero protein production rates for most of the period. Again, the protein production rates appear to oscillate in-phase for these parameters. Lastly, we compute a periodic solution in the thin region

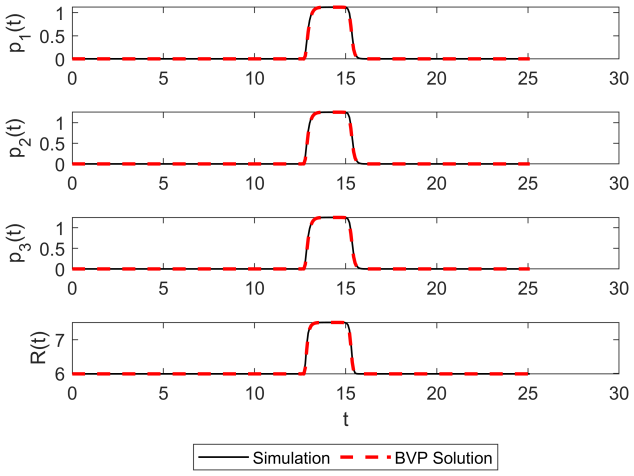


Figure 5.13 Boundary value problem solution for the three protein system with $\tau = 25$ and $R_T = 50$.

of larger amplitude feature at low total resource. To do this, we chose $\tau = 25$ and $R_T = 11.8$. The periodic solution was obtained using simulation data between 15,000 and 15,025 time units and the resulting solution is shown in Fig. 5.14. It is clear that the periodic solution at these parameters is much different from the others. We see that the peak for $p_1(t)$ has shifted out of phase with the other proteins. This behavior could indicate that at extremely low resource, the best way to share that resource is to separate production of different proteins to different times, as this results in a more efficient use of resource for the cell.

5.1.6.4 Three Protein Summary

The three protein system was found to behave similarly to the single protein system. Three equilibrium points were found numerically, and a subcritical Hopf bifurcation was found in the $\tau - R_T$ parameter space where all three proteins exhibited the same production time τ. A large region of periodic solutions was found by numerical simulation by way of a resource limiting history function which aligns with experimental observations [177]. It was found that for small delay and large total resource, the cell can produce all proteins at a constant rate at the top equilibrium

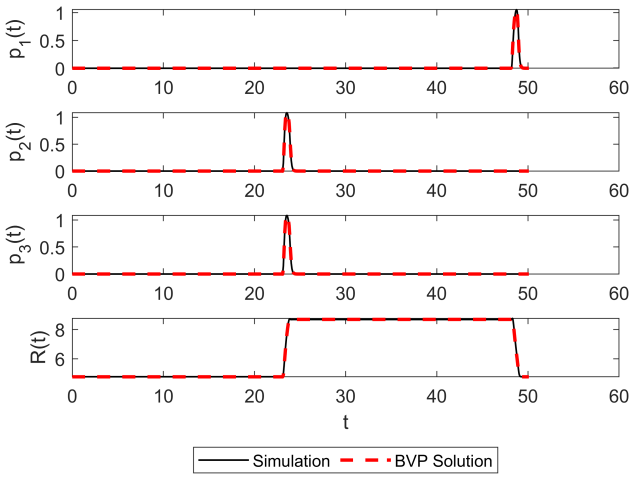


Figure 5.14 Boundary value problem solution for the three protein system with $\tau = 25$ and $R_T = 11.8$.

point. Due to the subcritical Hopf bifurcation of the top equilibrium point, an oscillation region appears in the parameter space which facilitates the transition from constant production to zero production just as was observed in the single protein system. Below the Hopf bifurcation curve, there is a bistability between the trivial equilibrium and the limit cycle, but slightly above the Hopf bifurcation curve there is a bistability between the top and trivial equilibrium points and the limit cycle. The trivial equilibrium was found to have a small basin of attraction and is only approached for small initial protein production rates p_0.

5.1.7 Conclusion

We introduced a nonlinear time delay framework for modeling metabolic oscillations during protein synthesis in yeast cells. The model contains many parameters that control the behavior of the system where the delays correspond to the respective protein production times. The single protein and three protein variants of this model were studied to locate regions in the parameter space where the metabolic activity contained oscillations. Due to the complexity of the model, three numerical methods were utilized for locating limit cycles in these systems. First, a spectral element method was used to approximate the system as a high dimensional map whose eigenvalues approximate the true spectrum of the system allowing for the stability of the fixed points to be characterized in a subset of the parameter space. Three equilibrium points were found for

each system and a subcritical Hopf bifurcation curve was located in the parameter space where an equilibrium point of the system loses stability and a limit cycle emerges leading to periodic solutions. The second numerical method utilized system simulations carried out over a range of τ to show the region of bistability where the stable steady state and a stable periodic orbit coexist. The simulation results were verified with the third numerical method where a finite dimensional boundary value problem (BVP) was solved by discretizing the system using a spectral element approach. We found a large region of the parameter space to have nonzero amplitude of oscillation for a range of delay and total resource values. It was observed that the oscillation region forms as a transition between two steady states in the system (constant production and zero production) for large enough production times. It was also observed that for the three protein model when the resources were shared and each protein had an equal production time, certain parameters at low total resource resulted in a temporal shift in the protein production rate peaks. Our simulation results are consistent with what has been observed in experiment, and our model helps argue that the observed temporal shift is a more efficient use of resources for the cell.

BIBLIOGRAPHY

[1] A. D. Myers and F. A. Khasawneh, "Damping parameter estimation using topological signal processing," *Mechanical Systems and Signal Processing*, vol. 174, p. 109042, 7 2022.

[2] S. Tymochko, E. Munch, and F. A. Khasawneh, "Using zigzag persistent homology to detect hopf bifurcations in dynamical systems," *Algorithms*, vol. 13, p. 278, oct 2020.

[3] M. C. Yesilli, F. A. Khasawneh, and A. Otto, "Chatter detection in turning using machine learning and similarity measures of time series via dynamic time warping," *Journal of Manufacturing Processes*, vol. 77, pp. 190–206, 2022.

[4] M. Carriere, F. Chazal, M. Glisse, Y. Ike, and H. Kannan, "Optimizing persistent homology based functions," in *International conference on machine learning*, 10 2020.

[5] J. Leygonie, S. Oudot, and U. Tillmann, "A framework for differential calculus on persistence barcodes," *Foundations of Computational Mathematics*, vol. 22, pp. 1069–1131, 7 2021.

[6] M. Gameiro, Y. Hiraoka, and I. Obayashi, "Continuation of point clouds via persistence diagrams," *Physica D: Nonlinear Phenomena*, vol. 334, pp. 118–132, 11 2016.

[7] M. M. Chumley, M. C. Yesilli, J. Chen, F. A. Khasawneh, and Y. Guo, "Pattern characterization using topological data analysis: Application to piezo vibration striking treatment," *Precision Engineering*, vol. 83, pp. 42–57, Sept. 2023.

[8] M. C. Yesilli, M. M. Chumley, J. Chen, F. A. Khasawneh, and Y. Guo, "Exploring Surface Texture Quantification in Piezo Vibration Striking Treatment (PVST) Using Topological Measures," *International Manufacturing Science and Engineering Conference*, vol. Volume 2: Manufacturing Processes; Manufacturing Systems, 06 2022. V002T05A061.

[9] M. M. Chumley, F. A. Khasawneh, A. Otto, and T. Gedeon, "A nonlinear delay model for metabolic oscillations in yeast cells," *Bulletin of Mathematical Biology*, vol. 85, Nov. 2023.

[10] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[11] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade: Second Edition*, pp. 421–436, Springer, 2012.

[12] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, 2019.

[13] M. J. Powell, "Direct search algorithms for optimization calculations," *Acta numerica*, vol. 7, pp. 287–336, 1998.

[14] W. Spendley, G. R. Hext, and F. R. Himsworth, "Sequential application of simplex designs in optimisation and evolutionary operation," *Technometrics*, vol. 4, no. 4, pp. 441–461, 1962.

[15] S. C. Endres, C. Sandrock, and W. W. Focke, "A simplicial homology algorithm for lipschitz optimisation," *Journal of Global Optimization*, vol. 72, pp. 181–217, 2018.

[16] T. M. Ragonneau and Z. Zhang, "Pdfo–a cross-platform package for powell's derivative-free optimization solver," *arXiv preprint arXiv:2302.13246*, 2023.

[17] M. J. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[18] M. J. Powell, "Uobyqa: unconstrained optimization by quadratic approximation," *Mathematical Programming*, vol. 92, no. 3, pp. 555–582, 2002.

[19] M. J. Powell, "The newuoa software for unconstrained optimization without derivatives," *Large-scale nonlinear optimization*, pp. 255–297, 2006.

[20] M. J. Powell, "Developments of newuoa for minimization without derivatives," *IMA journal of numerical analysis*, vol. 28, no. 4, pp. 649–664, 2008.

[21] M. J. Powell *et al.*, "The bobyqa algorithm for bound constrained optimization without derivatives," *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, vol. 26, 2009.

[22] M. J. Powell, "On fast trust region methods for quadratic models with linear constraints," *Mathematical Programming Computation*, vol. 7, pp. 237–267, 2015.

[23] "Chapter vi - vector optimization," in *Mathematics of Optimization* (G. Giorgi, A. Guerraggio, and J. Thierfelder, eds.), pp. 503–591, Amsterdam: Elsevier Science, 2004.

[24] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, pp. 369–395, 2004.

[25] W. Chen, A. Sahai, A. Messac, and G. J. Sundararaj, "Exploration of the effectiveness of physical programming in robust design," *J. Mech. Des.*, vol. 122, no. 2, pp. 155–163, 2000.

[26] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE transactions on Automatic Control*, vol. 8, no. 1, pp. 59–60, 1963.

[27] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2002.

[28] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*. Springer, Jan. 2004.

[29] R. Ghrist, "Barcodes: The persistent topology of data," *Builletin of the American Mathematical Society*, vol. 45, pp. 61–75, 2008. Survey.

[30] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, pp. 255–308, 1 2009. Survey.

[31] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction*. Rhode Island: American Mathematical Society, 2010.

[32] K. Mischaikow and V. Nanda, "Morse theory for filtrations and efficient computation of persistent homology," *Discrete & Computational Geometry*, vol. 50, no. 2, pp. 330–353, 2013.

[33] S. Y. Oudot, *Persistence theory: from quiver representations to data analysis*, vol. 209 of *AMS Mathematical Surveys and Monographs*. Rhode Island: American Mathematical Soc., 2017.

[34] E. Munch, "A user's guide to topological data analysis," *Journal of Learning Analytics*, vol. 4, pp. 47–61, jul 2017.

[35] H. Chintakunta, T. Gentimis, R. Gonzalez-Diaz, M.-J. Jimenez, and H. Krim, "An entropy-based persistence barcode," *Pattern Recognition*, vol. 48, no. 2, pp. 391–401, 2015.

[36] P. T. Schrader, "Topological multimodal sensor data analytics for target recognition and information exploitation in contested environments," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXXII*, vol. 12547, pp. 114–143, SPIE, 2023.

[37] F. J. Montáns, F. Chinesta, R. Gómez-Bombarelli, and J. N. Kutz, "Data-driven modeling and learning in science and engineering," *Comptes Rendus Mécanique*, vol. 347, no. 11, pp. 845–855, 2019.

[38] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003.

[39] A. Mouraud, "Innovative time series forecasting: auto regressive moving average vs deep networks," *Entrepreneurship and Sustainability Issues*, vol. 4, no. 3, p. 282, 2017.

[40] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019.

[41] K. Yeo, "Data-driven reconstruction of nonlinear dynamics from sparse observation," *Journal of Computational Physics*, vol. 395, pp. 671–689, 2019.

[42] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pp. 1394–1401, Ieee, 2018.

[43] G. A. Gottwald and S. Reich, "Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation," *Physica D: Nonlinear Phenomena*, vol. 423, p. 132911, Sept. 2021.

[44] G. A. Gottwald and S. Reich, "Combining machine learning and data assimilation to forecast dynamical systems from noisy partial observations," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 10, 2021.

[45] Z. Zhang and J. C. Moore, "Chapter 9 - data assimilation," in *Mathematical and Physical Fundamentals of Climate Change* (Z. Zhang and J. C. Moore, eds.), pp. 291–311, Boston: Elsevier, 2015.

[46] G. Evensen, F. C. Vossepoel, and P. J. van Leeuwen, *Data assimilation fundamentals: A unified formulation of the state and parameter estimation problem*. Springer Nature, 2022.

[47] E. Blasch, S. Ravela, and A. Aved, *Handbook of Dynamic Data Driven Applications Systems*, vol. 1. Springer, 01 2018.

[48] E. Blasch, "Dddas advantages from high-dimensional simulation," in *2018 Winter Simulation Conference (WSC)*, pp. 1418–1429, IEEE, 2018.

[49] L. Li, F.-X. Le Dimet, J. Ma, and A. Vidard, "A level-set-based image assimilation method: Potential applications for predicting the movement of oil spills," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 11, pp. 6330–6343, 2017.

[50] S. Cheng, I. C. Prentice, Y. Huang, Y. Jin, Y.-K. Guo, and R. Arcucci, "Data-driven surrogate model with latent data assimilation: Application to wildfire forecasting," *Journal of Computational Physics*, vol. 464, p. 111302, 2022.

[51] A. Albarakati, M. Budišić, and E. S. Van Vleck, "Projected data assimilation using sliding window proper orthogonal decomposition," *Journal of Computational Physics*, vol. 514, p. 113235, 2024.

[52] G. A. Gottwald and I. Melbourne, "The 0-1 test for chaos: A review," in *Chaos Detection and Predictability*, pp. 221–247, Springer Berlin Heidelberg, 2016.

[53] L. Jiang and N. Liu, "Correcting noisy dynamic mode decomposition with kalman filters," *Journal of Computational Physics*, vol. 461, p. 111175, 2022.

[54] A. Bryson and D. Johansen, "Linear filtering for time-varying systems using measurements containing colored noise," *IEEE Transactions on Automatic Control*, vol. 10, no. 1, pp. 4–10, 1965.

[55] A. Zare, "Data-enhanced kalman filtering of colored process noise," in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 6603–6607, IEEE, 2021.

[56] J. R. Tempelman, A. Myers, J. T. Scruggs, and F. A. Khasawneh, "Effects of correlated noise on the performance of persistence based dynamic state detection methods," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 83969, p. V007T07A023, American Society of Mechanical Engineers, 2020.

[57] L. Li, A. Vidard, F.-X. Le Dimet, and J. Ma, "Topological data assimilation using wasserstein distance," *Inverse Problems*, vol. 35, no. 1, p. 015006, 2018.

[58] C. Chatfield, *Time-Series Forecasting*. Chapman and Hall/CRC, Oct. 2000.

[59] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems* (J. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), vol. 20, Curran Associates, Inc., 2007.

[60] R. C. Staudemeyer and E. R. Morris, "Understanding lstm – a tutorial into long short-term memory recurrent neural networks," 2019.

[61] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[62] M. Pandey and B. Pal, "Adaptation to best fit learning rate in batch gradient descent," *International Journal of Science and Research (IJSR)*, vol. 3, no. 8, 2014.

[63] G. Schuëller, "Developments in stochastic structural mechanics," *Archive of Applied Mechanics*, vol. 75, pp. 755–773, 2006.

[64] J. Ding, C. Fan, and J. Lin, "Auxiliary model based parameter estimation for dual-rate output error systems with colored noise," *Applied Mathematical Modelling*, vol. 37, no. 6, pp. 4051–4058, 2013.

[65] J. Timmer and M. Koenig, "On generating power law noise.," *Astronomy and Astrophysics, v. 300, p. 707*, vol. 300, p. 707, 1995.

[66] A. Karimi and M. R. Paul, "Extensive chaos in the lorenz-96 model," *Chaos: An interdisciplinary journal of nonlinear science*, 2009.

[67] E. Ott, B. R. Hunt, I. Szunyogh, A. V. Zimin, E. J. Kostelich, M. Corazza, E. Kalnay, D. J. Patil, and J. A. Yorke, "A local ensemble kalman filter for atmospheric data assimilation," 2002.

[68] A. Bapat, P. B. Salunkhe, and A. V. Patil, "Hall-effect thrusters for deep-space missions: A review," *IEEE Transactions on Plasma Science*, vol. 50, no. 2, pp. 189–202, 2022.

[69] K. Hara, "An overview of discharge plasma modeling for hall effect thrusters," *Plasma Sources Science and Technology*, vol. 28, no. 4, p. 044001, 2019.

[70] J. M. Fife, *Hybrid-PIC modeling and electrostatic probe survey of Hall thrusters*. PhD thesis, Massachusetts Institute of Technology, 1998.

[71] B. D. Smith, I. D. Boyd, H. Kamhawi, and W. Huang, "Hybrid-pic modeling of a high-voltage, high-specific-impulse hall thruster," in *49th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, American Institute of Aeronautics and Astronautics, July 2013.

[72] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*. Cambridge: Cambridge University Press, nov 2004.

[73] M. W. Hirsch, S. Smale, and R. Devaney, *Differential Equations, Dynamical Systems, and an Introduction to Chaos (Pure and Applied Mathematics (Academic Press), 60.)*. Academic Press, 2003.

[74] G. L. Baker and J. P. Gollub, *Chaotic Dynamics*. Cambridge University Press, 1 1996.

[75] H. Sayama, *Introduction to the modeling and analysis of complex systems*. Open SUNY Textbooks, 2015.

[76] Y. A. Kuznetsov, *Elements of applied bifurcation theory*. New York: Springer, 1998.

[77] R. U. Seydel, *Practical Bifurcation and Stability Analysis*. Springer-Verlag GmbH, Nov. 2009.

[78] H. Dankowicz and F. Schilder, *Recipes for Continuation*. Society for Industrial and Applied Mathematics, 5 2013.

[79] J. Sieber and B. Krauskopf, "Control based bifurcation analysis for experiments," *Nonlinear Dynamics*, vol. 51, pp. 365–377, 2 2007.

[80] J. Sieber, B. Krauskopf, D. Wagg, S. Neild, and A. Gonzalez-Buelga, "Control-based continuation of unstable periodic orbits," *Journal of Computational and Nonlinear Dynamics*, vol. 6, 9 2010.

[81] D. A. Barton, B. P. Mann, and S. G. Burrow, "Control-based continuation for investigating nonlinear experiments," *Journal of Vibration and Control*, vol. 18, pp. 509–520, 2 2011.

[82] E. Bureau, F. Schilder, I. F. Santos, J. J. Thomsen, and J. Starke, "Experimental bifurcation analysis of an impact oscillator—tuning a non-invasive control scheme," *Journal of Sound and Vibration*, vol. 332, pp. 5883–5897, 10 2013.

[83] D. A. W. Barton and J. Sieber, "Systematic experimental exploration of bifurcations with noninvasive control," *Physical Review E*, vol. 87, p. 052916, 5 2013.

[84] D. A. Barton, "Control-based continuation: Bifurcation and stability analysis for physical experiments," *Mechanical Systems and Signal Processing*, vol. 84, pp. 54–64, 2 2017.

[85] S. Godwin, D. Ward, E. Pedone, M. Homer, A. G. Fletcher, and L. Marucci, "An extended model for culture-dependent heterogenous gene expression and proliferation dynamics in mouse embryonic stem cells," *npj Systems Biology and Applications*, vol. 3, 8 2017.

[86] B. Krauskopf, H. M. Osinga, E. J. Doedel, M. E. Henderson, J. Guckenheimer, A. Vladimirsky, M. Dellnitz, and O. Junge, "A survey of methods for computing (un)stable manifolds of vector fields," in *World Scientific Series on Nonlinear Science Series B*, pp. 67–95, World Scientific, 3 2006.

[87] M. Peeters, R. Viguié, G. Sérandour, G. Kerschen, and J.-C. Golinval, "Nonlinear normal modes, part II: Toward a practical computation using numerical continuation techniques," *Mechanical Systems and Signal Processing*, vol. 23, pp. 195–216, 1 2009.

[88] S. Huntley, D. Jones, and A. Gaitonde, "Bifurcation tracking for high reynolds number flow around an airfoil," *International Journal of Bifurcation and Chaos*, vol. 27, p. 1750061, 4 2017.

[89] C. Kuehn, "Exploring parameter spaces in dynamical systems," 2008.

[90] N. Atienza, R. Gonzalez-Díaz, and M. Soriano-Trigueros, "On the stability of persistent entropy and new summary functions for topological data analysis," *Pattern Recognition*, vol. 107, p. 107509, 2020.

[91] A. Myers, E. Munch, and F. A. Khasawneh, "Persistent homology of complex networks for dynamic state detection," *Physical Review E*, vol. 100, p. 022314, 8 2019.

[92] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," 2018.

[93] B. C. Goodwin, "Oscillatory behavior in enzymatic control processes," *Advances in Enzyme Regulation*, vol. 3, pp. 425–437, Jan. 1965.

[94] P. Ruoff and L. Rensing, "The temperature-compensated goodwin model simulates many circadian clock properties," *Journal of Theoretical Biology*, vol. 179, pp. 275–285, Apr. 1996.

[95] D. Gonze and W. Abou-Jaoudé, "The goodwin model: Behind the hill function," *PLoS ONE*, vol. 8, p. e69573, Aug. 2013.

[96] R. E. F. Harper, *"Time Flies": Multisensory Processing by Circadian Clocks in Drosophila Melanogaster*. PhD thesis, 2017. AAI28195801.

[97] R. Genesio, G. Innocenti, and F. Gualdani, "A global qualitative view of bifurcations and dynamics in the rössler system," *Physics Letters A*, vol. 372, pp. 1799–1809, Mar. 2008.

[98] J. Zhang, T. He, S. Sra, and A. Jadbabaie, "Why gradient clipping accelerates training: A theoretical justification for adaptivity," 2019.

[99] R. Bischof and M. Kraus, "Multi-objective loss balancing for physics-informed deep learning," 2021.

[100] B. Xiao, "Strategies for balancing multiple loss functions in deep learning." Medium, 2024.

[101] A. Myers and F. A. Khasawneh, "Dynamic state analysis of a driven magnetic pendulum using ordinal partition networks and topological data analysis," in *Volume 7: 32nd Conference on Mechanical Vibration and Noise (VIB)*, American Society of Mechanical Engineers, aug 2020.

[102] I. B. Masokano, W. Liu, S. Xie, D. F. H. Marcellin, Y. Pei, and W. Li, "The application of texture quantification in hepatocellular carcinoma using ct and mri: a review of perspectives and challenges," *Cancer Imaging*, 2020.

[103] I. Ymeti, D. Shrestha, V. Jetten, C. Lievens, and and, "Using color, texture and object-based image analysis of multi-temporal camera data to monitor soil aggregate breakdown," *Sensors*, vol. 17, p. 1241, may 2017.

[104] F. Gao and Y. Lu, "Moving target detection using inter-frame difference methods combined with texture features and lab color space," in *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pp. 76–81, 2019.

[105] T. Thomas, "Trends in surface roughness," *International Journal of Machine Tools and Manufacture*, vol. 38, pp. 405–411, may 1998.

[106] A. Spierings, T. Starr, and K. Wegener, "Fatigue performance of additive manufactured metallic parts," *Rapid Prototyping Journal*, vol. 19, pp. 88–94, 3 2013.

[107] W. E. Frazier, "Metal additive manufacturing: A review," *Journal of Materials Engineering and Performance*, vol. 23, pp. 1917–1928, apr 2014.

[108] K. S. Chan, M. Koike, R. L. Mason, and T. Okabe, "Fatigue life of titanium alloys fabricated by additive layer manufacturing techniques for dental implants," *Metallurgical and Materials Transactions A*, vol. 44, pp. 1010–1022, oct 2012.

[109] H. Yin and T. Emi, "Marangoni flow at the gas/melt interface of steel," *Metallurgical and Materials Transactions B*, vol. 34, pp. 483–493, 10 2003.

[110] D. Gu and Y. Shen, "Balling phenomena in direct laser sintering of stainless steel powder: Metallurgical mechanisms and control methods," *Materials & Design*, vol. 30, pp. 2903–2910, sep 2009.

[111] D. Gu, *Laser Additive Manufacturing of High-Performance Materials*. Springer Berlin Heidelberg, 2015.

[112] Y. Liu, L. Guo, H. Gao, Z. You, Y. Ye, and B. Zhang, "Machine vision based condition monitoring and fault diagnosis of machine tools using information from machined surface texture: A review," *Mechanical Systems and Signal Processing*, vol. 164, p. 108068, 2022.

[113] O. O. Khalifa, A. Densibali, and W. Faris, "Image processing for chatter identification in machining processes," *The International Journal of Advanced Manufacturing Technology*, vol. 31, pp. 443–449, feb 2006.

[114] N. Lei and M. Soshi, "Vision-based system for chatter identification and process optimization in high-speed milling," *The International Journal of Advanced Manufacturing Technology*, vol. 89, pp. 2757–2769, dec 2016.

[115] M. Szydłowski and B. Powałka, "Chatter detection algorithm based on machine vision," *The International Journal of Advanced Manufacturing Technology*, vol. 62, pp. 517–528, dec 2011.

[116] D.-D. Li, W.-M. Zhang, Y.-S. Li, F. Xue, and J. Fleischer, "Chatter identification of thin-walled parts for intelligent manufacturing based on multi-signal processing," *Advances in Manufacturing*, vol. 9, pp. 22–33, apr 2020.

[117] M.-Q. Tran, M. Elsisi, and M.-K. Liu, "Effective feature selection with fuzzy entropy and similarity classifier for chatter vibration diagnosis," *Measurement*, vol. 184, p. 109962, nov 2021.

[118] W. Zhu, J. Zhuang, B. Guo, W. Teng, and F. Wu, "An optimized convolutional neural network for chatter detection in the milling of thin-walled parts," *The International Journal of Advanced Manufacturing Technology*, vol. 106, pp. 3881–3895, jan 2020.

[119] N. N. Bhat, S. Dutta, S. K. Pal, and S. Pal, "Tool condition classification in turning process using hidden markov model based on texture analysis of machined surface images," *Measurement*, vol. 90, pp. 500–509, aug 2016.

[120] C. Bradley and Y. Wong, "Surface texture indicators of tool wear - a machine vision approach," *The International Journal of Advanced Manufacturing Technology*, vol. 17, pp. 435–443, apr 2001.

[121] A. Datta, S. Dutta, S. Pal, and R. Sen, "Progressive cutting tool wear detection from machined surface images using voronoi tessellation method," *Journal of Materials Processing Technology*, vol. 213, pp. 2339–2349, dec 2013.

[122] L. Li and Q. An, "An in-depth study of tool wear monitoring technique based on image segmentation and texture analysis," *Measurement*, vol. 79, pp. 44–52, feb 2016.

[123] D. Kerr, J. Pengilley, and R. Garwood, "Assessment and visualisation of machine tool wear using computer vision," *The International Journal of Advanced Manufacturing Technology*, vol. 28, pp. 781–791, may 2005.

[124] M. Danesh and K. Khalili, "Determination of tool wear in turning process using undecimated wavelet transform and textural features," *Procedia Technology*, vol. 19, pp. 98–105, 2015.

[125] A. Kassim, Z. Mian, and M. Mannan, "Connectivity oriented fast hough transform for tool wear monitoring," *Pattern Recognition*, vol. 37, pp. 1925–1933, sep 2004.

[126] K. Zhu and X. Yu, "The monitoring of micro milling tool wear conditions by wear area estimation," *Mechanical Systems and Signal Processing*, vol. 93, pp. 80–91, sep 2017.

[127] K. Stępień, "Research on a surface texture analysis by digital signal processing methods," *Tehnicki Vjesnik-Technical Gazette*, vol. 21, no. 3, pp. 485–493, 2014.

[128] A. J. S. Santiago, A. J. Yuste, J. E. M. Expósito, S. G. Galán, R. P. Prado, J. M. Maqueira, and S. Bruque, "Real-time image texture analysis in quality management using grid computing: an application to the MDF manufacturing industry," *The International Journal of Advanced Manufacturing Technology*, vol. 58, pp. 1217–1225, aug 2011.

[129] X. Xie, "A review of recent advances in surface defect detection using texture analysis techniques," *ELCVIA: electronic letters on computer vision and image analysis*, pp. 1–22, 2008.

[130] Ş. Öztürk and B. Akdemir, "Comparison of edge detection algorithms for texture analysis on glass production," *Procedia-Social and Behavioral Sciences*, vol. 195, pp. 2675–2682, 2015.

[131] V. R. Vijaykumar and S. Sangamithirai, "Rail defect detection using gabor filters with texture analysis," *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, mar 2015.

[132] M. Kilic, S. Hiziroglu, and E. Burdurlu, "Effect of machining on surface roughness of wood," *Building and Environment*, vol. 41, pp. 1074–1078, aug 2006.

[133] N. Myshkin, A. Grigoriev, S. Chizhik, K. Choi, and M. Petrokovets, "Surface roughness and texture analysis in microscale," *Wear*, vol. 254, pp. 1001–1009, jul 2003.

[134] B. Josso, D. R. Burton, and M. J. Lalor, "Frequency normalised wavelet transform for surface roughness analysis and characterisation," *Wear*, vol. 252, pp. 491–500, mar 2002.

[135] B. AlMangour and J.-M. Yang, "Improving the surface quality and mechanical properties by shot-peening of 17-4 stainless steel fabricated by additive manufacturing," *Materials &; Design*, vol. 110, pp. 914–924, nov 2016.

[136] O. Hatamleh, "The effects of laser peening and shot peening on mechanical properties in friction stir welded 7075-t7351 aluminum," *Journal of Materials Engineering and Performance*, vol. 17, pp. 688–694, oct 2008.

[137] Y. Liu, Y. Cao, H. Zhou, X. Chen, Y. Liu, L. Xiao, X. Huan, Y. Zhao, and Y. Zhu, "Mechanical properties and microstructures of commercial-purity aluminum processed by rotational accelerated shot peening plus cold rolling," *Advanced Engineering Materials*, vol. 22, no. 1, p. 1900478, 2020.

[138] E. Maleki and O. Unal, "Shot peening process effects on metallurgical and mechanical properties of 316 l steel via: Experimental and neural network modeling," *Metals and Materials International*, vol. 27, pp. 262–276, sep 2019.

[139] M. Jamalian and D. P. Field, "Effects of shot peening parameters on gradient microstructure and mechanical properties of TRC AZ31," *Materials Characterization*, vol. 148, pp. 9–16, feb 2019.

[140] L. Xie, Y. Wen, K. Zhan, L. Wang, C. Jiang, and V. Ji, "Characterization on surface mechanical properties of ti–6al–4v after shot peening," *Journal of Alloys and Compounds*, vol. 666, pp. 65–70, may 2016.

[141] P. Guo and K. F. Ehmann, "An analysis of the surface generation mechanics of the elliptical vibration texturing process," *International Journal of Machine Tools and Manufacture*, vol. 64, pp. 85–95, jan 2013.

[142] R. Kurniawan, G. Kiswanto, and T. J. Ko, "Micro-dimple pattern process and orthogonal cutting force analysis of elliptical vibration texturing," *International Journal of Machine Tools and Manufacture*, vol. 106, pp. 127–140, jul 2016.

[143] J. Jiang, S. Sun, D. Wang, Y. Yang, and X. Liu, "Surface texture formation mechanism based on the ultrasonic vibration-assisted grinding process," *International Journal of Machine Tools and Manufacture*, vol. 156, p. 103595, sep 2020.

[144] J. Chen, Y. Xu, J. Sandoval, P. Kwon, and Y. Guo, "On force-displacement characteristics and surface deformation in piezo vibration striking treatment (pvst)," *Journal of Manufacturing Science and Engineering*, pp. 1–27, 2021.

[145] M. H. Bharati, J. Liu, and J. F. MacGregor, "Image texture analysis: methods and comparisons," *Chemometrics and Intelligent Laboratory Systems*, vol. 72, pp. 57–71, jun 2004.

[146] G. Srinivasan and G. Shobha, "Statistical texture analysis," in *Proceedings of world academy of science, engineering and technology*, vol. 36, pp. 1264–1269, 2008.

[147] A. Materka, M. Strzelecki, *et al.*, "Texture analysis methods–a review," *Technical university of lodz, institute of electronics, COST B11 report, Brussels*, vol. 10, no. 1.97, p. 4968, 1998.

[148] Z.-Z. Wang and J.-H. Yong, "Texture analysis and classification with linear regression model based on wavelet transform," *IEEE transactions on image processing*, vol. 17, no. 8, pp. 1421–1430, 2008.

[149] F. C. Motta, R. Neville, P. D. Shipman, D. A. Pearson, and R. M. Bradley, "Measures of order for nearly hexagonal lattices," *Physica D: Nonlinear Phenomena*, vol. 380-381, pp. 17–30, oct 2018.

[150] T. K. Dey and Y. Wang, *Computational Topology for Data Analysis*. Cambridge University Press, 2021.

[151] S. Kaji, T. Sudo, and K. Ahara, "Cubical ripser: Software for computing persistent homology of image and volume data," 2020.

[152] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Discrete & Computational Geometry*, vol. 37, pp. 103–120, dec 2006.

[153] D. M. Lane, "Online statistics education." http://onlinestatbook.com/, 4 2013.

[154] SciPy, "Wasserstein distance." https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wasserstein_distance.html, 2008.

[155] M. Arizmendi, A. Jiménez, W. E. Cumbicus, M. Estrems, and M. Artano, "Modelling of elliptical dimples generated by five-axis milling for surface texturing," *International Journal of Machine Tools and Manufacture*, vol. 137, pp. 79–95, feb 2019.

[156] C. Grob and T.-K. Strempel, "On generalizations of conics and on a generalization of the fermat- torricelli problem," *The American Mathematical Monthly*, vol. 105, p. 732, oct 1998.

[157] E. Munch, "Teaspoon." https://github.com/lizliz/teaspoon, 2018.

[158] V. Behravan, "pointcloud2image(x,y,z,numr,numc)." https://www.mathworks.com/matlabcentral/fileexchange/55031-pointcloud2image-x-y-z-numr-numc, 1 2016.

[159] P. Benardos and G.-C. Vosniakos, "Predicting surface roughness in machining: a review," *International Journal of Machine Tools and Manufacture*, vol. 43, pp. 833–844, jun 2003.

[160] International Organization for Standardization, *ISO 4287:1997. Geometrical Product Specifications (GPS) – Surface texture: profile method – terms, definitions and surface texture parameters*, 1997.

[161] International Organization for Standardization, *ISO 25178-2:2012. Geometrical Product Specifications (GPS) – Surface texture: areal – part 2: terms, definitions and surface texture parameters*, 2012.

[162] *Surface texture : surface roughness, waviness, and lay*. New York: American Society of Mechanical Engineers, 2020.

[163] R. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing—from humans to humanoids," *IEEE Transactions on Robotics*, vol. 26, pp. 1–20, feb 2010.

[164] S. Ding, Y. Pan, M. Tong, and X. Zhao, "Tactile perception of roughness and hardness to discriminate materials by friction-induced vibration," *Sensors*, vol. 17, p. 2748, nov 2017.

[165] S. Hossain and S. Serikawa, "Features for texture analysis," in *2012 Proceedings of SICE Annual Conference (SICE)*, pp. 1739–1744, 2012.

[166] S. Facsko, T. Dekorsy, C. Koerdt, C. Trappe, H. Kurz, A. Vogt, and H. L. Hartnagel, "Formation of ordered nanoscale semiconductor dots by ion sputtering," *Science*, vol. 285, pp. 1551–1553, sep 1999.

[167] M. Torres and H. Voorwald, "An evaluation of shot peening, residual stress and stress relaxation on the fatigue life of aisi 4340 steel," *International Journal of Fatigue*, vol. 24, no. 8, pp. 877–886, 2002.

[168] T. Roland, D. Retraint, K. Lu, and J. Lu, "Fatigue life improvement through surface nanostructuring of stainless steel by means of surface mechanical attrition treatment," *Scripta Materialia*, vol. 54, no. 11, pp. 1949–1954, 2006.

[169] H. C. Yildirim and G. B. Marquis, "Fatigue strength improvement factors for high strength steel welded joints treated by high frequency mechanical impact," *International Journal of Fatigue*, vol. 44, pp. 168–176, 2012.

[170] X. Cao, Y. Pyoun, and R. Murakami, "Fatigue properties of a s45c steel subjected to ultrasonic nanocrystal surface modification," *Applied Surface Science*, vol. 256, no. 21, pp. 6297–6303, 2010.

[171] Y. Guo, S. E. Lee, and J. B. Mann, "Piezo-actuated modulation-assisted drilling system with integrated force sensing," *Journal of Manufacturing Science and Engineering*, vol. 139, no. 1, 2017.

[172] Y. Guo and J. B. Mann, "Control of chip formation and improved chip ejection in drilling with modulation-assisted machining," *Journal of Manufacturing Science and Engineering*, vol. 142, no. 7, p. 071001, 2020.

[173] G. Carlsson and A. Zomorodian, "The theory of multidimensional persistence," *Discrete & Computational Geometry*, vol. 42, no. 1, pp. 71–93, 2009.

[174] J. R. Munkres, *Elements of algebraic topology*. CRC press, 2018.

[175] U. Bauer, "Ripser: efficient computation of Vietoris-Rips persistence barcodes," *Journal of Applied and Computational Topology*, 2021.

[176] M. Kuenzi and A. Fiechter, "Changes in carbohydrate composition and trehalase activity during the budding cycle of *Saccharomyces cerevisiae*," *Arch Mikrobiol*, vol. 64, pp. 396–407, 1969.

[177] T. Tu, A. Kudlicki, M. Rowicka, and S. McKnight, "Logic of the yeast metabolic cycle: Temporal compartmentalization of cellular processes," *Science*, vol. 310, pp. 1152–1158, 2005.

[178] N. Slavov, J. Macinskas, A. Caudy, and D. Botstein, "Metabolic cycling without cell division cycling in respiring yeast," *PNAS*, vol. 108, no. 47, p. 19090–19095, 2011.

[179] J. Robertson, C. Stowers, E. Boczko, and C. Johnson, "Real-time luminescence monitoring of cell-cycle and respiratory oscillations in yeast," *PNAS*, vol. 105, no. 46, p. 17988–17993, 2008.

[180] S. Silverman and et al., "Metabolic cycling in single yeast cells from unsynchronized steady-state populations limited on glucose or phosphate," *PNAS*, vol. 107, p. 6946–6951, 2010.

[181] M. Brauer and et. al., "Coordination of growth rate, cell cycle, stress response, and metabolic activity in yeast," *Mol Biol Cell*, vol. 19, pp. 352–367, 2008.

[182] N. Slavov and D. Botstein, "Coupling among growth rate response, metabolic cycle, and cell division cycle in yeast," *Mol Biol Cell*, vol. 22, p. 1997–2009, 2011.

[183] R. Klevecz, J. Bolen, G. Forrest, and D. Murray, "A genomewide oscillation in transcription gates dna replication and cell cycle," *PNAS*, vol. 101, no. 5, pp. 1200–1205, 2004.

[184] M. Jules, J. Francois, and J. Parrou, "Autonomous oscillations in saccharomyces cerevisiae during batch cultures on trehalose," *FEBS J.*, vol. 272, pp. 1490–1500, 2005.

[185] D. Murray, R. Klevecz, and D. Lloyd, "Generation and maintenance of synchrony in saccharomyces cerevisiae continuous culture," *Experimental Cell Research*, vol. 287, pp. 10–15, 2003.

[186] M. A. Henson, "Modeling the sychronization of yeast respiratory oscillations," *Journal of Theoretical Biology*, vol. 231, pp. 443–458, 2004.

[187] H. Y. Sohn and H. Kuriyama, "Ultradian metabolic oscillation of saccharomyces cerevisiae during aerobic continuous culture: Hydrogen sulphide, a population synchronizer, is produced by sulphite reductase," *Yeast*, vol. 18, no. 2, pp. 125–135, 2001.

[188] C. A. Adams, H. Kuriyama, D. Lloyd, and D. B. Murray, "The gts1 protein stabilizes the autonomous oscillator in yeast," *Yeast*, vol. 20, no. 6, pp. 463–470, 2003.

[189] D. Muller, S. Exler, L. Aguilera-Vazquez, E. Guerrero-Martin, and M. Reuss, "Cyclic amp mediates the cell cycle dynamics of energy metabolism in saccharomyces cervisiae," *Yeast*, vol. 20, pp. 351–367, 2003.

[190] E. M. Boczko, T. Gedeon, C. C. Stowers, and T. R. Young, "Ode, rde and sde models of cell cycle dynamics and clustering in yeast," *Journal of biological dynamics*, vol. 4, no. 4, pp. 328–345, 2010.

[191] C. C. Stowers, T. R. Young, and E. M. Boczko, "The structure of populations of budding yeast in response to feedback," *Hypotheses in the Life Sciences*, vol. 1, pp. 71–84, 2011.

[192] L. Morgan, G. Moses, and T. Young, "Coupling of the cell cycle and metabolism in yeast cell-cycle-related oscillations via resource criticality and checkpoint gating," *Letter in Biomathematics*, vol. 5, no. 1, p. 113–128, 2018.

[193] J. J. Woolford and S. Baserga, "Ribosome biogenesis in the yeast saccharomyces cerevisiae," *Genetics*, vol. 195, no. 3, 2013.

[194] M. Scott, S. Klumpp, E. M. Mateescu, and T. Hwa, "Emergence of robust growth laws from optimal regulation of ribosome synthesis," *Mol Syst Biol.*, vol. 10, no. 8, p. 747, 2014.

[195] F. A. Rihan, C. Tunc, S. Saker, S. Lakshmanan, and R. Rakkiyappan, "Applications of delay differential equations in biological systems.," *Complexity*, vol. 2018, pp. NA–NA, 2018.

[196] A. Fowler, "Approximate solution of a model of biological immune responses incorporating delay," *Journal of mathematical biology*, vol. 13, pp. 23–45, 1981.

[197] H. Gulbudak, P. L. Salceanu, and G. S. Wolkowicz, "A delay model for persistent viral infections in replicating cells," *Journal of Mathematical Biology*, vol. 82, no. 7, p. 59, 2021.

[198] G. Huang, Y. Takeuchi, W. Ma, and D. Wei, "Global stability for delay sir and seir epidemic models with nonlinear incidence rate," *Bulletin of mathematical biology*, vol. 72, pp. 1192–1207, 2010.

[199] K. Gopalsamy and B. Aggarwala, "The logistic equation with a diffusionally coupled delay," *Bulletin of Mathematical Biology*, vol. 43, no. 2, pp. 125–140, 1981.

[200] A. Longtin and J. G. Milton, "Modelling autonomous oscillations in the human pupil light reflex using non-linear delay-differential equations," *Bulletin of Mathematical Biology*, vol. 51, no. 5, pp. 605–624, 1989.

[201] G. Rosen, "Time delays produced by essential nonlinearity in population growth models," *Bulletin of mathematical biology*, vol. 49, no. 2, pp. 253–255, 1987.

[202] B. Pell, S. Brozak, T. Phan, F. Wu, and Y. Kuang, "The emergence of a virus variant: dynamics of a competition model with cross-immunity time-delay validated by wastewater surveillance data for covid-19," *Journal of Mathematical Biology*, vol. 86, no. 5, p. 63, 2023.

[203] L. M. y Terán-Romero, M. Silber, and V. Hatzimanikatis, "The origins of time-delay in template biopolymerization processes," *PLoS Computational Biology*, vol. 6, p. e1000726, apr 2010.

[204] T. Gedeon, A. R. Humphries, M. C. Mackey, H.-O. Walther, and Z. Wang, "Operon dynamics with state dependent transcription and/or translation delays," *Journal of Mathematical Biology*, vol. 84, no. 1-2, p. 2, 2022.

[205] "vpasolve - solve symbolic equations numerically." https://www.mathworks.com/help/symbolic/sym.vpasolve.html. Accessed: 2023-06-20.

[206] F. A. Khasawneh and B. P. Mann, "A spectral element approach for the stability analysis of time-periodic delay equations with multiple delays," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, pp. 2129–2141, aug 2013.

[207] Y. A. Kuznetsov, *Numerical Analysis of Bifurcations*, pp. 505–585. New York, NY: Springer New York, 2004.

[208] W.-j. Beyn, A. Champneys, E. Doedel, W. Govaerts, Y. Kuznetsov, and B. Sandstede, "Numerical continuation, and computation of normal forms," vol. 2, 06 1999.

[209] D. Breda, S. Maset, and R. Vermiglio, "Pseudospectral differencing methods for characteristic roots of delay differential equations," *SIAM Journal on Scientific Computing*, vol. 27, no. 2, pp. 482–495, 2005.

[210] G. Stépán, *Retarded dynamical systems: stability and characteristic functions*. London and New York: Longman, co-published with Wiley, 1989.

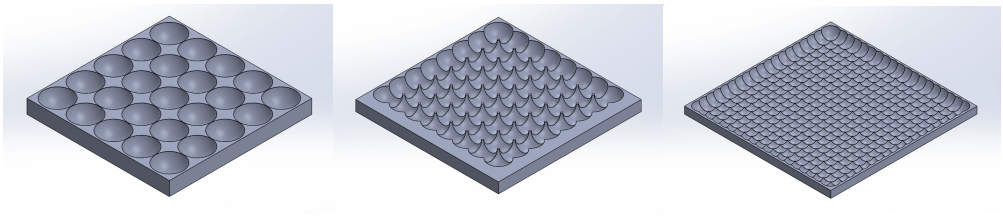
[211] G. Datseris, "Dynamicalsystems.jl: A julia software library for chaos and nonlinear dynamics," *Journal of Open Source Software*, vol. 3, p. 598, mar 2018.

[212] F. A. Khasawneh, D. A. Barton, and B. P. Mann, "Periodic solutions of nonlinear delay differential equations using spectral element method," *Nonlinear dynamics*, vol. 67, pp. 641–658, 2012.

APPENDIX A

VERIFYING THEORETICAL PVST RESULTS

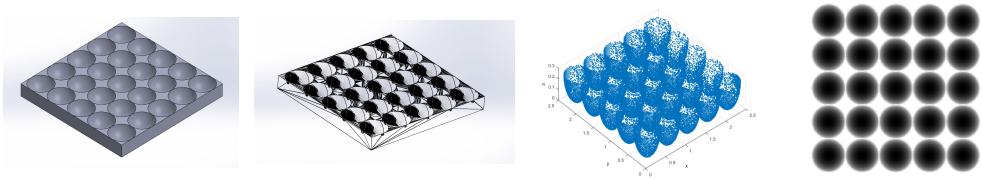
In order to verify the expressions in Section 4.1.2.2, we manufactured gray scale images consisting of perfect PVST strikes in the expected patterns, and computed sublevel persistence to determine whether the results are consistent with the expressions. CAD models were created to model the expected surfaces for 0, 25, and 50% overlap ratios as shown in Fig. A.1. The number of strikes in each case was decided by assuming a 2.5 $\times$ 2.5 mm surface and a striking frequency of 100 Hz. Knowing these two parameters allowed for the in plane speeds to be set to obtain a specified overlap ratio. Note that the model was set up to only allow for full strikes and any fractional strike outside of the 2.5 $\times$ 2.5 mm window was ignored.



(a) 0% Overlap (b) 25% Overlap (c) 50% Overlap

Figure A.1 Ideal PVST grid CAD models at various overlap ratios. (a) 0% overlap, (b) 25% overlap, and (c) 50% overlap.

To compute the sublevel persistence of a nominal texture the surface CAD model needed to be manipulated into a form that was compatible with the cubical ripser. The image pipeline shown in Fig. A.2 was used to convert the CAD information into a gray scale image and subsequently a CSV file for cubical ripser. The CAD model was scaled up by a factor of 10000 to increase the resolution of the point cloud. This was necessary to mitigate the Solidworks STL resolution limitations, but the results were not affected due to the normalization of the points at a later step. A Matlab script was implemented to convert the high resolution STL files into point clouds. Note that the point cloud shown in Fig. A.2c was only plotting one point per 75 points for viewing clarity. After converting the model to a point cloud, the algorithm in [158] was used to convert the point cloud to a gray scale image and a bilinear interpolation created a smooth image as shown in

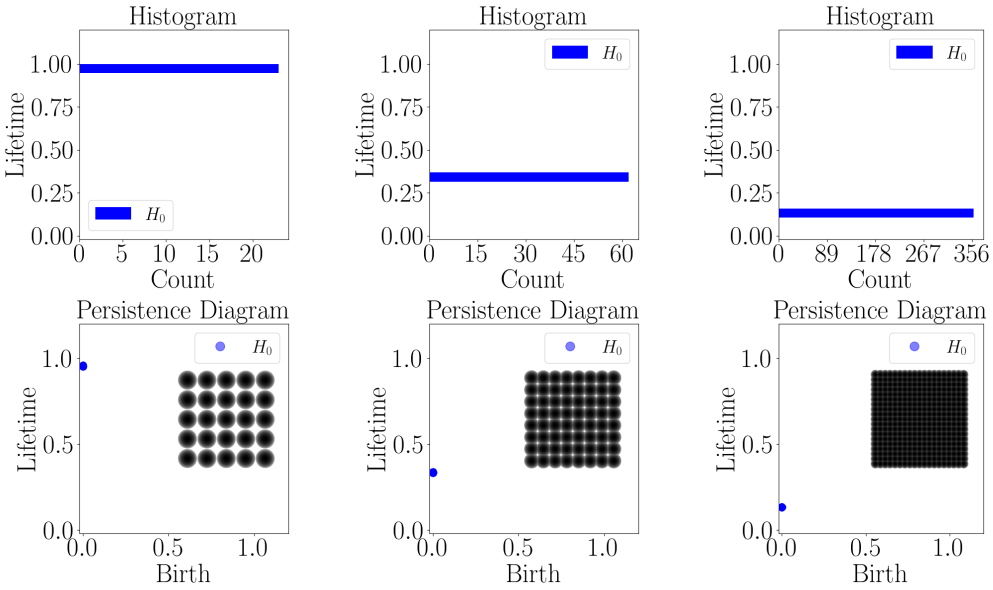


(a) CAD Model (b) STL Conversion (c) Point Cloud (d) Grayscale Image

Figure A.2 Pipeline for converting the PVST grid CAD model into a grayscale image. (a) shows the original CAD model, (b) the resulting STL file, (c) shows the point cloud obtained from the STL file, and (d) the final grayscale depth image.

Fig. A.2d.

Strike Depths: This process was applied to the 0%, 25%, and 50% overlap ratio grids and persistence diagrams were generated for each case as shown in Fig. A.3. Table A.1 shows a com-



(a) 0% Overlap (b) 25% Overlap (c) 50% Overlap

Figure A.3 Nominal CAD surface striking depth persistence diagrams and histograms for each overlap ratio.

parison of the expected lifetimes using the derived results and the results obtained from the CAD model persistence. We see that the lifetimes obtained were exceedingly close to the expected results. The percent differences in each case being below 5% allowed for the theoretical results for the striking depths to be verified and used to compare with the experimental images.

Table A.1 Comparison of the theoretical model lifetimes and the CAD Model generated striking **depth** lifetimes.

Overlap Ratio	0%	25%	50%
Theoretical Lifetime ($\bar{h}$)	1	0.339	0.134
CAD Model Lifetime	0.954	0.337	0.1337
Percent Difference	**4.6%**	**0.5%**	**0.22%**

Strike Roundness: To test the theoretical results for the strike roundness, we threshold the images at two different heights. One height below the critical height and one above to determine if both results are consistent with the expressions.

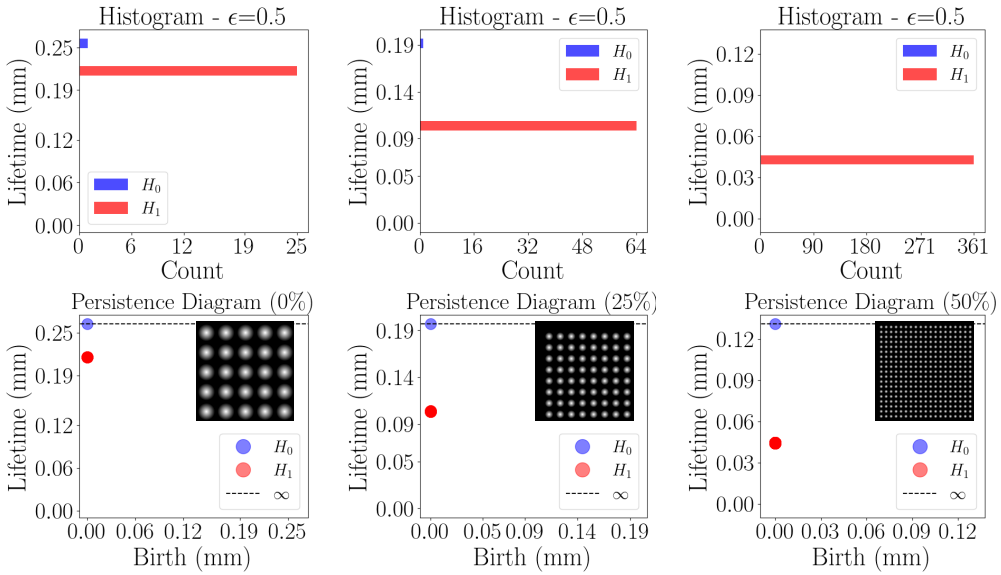
Roundness — No Overlap: First, the images were thresholded at half of the nominal depth ($\varepsilon = 0.5$)

$$T = 0.5h, \tag{A.1}$$

where T is the image threshold height and anything above T is set to black and any pixel below T is set to white. The threshold and distance transform results for the nominal images are shown in Fig. A.4. Because half of the nominal depth was chosen for thresholding, we expect zero overlap in each case. This means that the persistence diagram should have n^2 1D classes born at zero that die at $\sigma = \sqrt{(2R-T)T}$. The corresponding persistence diagrams for the half nominal depth threshold are shown in Fig. A.4. We see that the 1D persistence shows the expected number of loops that are born at time 0 and die at various heights. Using Eq. (4.8), we have converted the pixel distances into distances in mm using $W = 2.55$ mm and $P = 5000$. The results in Table A.2 are exceedingly

Table A.2 Comparison of CAD model persistence results and theoretical strike **roundness** for each overlap ratio ($\varepsilon = 0.5$).

Overlap Ratio	0%	25%	50%
1-D Death [mm]	0.2157	0.1035	0.0445
σ [mm]	0.21651	0.10438	0.04498
Percent Difference	**0.372%**	**0.843%**	**1.068%**

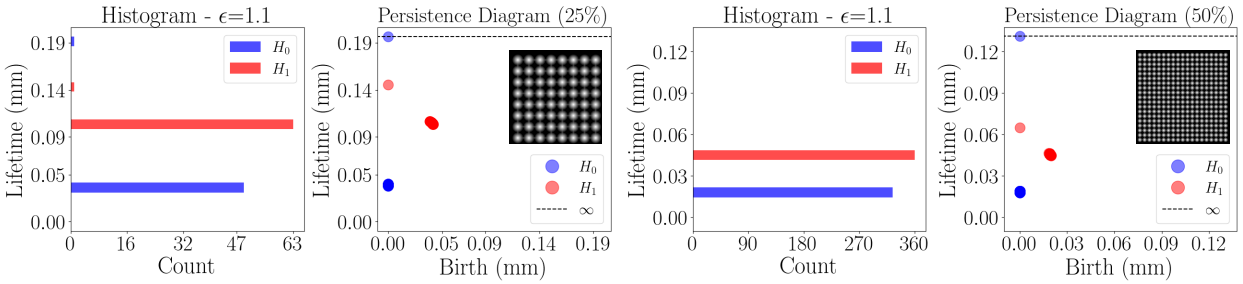


(a) 0% Overlap (b) 25% Overlap (c) 50% Overlap

Figure A.4 CAD model distance transformed image (strike roundness) persistence diagrams for $\varepsilon = 0.5$ ($T < \bar{h}$) at each overlap ratio.

close to the expected values from the theory. This implies that the theoretical model is correct for the case when the circles are not touching.

Roundness — Overlap: To consider a case of overlapping circles, only the 25% and 50% images can be considered. We test the theory for images that contain overlap by computing persistence on the CAD models at $\varepsilon = 1.1$. Figure A.5 shows the thresholded images at this height. The corresponding persistence diagrams for $\varepsilon = 1.1$ are also shown in Fig. A.5. We see that there are $n^2 - 1$ 1D classes born around a that die at the radius σ. The experimental values are compared to the nominal values in Table A.3. It was clear that the results were nearly identical to the theoretical results which verifies the expressions from Section 4.1.2.3.



(a) 25% Overlap (b) 50% Overlap

Figure A.5 CAD model distance transformed image (strike roundness) persistence diagrams for $\varepsilon = 1.1$ ($T > \bar{h}$) at each overlap ratio.

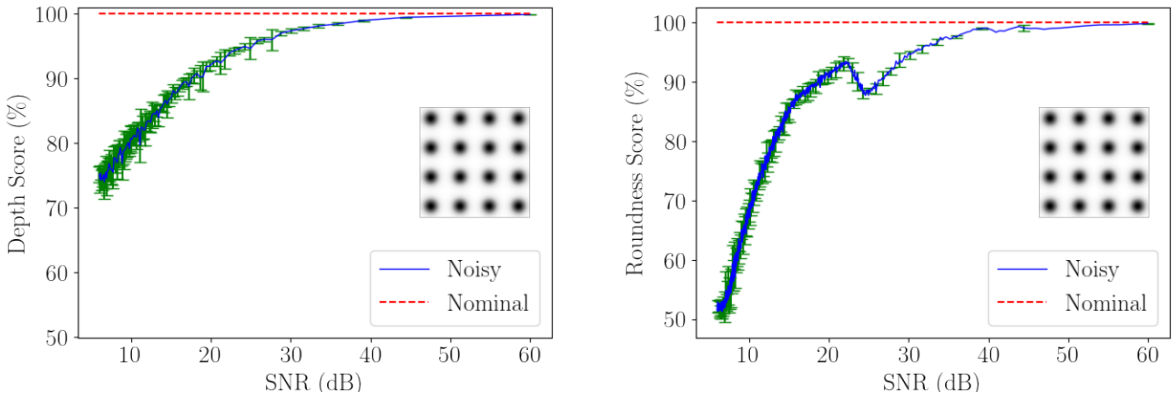
Table A.3 Comparison of CAD model persistence results and theoretical strike **roundness** for each overlap ratio ($\varepsilon = 1.1$).

Overlap Ratio	25%	50%
1-D Birth [mm]	0.0405	0.01916
a [mm]	0.03917	0.01897
Percent Difference	**3.396%**	**1.014%**
1-D Death [mm]	0.1452	0.06480
σ [mm]	0.1459	0.0653
Percent Difference	**0.533%**	**0.788%**

APPENDIX B

TEXTURE ANALYSIS SCORE NOISE STUDY

Feature Depth: A noise study was conducted on the feature depth score by generating a synthetic texture using a superposition of two-dimensional Gaussian distributions in a four by four grid as the features. The synthetic surface is shown in Fig. B.1 (a). Gaussian noise was added to this image by specifying an amplitude on a normal distribution and comparing this amplitude to the nominal strike depth (1) to generate a signal to noise ratio (SNR) in dB. The depth score was then computed and plotted over a range of SNRs to quantify the noise robustness of the score. Ten trials were conducted at each SNR with the average score plotted with error bars indicating one standard deviation from the mean. The resulting plot for the depth score is shown in Fig. B.1 (a). We see that the depth score remains within 5% of the nominal score (100%) for SNRs down to approximately 25 dB.



(a) Depth (b) Roundness

Figure B.1 Texture quantification scores plotted as a function of the SNR in dB. The average score of 10 trials is plotted as a solid line and the dashed line indicates the true score of the feature depths. Error bars are shown at one standard deviation of the 10 trials at each SNR.

Feature Roundness: The roundness score was then computed with varying SNR in the synthetic surface using same process and synthetic surface. Ten trials were conducted for each SNR and the average roundness score was plotted as a function of SNR with error bars indicating one standard deviation from the average shown in Fig. B.1 (b). We see that the roundness score remains within

5% of the nominal score down to approximately 30 dB. It is also clear that the variability in the roundness score is smaller compared to the depth score. This was likely due to each roundness score being made up of 30 earth movers distance computations which reduces the effect of outliers because a single outlier in the earth movers distance plot will not have a significant effect on the area under the curve.

APPENDIX C

ESTIMATING SURFACE SLOPE AND ANGULARITY

During the PVST process, the tool is set to strike the same depth for each cycle. If the sample surface is not perfectly flat relative to the CNC datum, the strike depths will vary across the surface. We see in Fig. 4.11 (c) that the strikes toward the bottom right of the image are deeper in general compared to the opposite corner due to the larger birth times in the top left corner. As a result, we expect that the surface is sloped toward the bottom right corner and we can approximate this slope by fitting a regression plane to the point cloud shown in Fig. 4.11 (c). For this image, the resulting plane has the form,

$$z = 0.1508 - 0.0003315 i_x - 0.0001764 i_y \tag{C.1}$$

where i_x and i_y are the pixel indices in the x and y directions respectively and z is the height in the image. The slope coefficients on the i_x and i_y terms have units of $\frac{1}{\text{pixel}}$ due to the normalization of the depths in the image. The slopes can be converted to units of $\frac{\mu m}{mm}$ using the maximum depth of the image in microns, the width of the image in millimeters and the number of pixels along one axis in the image. The resulting slopes are, $m_x = -0.941$ and $m_y = -0.501$ $\frac{\mu m}{mm}$. In other words, for each millimeter increase in the horizontal direction in this image, we expect the strike depth to increase by about 0.941 microns, This increase corresponds with the top of the surface in this location being closer to the CNC tool. This means that the sample is sloped in the opposite directions. These slopes helped explain why the observed strike depths are deeper toward the bottom right corner of the image and why the image thresholding cannot provide an ideal quantification of the roundness of the strikes and persistence diagram filtering needed to be used to get an optimal reference height. The slopes for the 25 and 50% overlap images are shown in Table C.1. Note that the image coordinate system was used for these slopes so the negative y direction points toward the top of the image.

Table C.1 Measured Radius at Half Nominal Depth.

Direction	m_x $[\mu m/mm]$	m_y $[\mu m/mm]$
0% Overlap	-0.941	-0.501
25% Overlap	-1.211	-2.307
50% Overlap	-0.832	-0.475