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

By

Maxwell Chumley

A DISSERTATION PROPOSAL

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

Mechanical Engineering and Computational Mathematics Science and Engineering—Doctor of Philosophy

2024

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 navigating the parameter space of dynamical systems, and for topological data assimilation. Specifically, in Chapter 1, I show preliminary work on 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. I also explore possibilities for a new topological data assimilation framework, and discuss some promising future applications in time series analysis. In chapter 2, connections to time series representations and attractor reconstruction are presented in my work performing dynamic state detection using persistent homology of network representations of time series signals and new methods for time delay estimation for attractor reconstruction using persistent homology. Other applications of persistent homology are also presented in Chapter 3 where a texture analysis pipeline was developed to quantify specific features of a texture using TDA. Finally, in Chapter 4 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.

ACKNOWLEDGEMENTS

- This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-22-1-0007.

- This work is supported in part by Michigan State University and the National Science Foundation Research Traineeship Program (DGE-2152014) to Maxwell Chumley.

PUBLICATIONS

Current Publications

Journal Papers

1. **Chumley, M. M.**, Khasawneh, F.A., Otto, A., Gedeon, T. (2023). "A Nonlinear Delay Model for Metabolic Oscillations in Yeast Cells." Bulletin of Mathematical Biology, 85, 122.

2. Myers, A. D., **Chumley, M. M.**, Khasawneh, F. A., & Munch, E. (2023). Persistent homology of coarse-grained state-space networks. Physical Review E, 107(3), 034303.

3. **Chumley, M. M.**, Yesilli, M. C., Chen, J., Khasawneh, F. A., & Guo, Y. (2023). Pattern characterization using topological data analysis: Application to piezo vibration striking treatment. Precision Engineering *[Editor's Recommendation]*, 83, 42-57.

Conference Papers

1. Yesilli, M. C., **Chumley, M. M.**, Chen, J., Khasawneh, F. A., & Guo, Y. (2022, June). Exploring surface texture quantification in piezo vibration striking treatment (PVST) using topological measures. In International Manufacturing Science and Engineering Conference (Vol. 85819, p. V002T05A061). American Society of Mechanical Engineers.

Future Publications

Journal Papers

1. Myers, A. D., **Chumley, M.M.**, & Khasawneh, F. A. (2024). Delay parameter selection in permutation entropy using topological data analysis [Under Review].

2. **Chumley, M.M.**, & Khasawneh, F. A. Optimal dynamical system parameter space paths using persistence optimization.

3. **Chumley, M.M.**, & Khasawneh, F. A. Target tracking and forecasting using topological data analysis and data assimilation.

TABLE OF CONTENTS

CHAPTER 0

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 [**?**] and chatter detection in machining [2]. 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 [3–5] that enables a gradient descent optimization of persistence based functions. Overall, my work is organized into the four chapters shown in Fig. 0.1 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. Proposed and current work are presented first with past work being included later.

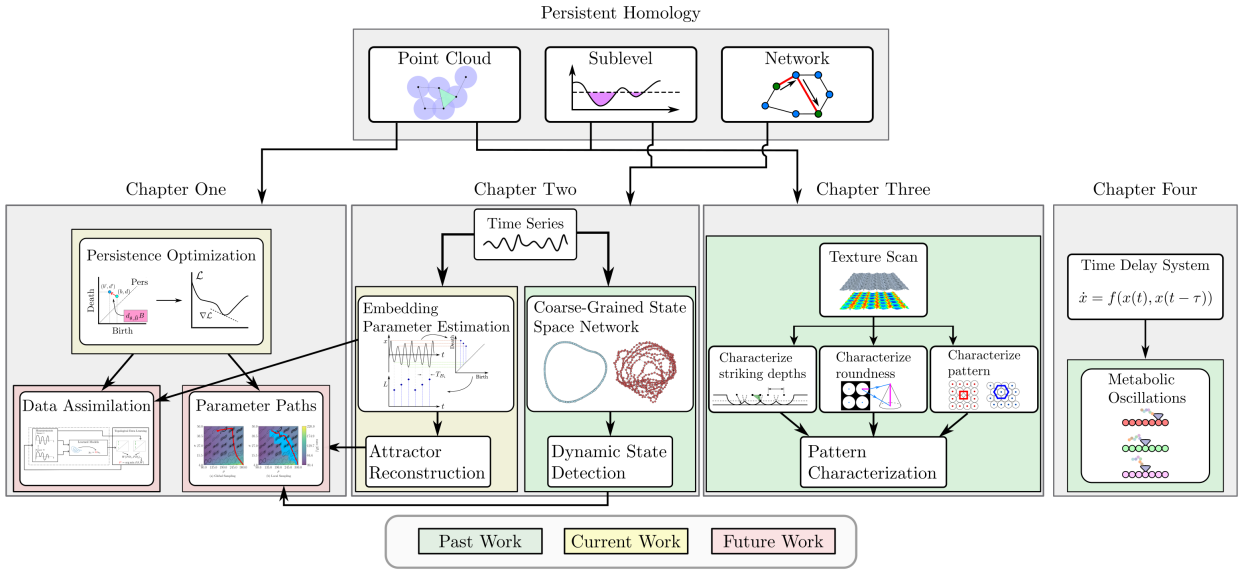


Figure 0.1 Overview of past, current and future work covered in this proposal.

Chapter 1 gives an overview of persistent homology and the differentiation framework on persistence diagrams. This overview is followed by two novel proposed applications of this tool. The first application in Section 1.2 is focused on optimal parameter space navigation of dynamical systems. 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 perfor-

mance characteristics of the system without the need for a model. I present a proposed pipeline for implementing dynamical system parameter spaces into the persistence differentiation framework to allow for the full inverse problem to be solved. Experimental validation plans are also outlined in the context of analyzing Hall Effect Thrusters (HET) data from the Air Force Research Lab (AFRL). For the second application of persistence optimization, I introduce a new data assimilation framework for performing optimal state estimation using persistence optimization in Section 1.3. Time series forecasting methods and data assimilation concepts are reviewed and the methodology for the new framework is presented. Preliminary results demonstrating the effectiveness of persistence optimization in a data assimilation pipeline are also included.

Chapter 2 contains my work on time series representations. In many engineering applications, access to the state space is limited to only a single time series signal that represents a much higher dimensional system. Two common approaches are used for analysis: network representations and time delay embeddings. In Section 2.1, a network representation method called Coarse Grained State Space Networks (CGSSN) are studied using persistent homology of networks to perform dynamic state detection between periodic and chaotic states. This work is published in [6]. Section 2.2 presents novel methods for automatically estimating the Takens embedding delay for a time series using persistent homology. Many methods exist for doing this such as the mutual information and multi-scale permutation entropy methods, however, in practice these methods are not guaranteed to work and can be difficult to tune hyperparameter for the specific system being studied. I aimed to introduce an alternative method using sublevel persistent homology for automatically determining the embedding delay and demonstrate the success of this approach in Section 2.2. The methods in this chapter provide a clear connection to Chapter 1 allowing for extensions of those methods to the case where other representations of signals are implemented.

In Chapter 3, 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 3.1 introduces the methods for quantifying the depth and roundness features and this work is published in [**?**]. For the second paper in Section 3.2, the method for quantifying pattern shape of a texture is presented and this work is published in [**?**].

Lastly, in Chapter 4 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 [**?**].

CHAPTER 1

PERSISTENCE OPTIMIZATION

This chapter contains proposed work of two novel applications using optimization of persistent homology based functions. 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 background theory on persistent homology and persistence optimization in Section 1.1, then I present the first project in Section 1.2, where I aim to introduce a dynamical system parameter space navigation layer to the persistence optimization pipeline to allow for optimally controlling topological properties of a systems response in a data driven approach. In the second project, I introduce a new data assimilation framework using persistence optimization to perform target tracking by optimally assimilating data from many different modalities and models. This work is proposed in Section 1.3.

1.1 Persistence and Optimization Background

Topological data analysis (TDA) quantifies structure in data. One specific form of data in this proposal is a point cloud in $\mathbb{R}^n$. This section provides a brief review of persistent homology. More specifics can be found in [7–14].

1.1.1 Topological Persistence

Homology: If a simplicial complex K is fixed, then homology groups can be used to quantify the holes of the structure in different dimensions. This is done using homology, $H_p(K)$, which is a vector space computed from the complex, with p denoting 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. 1.1(b) where we see that the simplicial complex contains three distinct holes meaning that the rank of the 1D homology at this particular value of the connectivity parameter is 3.

Persistent homology: The main goal is to study the structure of a changing simplicial complex (a generalization of a graph) by measuring its changing homology. Given a real valued function on the simplices of K, such as the one induced on a point cloud $\{x_1, \cdots, x_N\} \subseteq \mathbb{R}^d$ by obtaining the Vietoris-Rips or simply Rips complex (VR). 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 ε. 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 in Fig. 1.1.

The appearance and disappearance of holes in the filtration is encoded in this sequence. This information is then encoded in a persistence diagram, where three holes appear at $\varepsilon = b$ and disappear at increasingly larger ε values d_1, d_2, and d_3 respectively. This information is represented as points in $\mathbb{R}^2$ at (b, d_1), (b, d_2), and (b, d_3). 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

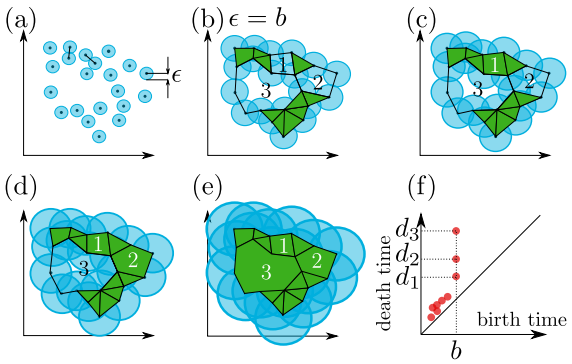


Figure 1.1 Persistence for point clouds. Each snapshot in (a)–(e) shows the rips complex for increasing values of a disc of radius ε. Three prominent loops are formed (or born) at $\varepsilon = b$ in (b), and they start filling (or die) in (c)–(e) at ε values d_1, d_2, and d_3, respectively. These loops are represented in the 1D persistence diagram (f) as (birth,death) pairs (b, d_1), (b, d_2), and (b, d_3). Non-prominent loops form and die quickly as shown by the points near the diagonal.

the diagonal represent structures that persist for a long time, and thus are often considered to be prominent features. See Fig. 1.1(f) for an example persistence diagram.

1.1.2 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 dimension 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. 1.1.2.1, and derivative free methods are explored in Section. 1.1.2.2.

1.1.2.1 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 [15]. 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 [15]. **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 [15]. 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 [16]. 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 desriable local minimum due to the overshooting [15]. 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 [15]. 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 [16].

1.1.2.2 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 [17]. 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 [17].

Direct-Search Methods Within the area of direct-search methods, three types of algorithms are commonly used: line-search, discrete grids, and simplex methods [18]. 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 [18]. 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 [18]. 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 [18]. 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 [18]. 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 $f(\vec{x}_{k+1}) \leq f(\vec{x}_k)$ [18]. The evaluation points are typically chosen by sequentially stepping along the coordinate directions to determine which direction to move [18]. **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 [19] 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 [20] 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 [20].

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 [17]. Many different model-based approaches exist for DFO such as simply using interpolation or regression on output data points of the original function [21]. Many of the commonly used model-based DFO algorithms were developed by Powell [21]. 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 $\|x - x_k\| \leq r_k$ [21]. 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 [21]. 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 [21]. 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) [21]. See [21–27] for more details.

1.1.3 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 [28]. 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 $f_j(\vec{x}) < f_j(\vec{x}_0)$ [28]. 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 [29,30]. 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 $f_i(\vec{x}) < f_i(\vec{x}_0)$ [29]. Many methods exist for solving for Pareto and weakly Pareto optimal points that are summarized in [29], 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 [29] 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 [31]. 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 [29]. 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 [29]. This method is used to generate preliminary results in Section 1.2.3.

1.1.4 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 [3], $E(D) = \sum_{i=1}^{p} |d_i - b_i|$, which gives a measure of how far the persistence pairs

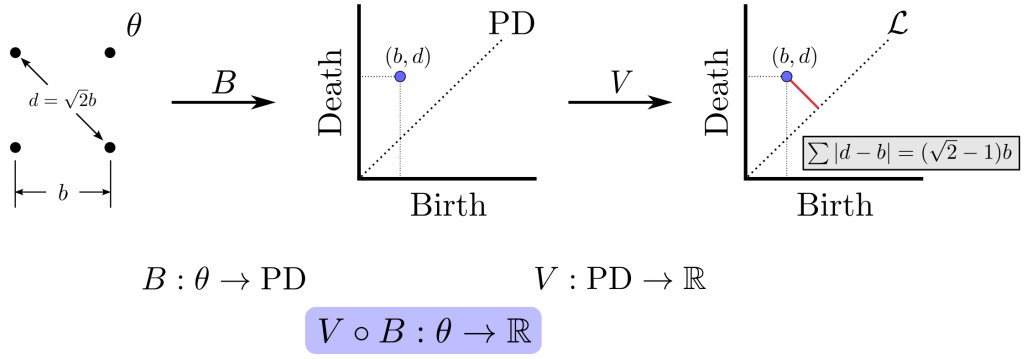


Figure 1.2 Mapping a point cloud θ to a real values persistence feature using the map composition $V \circ B$.

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* [3]. Other examples of functions of persistence include maximum persistence, $E(D) = \max_i |d_i - b_i|$, and persistent entropy $E(D) = -\sum_i p_i \log_2(p_i)$ where $p_i = \frac{\ell_i}{\sum_i \ell_i}$ [32] which gives a measure of order of the persistence diagram. Other features such as the Wasserstein or bottleneck distance are used for measuring dissimilarities between two PDs [3, 33]. In [3], it is specified that in order to have differentiability of the persistence map, the function of persistence must by 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 [4]. An example of this mapping is shown in Fig. 1.2 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 a persistence features. Reference [3] 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 $\mathbb{R}$ 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 $\mathcal{M}$ [3–5]. 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 pairs form the derivative of B with respect to $\tilde{B}$ [4]. This process is pictorially represented using

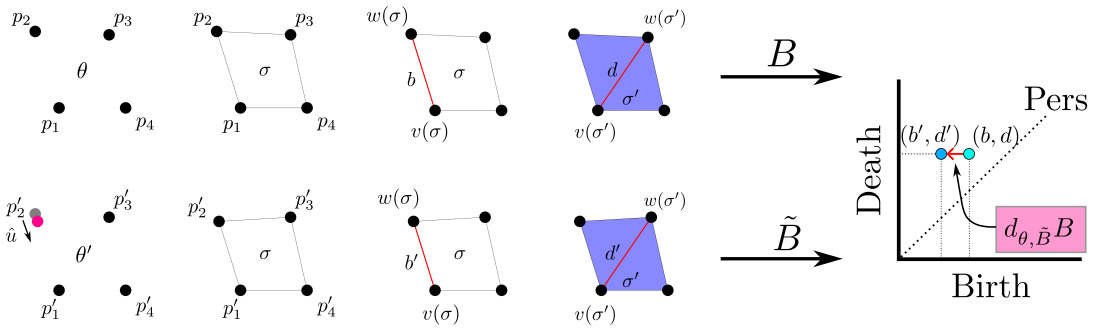


Figure 1.3 Persistence diagram differentiation process. The top row shows the process of tracking the birth and death of the loop from the original point cloud along with using the map B to obtain its persistence diagram. The bottom row performs the same process on a perturbed point cloud and demonstrates how the change in the persistence pair forms the derivative $d_{\theta,\tilde{B}}B$.

a simple point cloud in $\mathbb{R}^2$ consisting of a single loop in Fig. 1.3. 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. 1.3 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 $\tilde{B}$. Notationally, this is represented as $d_{\theta,\tilde{B}}B$ [4].

This process is illustrated more generally and for 0D persistence in the example shown in Fig. 1.4. 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. 1.3, 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 [3, 4]. Attaching edges are the output of the corresponding filter 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 [4] 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 [4]. 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* [4, 5] (i.e., no two points in the cloud coincide or are equidistant). 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 [5]. 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.

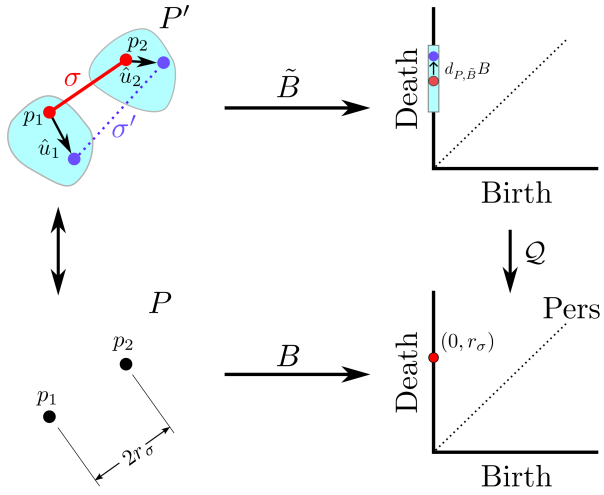


Figure 1.4 Persistence differentiation for point clouds. The point cloud P is perturbed to P' and the 0D persistence diagram is differentiated with respect to this perturbation.

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 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 $\hat{u}$ [4]. Note that the form Eq. (1.2) has been represented for m finite persistence pairs generalizations are presented in [4] 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 [4]. 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 [3] 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 [3] 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 [5] 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. A Newton-Raphson root-finding process is then used to transform the original point cloud into a new point cloud that has the desired homology.

1.1.4.1 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 [3].

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 1.5 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. 1.6. 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.

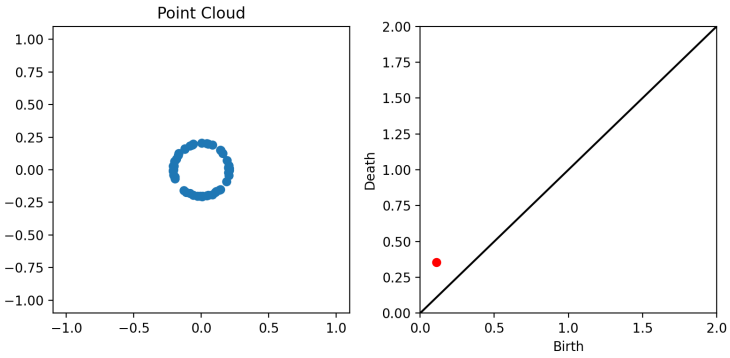


Figure 1.5 Persistence optimization expanding loop example. Initial circular point cloud and 1D persistence diagram.

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 Fig. 1.7 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

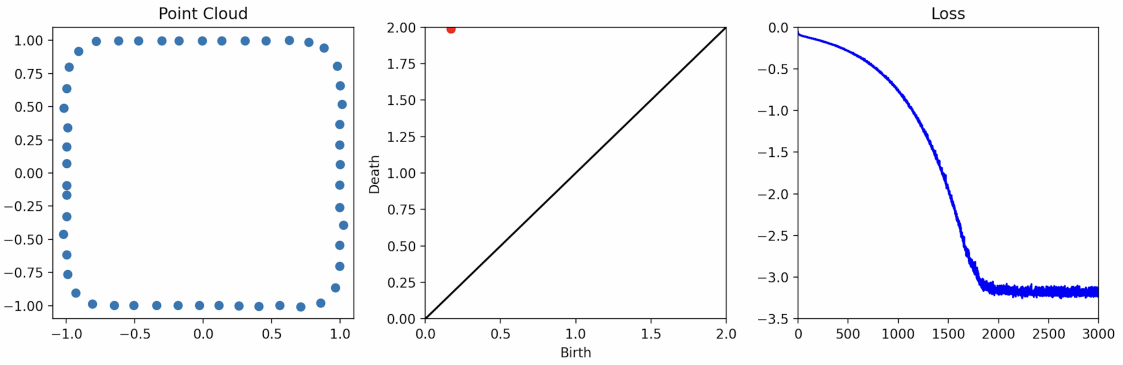


Figure 1.6 Persistence optimization expanding loop example. Resultant point cloud and 1D persistence diagram after 3000 gradient descent steps using the corresponding cost function.

the point cloud and persistence diagram in Fig. 1.8 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. 1.7 and after performing the optimization in this case, the resulting point cloud is shown in Fig. 1.9. 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.

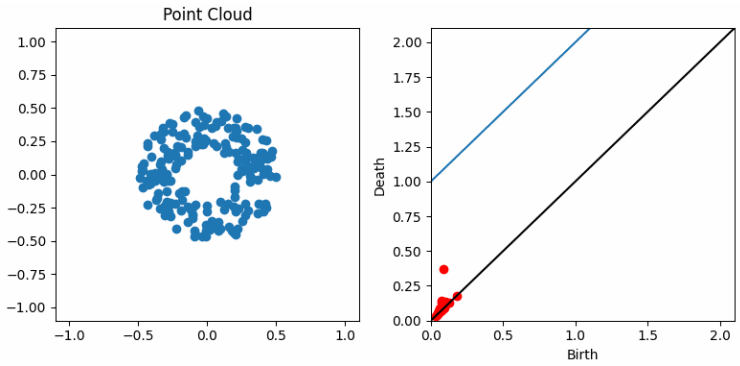


Figure 1.7 Initial point cloud for the second persistence optimization example. The cost function was defined in the same way as the first example with the addition of a term to penalize lifetimes larger than 1 or above the blue line.

1.2 Optimal Parameter Space Navigation

This leads to the first proposed project using persistence optimization for optimal parameter space navigation of dynamical systems. The necessary dynamical systems and bifurcation background are given in Section 1.2.1 and 1.2.2.

1.2.1 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

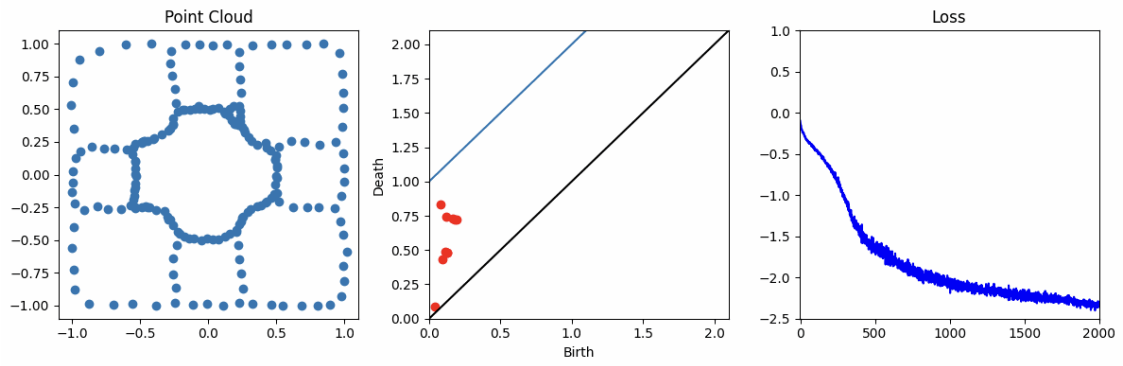


Figure 1.8 Final point cloud for the second persistence optimization example. The cost function was defined in the same way as the first example with the addition of a term to penalize lifetimes larger than 1 or above the blue line.

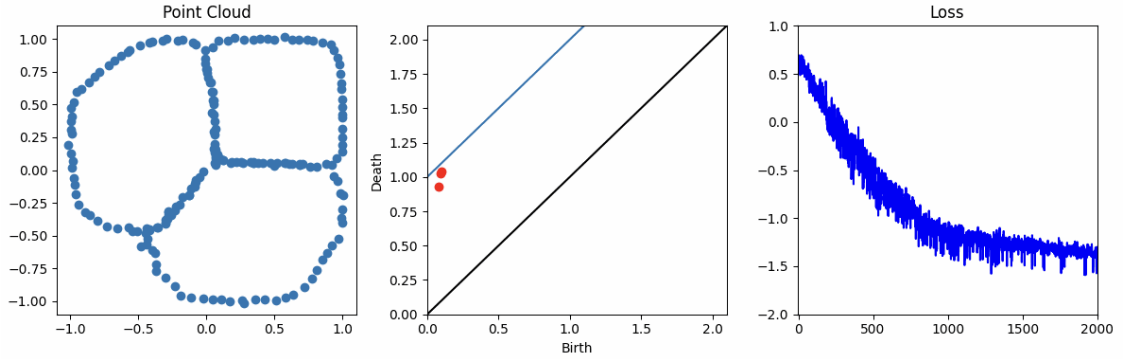


Figure 1.9 Resulting point cloud for the third persistence optimization example. The cost function was defined to maximize loop size while restricting persistence pairs to have a lifetime less than 1. A persistent entropy term was added in this case to give a point cloud with fewer loops.

optimal paths in this space that connect an initial state to a desirable state while avoiding regions of the space that lead to unsafe dynamics. To chart paths in the parameter space for model-free 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 [34–37] for further details.

1.2.2 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 normal operation to an unsafe state. 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.

1.2.3 Research Aim #1: Parameter Space Navigation Using Persistent Homology

Background: 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 [38–40], 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) [41–46]. CBC was successfully used in many scientific domains including biochemistry [47], physics [48], mechanics [49], and fluid dynamics [50]. In this setting, numerical continuation is applied to a feedback-controlled physical experiment such that the control becomes non-invasive [46]. 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.

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. A framework will be created 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 will accomplish this goal by completing three related tasks.

Task I: 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 will assume that there is a target persistence diagram that corresponds to desirable criteria for the system response.

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 as shown in Fig. 1.10(a). 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. 1.10(b) and the state space plot would have localized points to promote steady state stability. The third case shown in Fig. 1.10(c) is an avenue for classifying chaotic behavior using the persistence diagram by computing persistent entropy as in [51]. In the state space this could correspond to a safe region being within a small annular region. I aim to classify many more criterion such as constraining the frequency of the response. I conjecture that using sublevel persistence similar to [1] by controlling the spacing between points, the response frequency can be limited. Together these criteria specified by the user will form the desired characteristics of the target persistence diagram which will be used for intuitive loss function engineering for computing the optimal path in the parameter space.

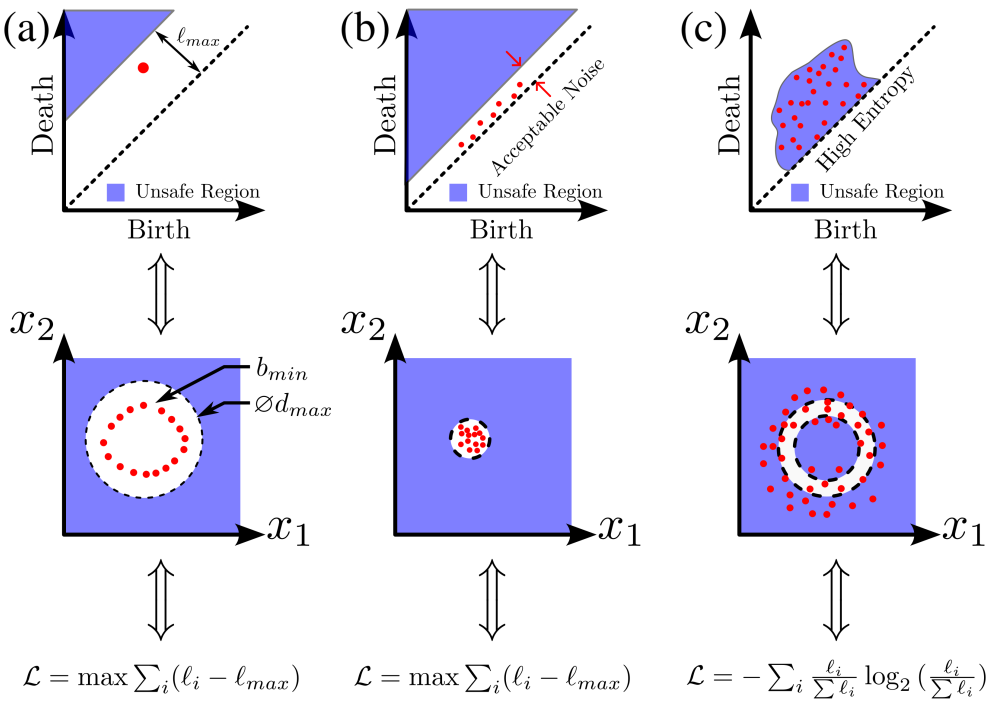


Figure 1.10 Example response criteria mapped into persistence diagrams. (a) Constraint on the amplitude of a periodic solution, (b) Limiting persistent loops to be close to the diagonal to encourage fixed point behavior, and (c) Using high persistent entropy to classify chaotic regions in the persistence diagram.

Task II: Once the desired persistence diagram is identified, 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. For this task, I will research different search algorithms and optimization methods to compute the optimal path that will minimize the cost function to promote the desired topological properties. Given a starting point in the parameter space, there are many ways to determine which direction to move. This task focuses on using scalar derivative-free optimization methods to sample points near the starting position in the parameter space to determine which direction gives a persistence diagram that is *closest* to the desired behavior. To demonstrate the goal of this process, I generated a path in a two dimensional parameter space in an optimal sense to drive the lorenz system to a specified type of solution using topological features of the point clouds as shown in the preliminary work in Section 1.2.4. There I show path generation for two potential sampling methods, however many more sampling methods will be researched such as implementing an adaptive search region that can change shape based on previous sampling information. Specifically, I will test the optimization methods discussed in Sections 1.1.2 to determine which method provides a path that converges to the desired response in the fewest objective function evaluations while adhering to the desired response criteria set from task I. Simple, low dimensional systems with known behavior will be used at this step to benchmark the algorithms for different objectives such as moving away from chaotic regions of the parameter space.

Task III: Scalar optimization methods are suitable when the objective function is carefully chosen to provide the desired response, but persistence diagrams are more complex objects. In order to produce a solution that has a persistence diagram closest to the desired diagram, multi-objective optimization methods should be used. For this task, I will leverage the differentiability of persistence diagrams [3–5] to the map that includes a function from the parameter space to time series generating systems according to $\mathcal{D} \xrightarrow{B'} \mathcal{M} \xrightarrow{B} \text{Pers} \xrightarrow{V} \mathbb{R}$, where B' is the map from the parameter space $\mathcal{D}$ to the time series or state space point cloud. The goal is to check the properties of the map B' such as differentiability conditions to show that $V \circ B \circ B'$ fits within the theoretical framework of [4] ($V \circ B$ already fits with the map types used in [4]).

This will enable a gradient descent approach to moving through the parameter space using the full persistence diagrams to move to a set of parameters that meet the criteria from task I. I will utilize the search algorithms researched in task II in conjunction with the gradient of the persistence diagrams approximate the gradient of B' to locate a direction vector in the parameter space and walk the system to the optimal response. The process is demonstrated graphically in Fig. 1.11 where a 3-dimensional parameter space is shown in Fig. 1.11(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. 1.11(d) and the step is propagated back using gradient descent through the persistence diagram (Fig. 1.11(c)) and state space (Fig. 1.11(b)) to obtain a direction in the parameter space for the next point on the path. The process is continued until a minimum is reached in the cost function. The success of the framework will be evaluated based on speed of computation, convergence, and the ability to reach the target persistence diagram while avoiding unsafe regions.

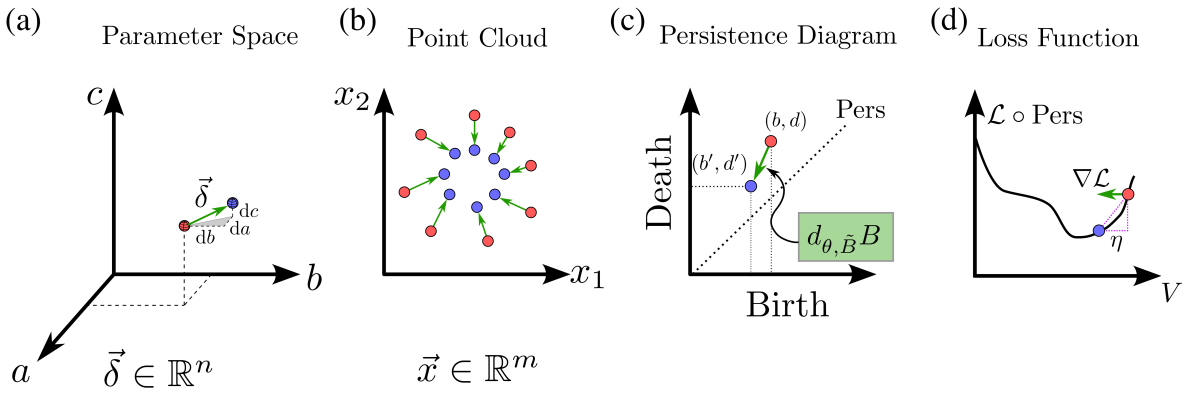
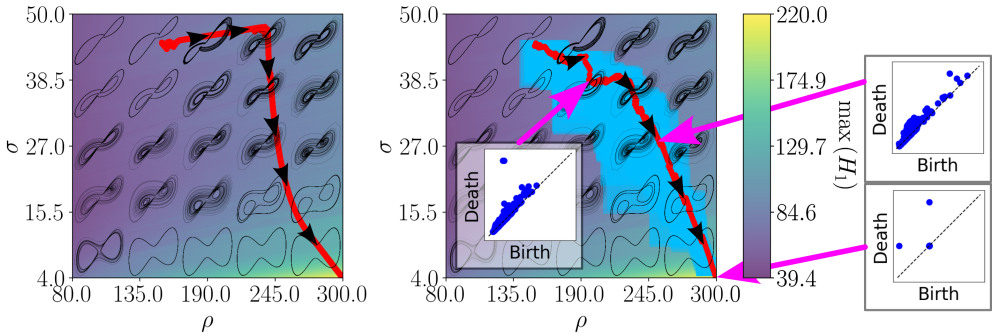


Figure 1.11 Diagram demonstrating the map from the parameter space to the loss function as solving the inverse problem from taking a step against the gradient of the loss function to reach a new set of parameters in the parameter space propagated through the persistence diagram and the state space point cloud.

1.2.4 Preliminary work:

Figure 1.12 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. 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 pa-



(a) Global Sampling (b) Local Sampling

Figure 1.12 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.

rameters and x, y, and z are the system states. For now, I will restrict the parameter space to the plane $\beta = 8/3$ 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. 1.12. 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.

1.2.4.1 Navigation Schemes

To generate the next point along the path towards the target state in Fig. 1.12, 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. 1.12b 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, $\Omega_k = \{(x, y) \in \frac{1}{N}[(N-k)x_0 + kx_{min}, (N-k)x_0 + kx_{max}] \times \frac{1}{N}[(N-k)y_0 + ky_{min}, (N-k)y_0 + ky_{max}] : k = 1 \ldots N\}$ 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. 1.12(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 path found in Section 1.2.4.1 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 Section 1.2.4.1 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, we 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. 1.12(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.

1.2.5 Research Aim #2: Numerical and Experimental Validation

I plan to perform extensive validation studies on the methods developed from the output of research aim 1.

Numerical studies: To validate the path generating algorithms, numerical system simulations will be conducted on nearly 40 dynamical systems with parameter spaces of dimension two or larger in the teaspoon python library [52]. Results will be tested against conventional optimization methods and the noise robustness of the algorithms will be quantified in the process.

Hall-Effect Thrusters: In addition to numerical validation, the proposed framework will be experimentally validated on unique experimental data from Hall-effect Thrusters (HET). This data will be obtained in collaboration with the AFRL Rocket Propulsion Division. The HET data is collected from a sophisticated test chamber with continuous pumping to simulate the vacuum of space. The complexity of this apparatus makes this data unattainable elsewhere. HETs are a class of ion thrusters that generate thrust by accelerating ions through an electromagnetic field to eject

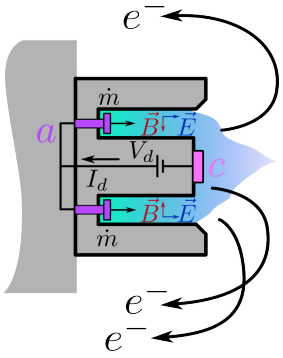


Figure 1.13 Hall-Effect Thruster (HET).

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 accurately simulate. Therefore, data driven methods are essential for analyzing HETs.

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 [53]. 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 [54]. 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 [54]. These operating modes are induced by changes to the system parameters (shown in Fig. 1.13) 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 [53, 54]. Therefore, when maneuvering the space craft by gradually changing the HET parameters, it is important to chart an optimal path in the parameter space that avoids these unsafe/inefficient modes of operation.

1.3 Topological Data Assimilation for Target Tracking

This section introduces a new data assimilation framework for optimal time series forecasting and target tracking using the persistence optimization framework.

1.3.1 Time Series Forecasting

Time series forecasting is a critical component in data assimilation and these concepts are necessary for understanding the data assimilation process. There are many methods for doing this and this section will outline some of the classical approaches.

1.3.1.1 Autoregressive (AR) Forecasting

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

$$X_n = \sum_{i=1}^{p} \varphi_i X_{n-i} + \varepsilon_n, \tag{1.3}$$

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 overfitting [55]. The order of the model is usually determined by the value of p that results in the autocorrelation function being close to zero [55].

1.3.1.2 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,

$$X_n = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{n-i} + \varepsilon_n, \tag{1.4}$$

where μ is the average of the signal and θ_i are the model coefficients [55].

1.3.1.3 Autoregressive Moving Average (ARMA) Forecasting

These two methods can be combined using the AutoRegressive Moving Average or ARMA model [55], which learns both φ and θ coefficients simultaneously and can be represented with a model of the form,

$$X_n = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{n-i} + \sum_{i=1}^{p} \varphi_i X_{n-i} + \varepsilon_n. \tag{1.5}$$

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 [55]. Many extensions to this model have also been introduced such as the AutoRegressive Integrated Moving Average (ARIMA) which allows for nonstationary signals, and the Seasonal AutoRegressive Integrated Moving Average (SARIMA) allows for incorporating known a known period into the forecast [55].

1.3.1.4 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 [56]. This approach has been used for time series forecasting in [57] 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 [57]. 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 [55]. 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

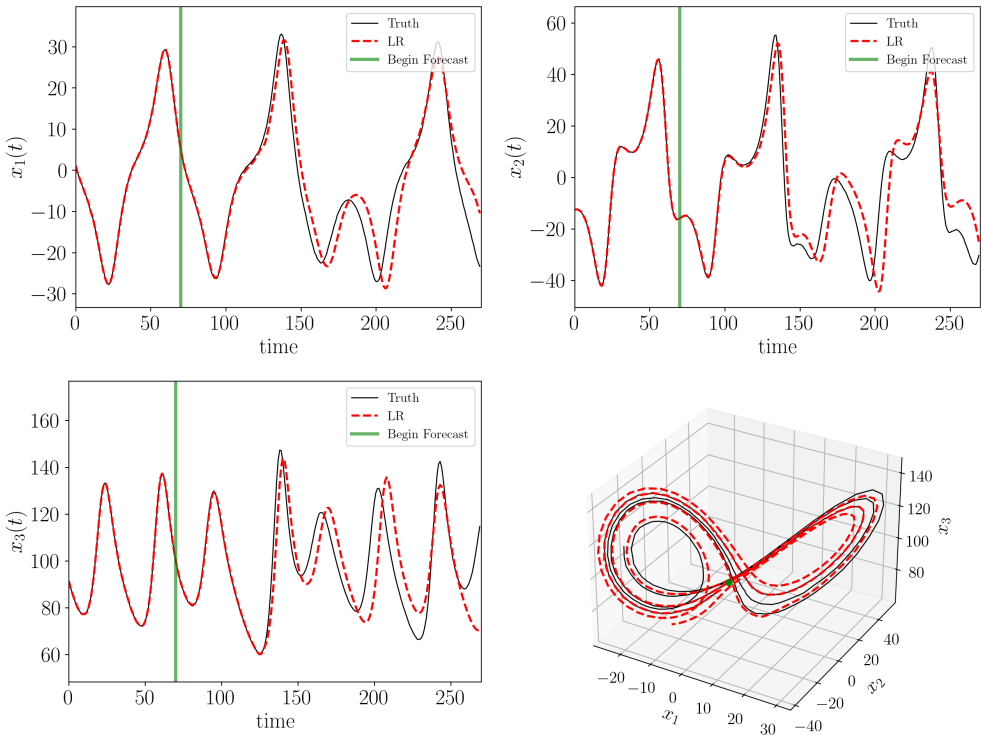


Figure 1.14 Random feature map forecast example using the chaotic lorenz system. the forecast begins at the green vertical line and follows the true trajectory for a period of time before deviating.

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. 1.14. 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.

1.3.2 Data Assimilation

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 [58–61]. Its use has shown successes across many fields such as weather forecasting, oceanography, and predicting the movement of pollution [58, 59, 62]. A common implementation of data assimilation is the Kalman filter applied to dynamical systems where unmeasured system states or noisy measurements are optimally estimated and a system forecast is generated by combining information from all available measurements [57, 59]. This process is demonstrated in Fig. 1.15 where the

observations move the model results closer to the ground truth to improve the predictions. In data assimilation, observed data streams are combined by solving for optimal weights on data streams to place more importance on sources with lower uncertainties typically by minimizing a cost function of the form [58]

$$J(\vec{x}) = (\vec{x}-\vec{x}_b)^T B^{-1}(\vec{x}-\vec{x}_b) + (\vec{y}-h(\vec{x}))^T R^{-1}(\vec{y}-h(\vec{x})), \tag{1.6}$$

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 [58]. Thus finding,

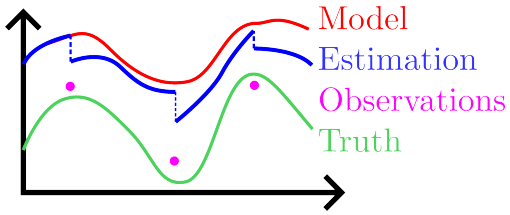


Figure 1.15 Data Assimilation Concept—Improving model results by considering observations to obtain an estimation closer to the ground truth.

$\vec{x}_a = \arg\min J(\vec{x})$, yields an optimal combination of the model predictions and measurement results [58] where x_a is referred to as the analysis solution. It was shown in [58] 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. The distances in Eq. (1.6) are computed using the Euclidean metric. Other considered metrics such as the Wasserstein distance measures pairwise dissimilarity between ground truth and model predictions enabling the use of topologically based cost functions [63]. This is done by generalizing the cost function in Eq. (1.6) using an arbitrary metric of the form, $J(\vec{x}) = d_b(\vec{x},\vec{x}_b) + d_y(\vec{y},h(\vec{x}))$, where d_b is the model discrepancy and d_y is the observation discrepancy.

1.3.3 Topological Data Learning for Target Tracking

I plan to use a data driven approach to obtain a surrogate model of a given targets' trajectory from time series-like data and improve the model based on new observations. It is assumed that the measurements do not span the full state space of the system and that they are infected with noise. Generally surrogate modeling is divided into two broad categories: data-driven and physically-based [64]. The data-driven approach, with access to vast data banks, is selected as the tracking methodology due to target complexities. Many different approaches have been taken in data-driven surrogate modeling such as Bayesian networks, transfer functions, and regression with the results being application dependent [64–66]. Recently, a surrogate modeling approach was combined with a data assimilation algorithm to improve the model in real-time [57, 67]. The method utilizes random feature map forecasting to learn a model for the system at the next time step using ridge regression based on previous noisy measurements.

The method in [67] assumes access to the full state space measurements and it was generalized to the case where only one state is measured in [57]. It assumes that only partial observations are available to the system and employs the Takens embedding to reconstruct the attractor for the system based on a single measurement [57]. A data assimilation algorithm (ensemble Kalman filter)

is then used to optimally estimate the states at the next time step based on the next measurement, and the accurate forecasting times were found to be significantly longer when compared with only using linear regression. The method is referred to as Random Feature Maps and Data Assimilation (RAFDA) [57,67]. This method requires ensemble sampling of the measurements to generate data that can be used to obtain an optimal estimate based on the known noise statistics. While RAFDA integrates data assimilation in the training process and it has been demonstrated that it results in improved forecast horizons with respect to the traditional random feature map forecasting, it will still break down eventually as more measurements stream in. The goal of this research task is to perform data fusion from many different time series-like data modalities to estimate an optimal model for a targets trajectory for forecasting future system states at future times leveraging the inherent connection between dynamical systems and topology. This connection will implement persistence optimization to define a cost function that leads to a new topological data assimilation framework. This goal will be accomplished using the method outlined in 1.3.3.1.

1.3.3.1 Methodology

To efficiently combine data from many different modalities and obtain an accurate model for target tracking, I propose to integrate the random feature map and RAFDA forecasting methods with a new data assimilation scheme driven by TDA to optimally combine learned models from each modality based on noise characteristics of the measurements and improve forecast horizons. More specifically, this method will work by assuming that I have N sensor observations with additive noise. These measurements will then be used for generating a forecast model using any of the methods in Section 1.3.1 such as random feature maps. I will then compute persistence on the measurement data (potentially 0D and 1D) and compute persistence on the forecast model. Using the Wasserstein distance, the dissimilarity between these two diagrams can be quantified and persistence optimization can be used to minimize the Wasserstein distance between them. This will give a new forecast model that has a persistence diagram that is closer to the observation persistence diagram. The new forecast makes up the analysis which is fed back into the loop to be used as the model in the next assimilation window and the process is continued as new observations stream in. The cost functions will be defined in terms of common discrepancy measures between persistence diagrams such as the Wasserstein and bottleneck distances similar to [63] and persistence optimization will be exploited to compute a new, optimal forecast for all of the input time series signals. This will yield a combined model that utilizes information from all input signals and naturally favors the signals that are more significant to the behavior of the system by learning a local minimum of the cost function to filter noise from the signals in the persistence diagrams. As new measurements are obtained, the analysis solution is used as the forecast model in the next assimilation window and the process is repeated. A schematic diagram for this process is shown in Fig. 1.16 where the N measurements are used to generate a forecast model and the two persistence diagrams are compared to yield an optimal forecast for the next measurements.

1.3.4 Preliminary Results

I generated preliminary results for this pipeline by performing persistence optimization using the Wasserstein distance to obtain a point cloud with a target persistence diagram. I started with the circular point cloud shown in Fig. 1.17 where the red persistence pairs indicate the starting 1D persistence diagram and the green point is a constructed target persistence diagram. Using a cost

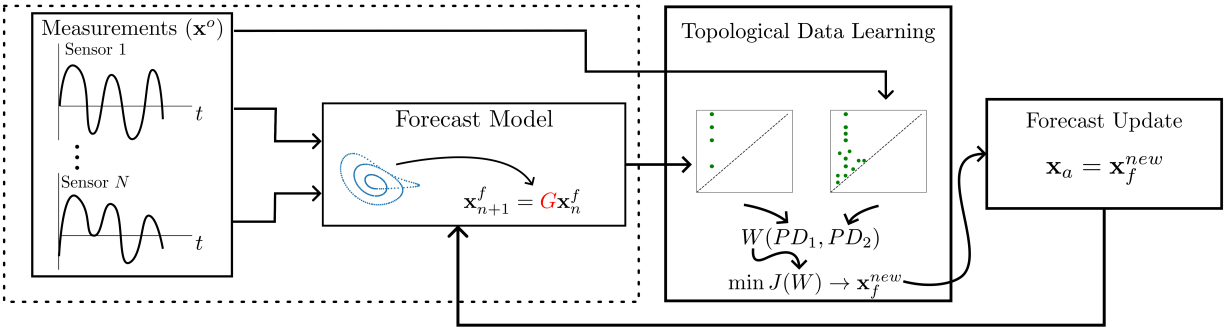


Figure 1.16 Schematic diagram for integrating target tracking models and data fusion/assimilation with topological data analysis using forecasting to estimate models from the data and persistence optimization approaches to optimally combine the signals for the next assimilation window.

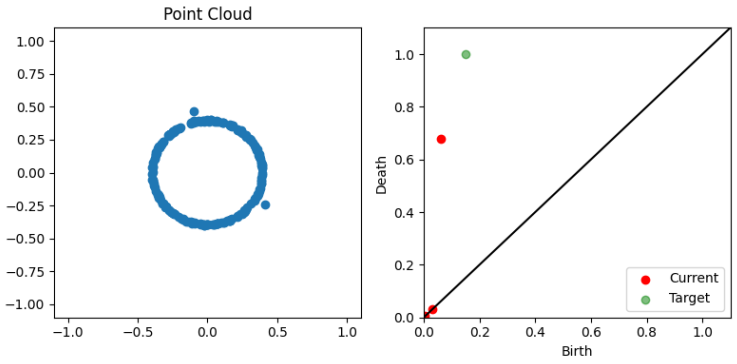


Figure 1.17 Starting point cloud and 1D persistence diagram for optimization using the Wasserstein distance. The target persistence diagram was constructed with the green point and the actual persistence diagram is indicated using red points.

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 result in Fig. 1.18 where the point cloud expanded to reach the target persistence diagram. The same process was performed but the target persistence diagram was modified by adding points near the diagonal to simulate persistence pairs due to noise in the measurements. Using the same cost function and starting point cloud the positions of points were optimized according to the Wasserstein distance. Doing this resulted in Fig. 1.19 where it is clear that the prominent topological features of the target persistence diagram were learned by finding a local minimum of the cost function and filtering out the features due to noise. These results suggest that this method will be successful in a data assimilation framework because the noise will be effectively filtered from the observations.

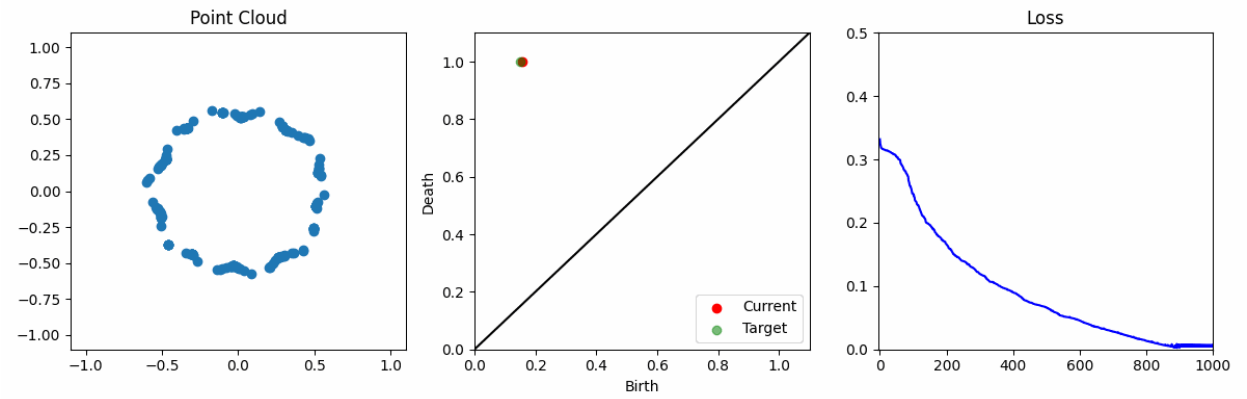


Figure 1.18 Persistence optimization results when minimizing the Wasserstein distance between the model and measurement persistence diagrams with no noise in the target persistence diagram.

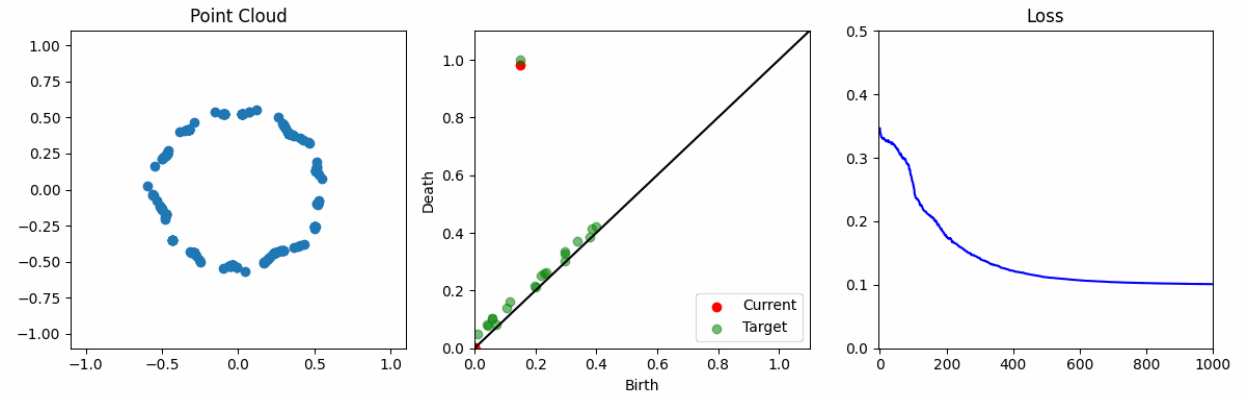


Figure 1.19 Persistence optimization results when minimizing the Wasserstein distance between the model and measurement persistence diagrams with artificial noise in the target persistence diagram.

CHAPTER 2

TIME SERIES REPRESENTATIONS

This chapter outlines my work studying different time series representations. Network representations are studied Section 2.1 where I include my work performing dynamic state identification methods using persistent homology of coarse-grained state space networks. Time delay embeddings for attractor reconstruction are also studied in Section 2.2 where novel methods for embedding delay estimation using persistent homology are shown.

2.1 Persistent Homology of Coarse Grained State Space Networks

This work is dedicated to the topological analysis of complex transitional networks for dynamic state detection. Transitional networks are formed from time series data and are typically studied using tools from graph theory to reveal information about the underlying dynamical system. However, traditional tools can fail to summarize the often complex topology present in such graphs. In this work we leverage persistent homology from Topological Data Analysis (TDA) to study the structure of these networks. Namely, we study the persistent homology of the Ordinal Partition Network (OPN) and Coarse Grained State Space Network (CGSSN) and analyze their dynamic state detection performance. The results show that the persistent homology of the CGSSN was able to accurately classify with 100% accuracy the dynamic state of 22 dynamical systems in comparison to the OPN at most having 95% accuracy. Additionally, we show that the CGSSN is more noise robust compared to OPNs with maximum allowable SNRs of 23 dB and 32 dB, respectively, for the Rossler system.

2.1.1 Introduction

Signal processing has been successfully and widely utilized to extract meaningful information from time series of dynamical systems including dynamic state detection [**?**, **?**, **?**, **?**, **?**, **?**], structural health monitoring for damage detection [**?**, **?**, **?**, **?**, **?**], and biological health monitoring [**?**, **?**, **?**, **?**, **?**, **?**, **?**]. A promising direction for signal processing is through studying the shape of signals. This is done by implementing tools from Topological Data Analysis (TDA) [**?**, 14] to study the shape of the attractor of the underlying dynamical system. This field of signal processing is known as Topological Signal Processing (TSP) [**?**], which has had many successful applications, including biological signal processing [**?**, **?**], dynamic state detection [**?**, 51], manufacturing [**?**, **?**, **?**, 2, 68], financial data analysis [**?**, **?**, **?**, **?**], video processing [**?**, **?**], bifurcation detection [**?**], and weather analysis [**?**, **?**].

The standard pipeline for TSP constructs a filtration of simplicial complexes (called the Vietoris-Rips complex) based on point cloud data generated from the State Space Reconstruction (SSR) of an input time series [**?**, **?**, **?**, 68]. Given a uniformly sampled signal $x = [x_1, x_2, \ldots, x_L]$, the SSR (also called the delay embedding) consists of n-dimensional delayed vectors

$$\mathbf{X} = \{v_i = [x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+\tau(n-1)}] \mid i \in \{1, \ldots, L-\tau(n-1)\}\}. \tag{2.1}$$

A simplicial complex is formed by including simplices for all collections of points which are within distance r of each other. We can measure the shape of the simplicial complex by forming simplicial complexes at increasing values of r, and tracking the changing homology through a linear mapping. This allows for quantifying when specific topologies form and disappear throughout the filtration

giving a sense of shape. The persistence diagram encodes this information for various dimensions, e.g., connected components (dimension zero), loops (dimension one), voids (dimension two). For example, one can examine the one dimensional homology to track loop structures in the SSR that are related to the periodicity of the signal. A problem with this pipeline is its computational demand having complexity $O(N^3)$, where $N = \binom{n}{d+1}$ is the size of the simplicial complex with n as the number of points in the simplical complex and d as the maximum dimension of the used homology. For long signals, this makes this standard pipeline computationally infeasible. A common solution is to subsample the point cloud, but it can be challenging to select an appropriate subsampling rate that preserves the topology of interest.

An alternative, promising direction for signal processing is analyzing time series via representations as complex networks [**?**, **?**, **?**]. Network representations of time series generally fall within three categories: proximity networks, visibility graphs, and transitional networks. Proximity networks are formed from closeness conditions in the reconstructed state space. Examples include the k-Nearest Neighbors (k-NN) [**?**] and recurrence networks [**?**], where the recurrence network is the network underlying the Vietoris-Rips complex of the point cloud data. For proximity networks, the graph representation includes all points in the state space reconstruction as part of the vertex set and does not reduce the computational complexity. Additionally, these networks require choosing a proximity parameter dependent on the signal, where careful consideration is needed in selecting the number of neighbors k or proximity distance ε to generate a graph that captures the correct topology. The visibility graphs [**?**] are formed by adding vertices for each data point and adding connecting edges if a visibility line can be drawn between the two vertices which do not pass below any other data point between the two. As our focus in this work is on building upon the strong theory developed for the SSR embedding, we will not utilize the visibility graph constructions at this stage. Instead, in this work we focus on transitional networks.

Transitional networks partition a time series x such that it has a vertex set of states $\{s_i\}$ for each visited state and an edge for temporal transitions between states. The resulting transitional network constitutes a finite state space $\mathcal{A}$ as the alphabet of possible states. One interpretation of a topological system on a finite state space is as a finite graph where the edges describe the action of a function ϕ, i.e., if there is a directed edge from vertex a to vertex b, then $\phi(a) = b$. Therefore, the transitional networks we obtain from a time series are topological systems, and they yield themselves to further analysis within the framework of topological dynamics. The two most common transitional networks for time series analysis are the Ordinal Partition Network (OPN) [**?**] and the Coarse Grained State Space Network (CGSSN) [**?**, **?**, **?**, **?**]. In Fig. 2.1 we demonstrate the rich topological structure of the CGSSN for periodic and chaotic dynamics from the Rossler system. This example shows the periodic dynamics corresponding to an approximate cycle graph while the network of the chaotic signal is highly intertwined.

To date, the majority of evaluation of these complex network representations is through standard graph theory tools [**?**, **?**, **?**, **?**], but the results can only provide local structural measurements based on the node degree distribution or shortest path measurements. In our previous work [51], we studied the global shape of these networks using persistent homology for dynamic state detection using the ordinal partition network. However, we only used the shortest unweighted path to define distances between nodes, which discarded edge weight and direction information. In our recent work [**?**] we investigate the use of weighted edge information based on the number of edge transitions. We found that this improved dynamic state detection performance.

However, we show here that there is an issue with the OPN, namely, amplitude information is

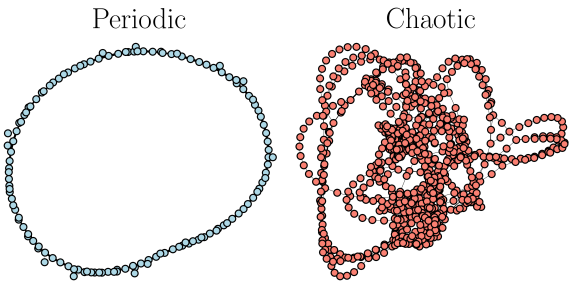


Figure 2.1 Example periodic and chaotic CGSSNs generated from the $x(t)$ solution to the Rössler system.

discarded because the ordinal partition network is built from permutations. Permutations can be thought of as partitioning the state space via intersections of hyperplanes of the form $x_i \leq x_j$. As such, the resulting OPN can have reduced dynamic state detection performance and extreme sensitivity to additive noise for some signals. This can be partially explained by noting that proximity of the trajectory to the hyperdiagonal can cause failures in network construction, particularly when there is noise in the signal (details of this issue are provided in Section 2.1.4.5). Further, due to the hyperdiagonal intersection issue, we cannot guarantee the stability of the persistence diagram for all signals. Therefore, we turn our attention to the CGSSN to bypass the limitations in OPN. We investigate the applicability of the CGSSN for enhanced noise robustness and dynamic state detection compared to the OPN. The results presented are based on analyzing the complex networks using persistent homology and tools from information theory and machine learning. Our results show an improvement in dynamic state detection performance with 100% separation between periodic from chaotic dynamics for noise-free signals using a nonlinear support vector machine compared to at most 95% for the OPN. Additionally, we show an improved noise robustness with the CGSSN functioning down to a signal-to-noise ratio of 22 dB compared to 29 dB for the OPN.

Organization

In Section 2.1.2 we overview the necessary background information. We begin with an introduction to the two transitional networks we study—OPN and CGSSN—and an overview of how they are related to state space reconstruction. Next, we introduce four standard methods for measuring the distance between nodes in a weighted graph. We subsequently describe persistent homology and how it is applied to study the shape of the weighted complex networks. In Section 2.1.3, we demonstrate how to apply our pipeline for studying the shape of complex transitional networks for a simple periodic example. In Section 2.1.4, we show results for studying the persistent homology of both the OPN and CGSSN. We begin with results for dynamic state detection for the Lorenz system with a periodic and chaotic response. We then apply the method to 23 continuous dynamical systems, and utilize machine learning to quantify the dynamic state detection performance over a broad range of signals. Lastly, we show results on the noise robustness of the CGSSN in comparison to the OPN. In Section 2.1.5, we provide conclusions future work on applying persistent homology to study the structure of transitional networks.

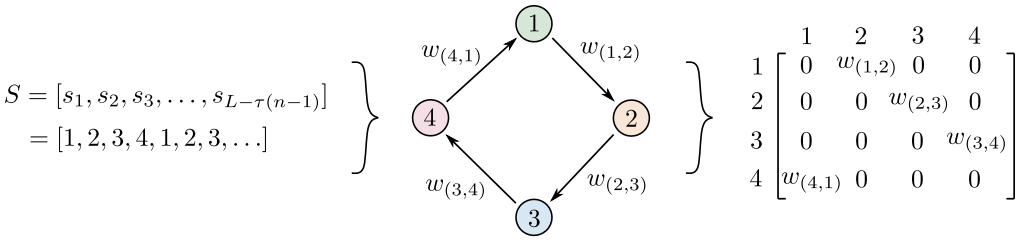


Figure 2.2 Example formation of a weighted transitional network as a graph (middle figure) and adjacency matrix (right figure) given a state sequence S (left figure).

2.1.2 Background

2.1.2.1 Transitional Complex Networks

A graph $G = (V,E)$ is a collection of vertices V and edges $E = (u,v) \subseteq V \times V$. We assume all graphs are simple (no self-loops or hypergraphs) and undirected. Additional stored information comes as a weighted graph, $G = (V,E,\omega)$ where $\omega : E \to \mathbb{R}_{\geq 0}$ gives a non-negative weight for each edge in the graph. Given an ordering of the vertices $V = \{v_1, \ldots, v_n\}$, a graph can be stored in an adjacency matrix $\mathbf{A}$ where the weighting information is stored by setting $\mathbf{A}_{ij} = w_{(v_i,v_j)}$ if $(v_i, v_j) \in E$ and 1 otherwise.

Transitional networks are graphs formed from a chronologically ordered sequence of symbols or states derived from the time series data. In our construction, these states are mapped from the measurement signal by first creating an SSR $\mathbf{X}$ from Eq. (2.1) and then assigning a symbolic representation for each vector $v_i \in \mathbf{X}$. To form a symbolic sequence from the time series data, we implement a function to map the SSR to symbol in the alphabet $\mathcal{A}$ of possible states as $f : v_i \to s_j$, where $s_j \in \mathcal{A}$ is a symbol from the alphabet. In this work, we consider the symbols from the alphabet as integers such that $s_i \in \mathbb{Z} \cap [1,N]$, where N is the number of possible symbols. Applying this mapping over all embedding vectors we get a symbol sequence as $S = [s_1, s_2, \ldots, s_{L-\tau(n-1)}]$. This work investigates two methods for mapping SSR vectors v_i to symbols s_j. The first is the OPN which is defined in Section 2.1.2.2 and is based on permutations. The second method is the CGSSN defined in section 2.1.2.3 which uses an equal-sized hypercube tessellation.

The symbol sequence S forms a transitional network by considering a graph $G = (V,E)$, where the vertices V are the collection of the used symbols, and the edges are added based on transitions between symbols in S. We represent the graph using the adjacency matrix $\mathbf{A}$ data structure of size $N \times N$. We add edges to the adjacency matrix $\mathbf{A}$ via the symbolic transitions with an edge between row s_i and column s_{i+1} for each i. This is represented in the adjacency matrix structure by incrementing the value of $\mathbf{A}_{s_i,s_j}$ by one for each transition between s_i and s_{s+1}, where $\mathbf{A}$ begins as a zero matrix. We set the total number of transitions between two nodes as the edge weight $w_{(s_i,s_j)}$. We ignore self-loops by setting the diagonal of $\mathbf{A}$ to zero. To better illustrate the transitional network formation process, consider the simple cycle shown in Fig. 2.2. In this example, we take the state sequence S on the left side of Fig. 2.2 with symbols in the alphabet $\mathcal{A} = [1,2,3,4]$ and create the network shown network in the middle of the figure. This network is represented as a directed and weighted adjacency matrix, as shown on the right side of Fig. 2.2. In this paper, we discard the directionality information and make $\mathbf{A}$ symmetric by adding its transpose, $\mathbf{A} + \mathbf{A}^T$.

2.1.2.2 Ordinal Partition Network

To form an OPN, the SSR $\mathbf{X}$ must first be constructed requiring the choice of two parameters: the delay τ and dimension n. We select the delay τ using the method of multi-scale permutation entropy [**?**, 69] and the dimension as $n = 7$ as suggested for permutation entropy [69, 69]. For the OPN, the vector v_i is assigned to a permutation π based on its ordinal partition. For dimension n there are $n!$ permutations (e.g., 6 possible permutations for dimension $n = 3$ shown in Fig. 2.3) which can order arbitrarily $\pi_1, \cdots \pi_{n!}$. Then v_i is assigned to a permutation π_k following that π_k satisfies $v_i(\pi_k(0)) \leq v_i(\pi_k(1)) \leq \cdots \leq v_i(\pi_k(n-1))$. An example of this for the vector $v_i = [-0.08, 0.48, -0.34]$ is shown on the top Ordinal Partition (OP) route of Fig. 2.3 where v_i is mapped to permutation π_5 and state $s_i = 5$.

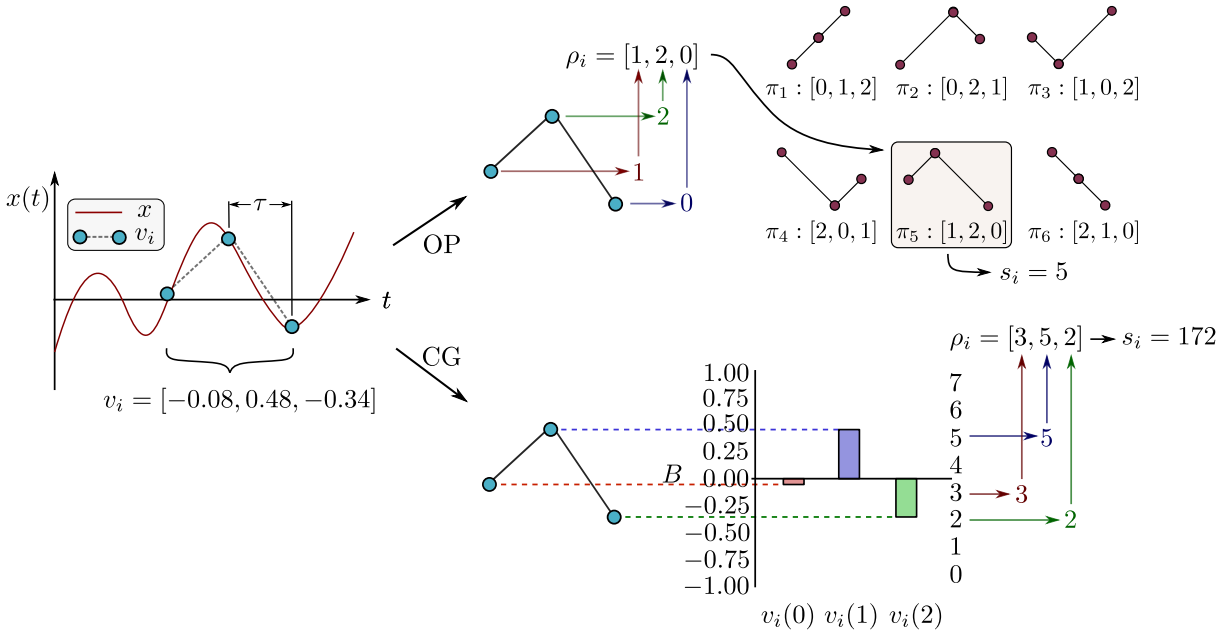


Figure 2.3 Example state assignment using the Ordinal Partition (OP) method (top) and Coarse Graining (CG) method (bottom). The state for the OP method is based on the assigned permutation number with $s_i = 5$ for the example. The state assignment for the CG method is based on the number of bins where $s_i = 1 + \sum_{j=0}^{n-1} \rho_i(j) b^j$, ρ_i is the digitization of vector v_i based on binning into b equal-sized bins spanning $[\min(x), \max(x)]$. For this example, $s_i = 3(8^0) + 5(8^1) + 2(8^2) + 1 = 172$ with $b = 8$ bins.

2.1.2.3 Coarse Grained State Space Network

The CGSSN begins by constructing the SSR, where we select the delay τ using the multi-scale permutation entropy method [**?**, 69] and dimension n using the false nearest neighbors [**?**] based on only needing a dimension great enough for periodic orbits to not self-intersect. For the CGSSN, the vector $v_i \in \mathbf{X}$ is assigned to a state based on which partitioned region the vector v_i lies within. We define the domain $\mathbb{D}$ of the SSR as the non-empty connected, open set that encloses all vectors of the SSR. Specifically, we use an n-dimensional hypercube domain bounded by the intervals $[\min(x), \max(x)]$ for each dimension. In this work we cover this domain using a tessellation of $N = b^n$ hypercubes with side length $(\max(x) - \min(x))/b$, where b is the number of

bins per dimension. We assign each n-dimensional hypercube in the tessellation a unique symbol by converting it to a decimal representation denoted as s_i. An introductory example formation of the entire CGSSN for a sinusoidal function is provided in Section 2.1.3. Some generalizations exist to the described method where instead of assigning symbols to the individual hypercubes, we could assign words of length m which would allow for studying a sequence of coarse grained states of the system which reduces the information load in the process [**?**]. For the purpose of this paper, a symbolic representation was sufficient.

2.1.2.4 Vertex similarity and dissimilarity measures

To study the structure of the complex network we define functions of the form $V \times V \to \mathbb{R}_{\geq 0}$ combining information about path lengths and weights from the graph in various ways. Some of these definitions are distances, but not all. Despite this, the framework can still be used to define a filtered simplicial complex in the spirit of the Vietoris Rips complex which will be required in the next section.

The measures are encoded in a matrix $\mathbf{D}$, where $\mathbf{D}(a,b)$ is the similarity or dissimilarity between vertices a and b. Note that $\mathbf{D}$ can optionally be normalized by dividing all entries by its maximum value to contain values between 0 and 1. We investigated the use of four choices of measures: the unweighted shortest path distance, the shortest weighted path dissimilarity, the weighted shortest path distance, and the diffusion distance.

Shortest Path Distances and Dissimilarities Commonly used in graph theory, the *shortest path distance* is based on minimizing the cost of taking a path from node a to b. This assumes a path $P = [n_0, n_1, \ldots, n_s]$ consisting of s nodes where $a = n_0$ and $b = n_s$ exists, but we note that all graphs in this paper are connected by construction. The path P can alternatively be represented as the sequence of connected edges between a and b: $P = [e_{0,1}, e_{1,2}, \ldots, e_{s-1,s}]$. The shortest path is determined based on minimizing the path cost function

$$C(P) = \sum_{e \in P} w(e). \tag{2.2}$$

In the case of an weighted graph, we then define $D(a,b) = \min_P C(P)$. Note that in the case of an unweighted graph, we have all weights equal to 1 and thus the cost of a path is simply the number of edges included in it.

We next define two variations on this idea, although they are not quite distances but are useful for the kinds of input graph data we study. In particular, the weights on edges are higher for those that are more highly traversed with the transitional networks. We thus want these paths to be considered more important than those only traversed a few times. To that end, we will focus on paths whose length using the reciprocal of the weights is as small as possible.

The first variation, called the *weighted shortest path* measure, is defined as follows. First, we find the path from a to b with the minimum total path weight in terms of the reciprocal weights. That is, P such that

$$C'(P) = \sum_{e \in P} 1/w(e). \tag{2.3}$$

is minimized. We then define $D(a,b) = \sum_{e \in P} w(e)$. For this definition, D encodes information about frequency of traversal of the edges.

The second variation, called the *shortest weighted path*, still uses the path P for which $C'(P)$ is minimized. However, in this case, we define $D(a,b)$ to be the length of the path; i.e. the number of edges in P. For this variant, we are essentially giving higher priority to well traveled paths, but using a measurement of this path related to the number of regions of state space are traversed.

Diffusion Distance The final vertex similarity measure we use is the diffusion distance for graphs [**?**]. The diffusion distance leverages the transition probability distribution matrix $\mathbf{P}$ of the graph, where $\mathbf{P}(a,b)$ is the probability of transitioning to b when at a in a single step based on the random walk framework. Specifically, given the weighted, undirected adjacency matrix $\mathbf{A}$ with no self-loops (i.e., zero diagonal), the transitional probability matrix is

$$\mathbf{P}(i,j) = \frac{\mathbf{A}(i,j)}{\sum_{k=1}^{|V|} \mathbf{A}(i,k)}. \tag{2.4}$$

Equation (2.4) can be extended to calculate the transition probabilities for non-adjacent neighbors by raising them to higher powers. For example, transitioning to vertex b from vertex a in t random walk steps is $\mathbf{P}^t(a,b)$. A common modification of Eq. (2.4) is to include a probability that a random walk can stay at the current vertex, which is commonly referred to as the lazy transition probability matrix. This is given by

$$\widetilde{\mathbf{P}} = \frac{1}{2}\left[\mathbf{P}(a,b) + \mathbf{I}\right], \tag{2.5}$$

where $\mathbf{I}$ is the identity matrix matching the size of $\mathbf{P}$. The diffusion distance measures how similar two nodes are based on comparing their t-step random walk probability distributions. This is done by taking the degree-normalized ℓ_2 norm of the probability distributions between nodes and is calculated as

$$d_t(a,b) = \sqrt{\sum_{c \in V} \frac{1}{\mathbf{d}(c)} \left[\widetilde{\mathbf{P}}^t(a,c) - \widetilde{\mathbf{P}}^t(b,c)\right]^2} \tag{2.6}$$

where $\mathbf{d}$ is the degree vector of the graph with $\mathbf{d}(i)$ as the degree of node i. Applying the diffusion distance to all node pairs results in the distance matrix $\mathbf{D}_t$.

2.1.2.5 Persistent Homology of Complex Networks

A simplicial complex is a generalization of a graph to higher dimensions, which are collections of simplices at various dimensions (e.g., points are zero-dimensional, edges are one-dimensional, and faces are two-dimensional simplices). These simplices are subsets of a vertex set $\sigma \subset V$, and we require for the complex that if $\sigma \in K$ and $\tau \subseteq \sigma$, then τ is also in K. Using a distance matrix to describe similarity between nodes, or indeed any function of the form $d : V \times V \to \mathbb{R}$ where $d(v,v) = 0$ although we still call this a distance matrix for simplicity, we can construct simplicial complex representations from graphs at a distance level r. This idea is related to the Vietoris Rips complex, where we build a simplicial complex K_r for any fixed parameter $r \geq 0$ by including all simplices with pairwise relationships at most r; i.e. $K_r = \{\sigma \subseteq V \mid d(u,v) \leq r \text{ for all } u,v \in \sigma\}$. Zero-dimensional simplices, the vertices of the complex, are all added at $r = 0$. An edge uv, which is a 1-dimensional simplex, is present in K_r for any r value above $d(u,v)$. Higher dimensional simplices such as triangles are included when all subedges are present; equivalently this means a simplex is added for every clique in the complex. For example, consider Fig. 2.4 which shows a

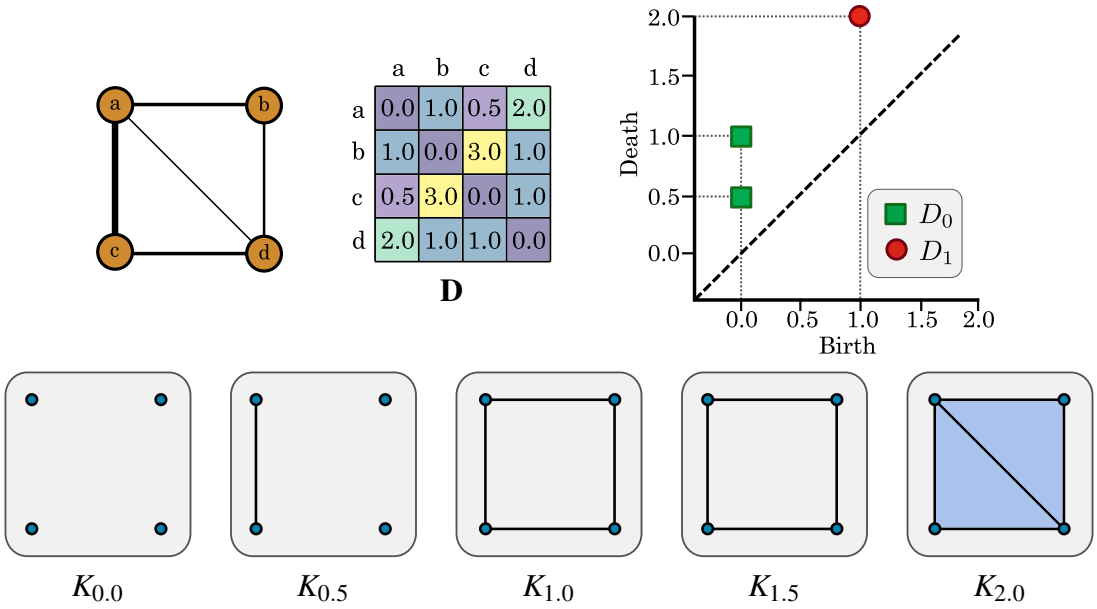


Figure 2.4 Example demonstrating persistent homology of a graph using the matrix **D** with resulting persistence diagram shown top right. The filtration of simplicial complexes are shown in the bottom row.

graph with four nodes, and the associated distance matrix **D**. For each $r \in [0.0, 0.5, 1.0, 1.5, 2.0]$ the associated simplicial complex is shown as K_r in the bottom row.

We can use homology [**?**, 7] to measure the shape of any such simplicial complex K which is denoted $H_d(K)$. This mathematical object is a vector space, where elements are representative of d-dimensional features (i.e., connected components (zero-dimensional structure), loops (one-dimensional structure), voids (two-dimensional structure), and higher dimensional analogues) in K. In this work we will only utilize the 0-dimensional and 1-dimensional features to measure the connected components and holes in the simplicial complex. For example, consider the simplicial complex K_r at $r = 1.0$ in Fig. 2.4, which has one H_0 classes with a single connected component and one H_1 class with a single loop or hole in the simplicial complex.

An issue with just using homology to measure the shape of a simplicial complex to understand the shape of a graph is that the correct distance value r needs to be selected. Additionally, it does not provide any information on the geometry or size of the underlying graph. To alleviate these issue we use persistent homology [**?**], which studies the *changing* homology of a sequence of simplicial complexes. We will again use Fig. 2.4 as an example for demonstrating how the persistent homology is calculated. To calculate the persistent homology we begin with a collection of nested simplicial complexes

$$K_{r_1} \subseteq K_{r_2} \subseteq \cdots \subseteq K_{r_N}.$$

The bottom row of Fig. 2.4 shows an example of this filtration over the distance parameter r with $K_{r=1.0} \subseteq K_{r=0.5} \subseteq \cdots \subseteq K_{r=2.0}$. We then calculate the homology of each simplicial complex and create linear maps between each homology class for each dimension d as

$$H_d(K_{r_1}) \to H_d(K_{r_1}) \to \cdots \to H_d(K_{r_N}).$$

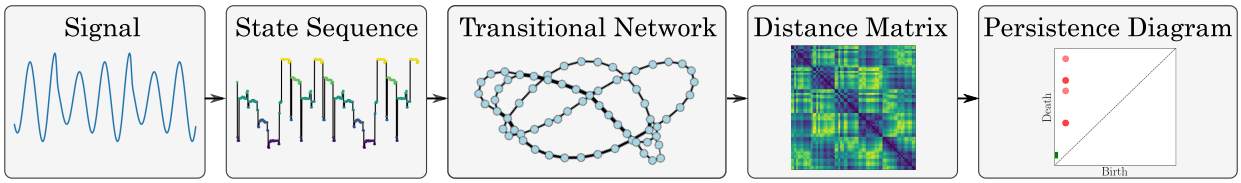


Figure 2.5 Pipeline for studying transitional networks using persistent homology. From left to right, we begin with a signal or time series and represent it as a state sequence which is summarized using a transitional network as described in Section 2.1.2.1. A distance between nodes is then used to create a distance matrix (see Section 2.1.2.4 for graph distances) which can be directly analyzed using persistent homology shown in Section 2.2.2.4.

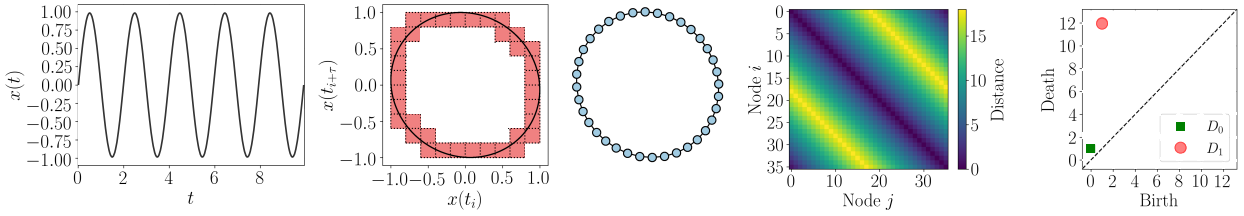
By studying the formation and disappearance of homology classes we can understand the shape of the underlying graph. Specifically, class $[\alpha] \in H_d(K_{r_i})$ is said to be born at r_i if it is not in the image of the map $H_d(K_{r_{i-1}}) \to H_d(K_{r_i})$. The same class dies at r_j if $[\alpha] \neq 0$ in $H_d(K_{r_{j-1}})$ but $[\alpha] = 0$ in $H_d(K_{r_j})$. In the case of 0-dimensional persistence, this feature is encoding the appearance of a new connected component at K_{r_i} that was not there previously, and which merges with an older component entering K_{r_j}. For 1-dimensional homology, this is the formation (birth) and disappearance (death) of a loop structure. We store this information in what is known as the persistence diagram using the persistence pair $x_i = (b_i, d_i) \in D_d$, where D_d is the persistence diagram of dimension d with a homology class of dimension d being born at filtration value b_i and dying at d_i. We also define the lifetime or persistence of a persistence pair as $\ell_i = \text{pers}(x_i) = d_i - b_i$. The set of lifetimes for dimension d is defined as L_d. For a more detailed roadmap for the calculation of persistent homology we direct the reader to the work of Otter et al [**?**].

Returning to our example, the persistence diagram is shown in Fig. 2.4 for both D_0 and D_1. For D_0 all four persistence pairs were born at $r = 0.0$ with one dying at $r = 0.5$ and two dying at $r = 1.0$. The fourth persistence pair in D_0, not drawn, is an infinite-class dying at ∞ since there is a single component for $r \geq 1.0$. In this work we do not utilize infinite-class persistence pairs and will not include them in the persistence diagrams. For D_1 there is a single persistence pair born at $r = 1.0$ with the formation of the loop in K_1 and filling in at K_2.

2.1.3 Method

This section describes the method for studying complex transitional networks using persistent homology. The pipeline for doing this is outlined in Fig. 2.5. We begin with a signal or time series and represent it as a state sequence described in Section 2.1.2.1. The state sequence can be summarized using a weighted transitional network as described in Sec. 2.1.2.1. A distance between nodes (see Section 2.1.2.4) is then used to create a distance matrix which can be directly analyzed using persistent homology as described in Section 2.2.2.4.

To further describe the method we develop here, we use a simple periodic signal example shown in Fig. 2.6. The signal is defined as $x(t) = \sin(\pi t)$ sampled at a uniform rate of $f_s = 50$ Hz. The SSR was constructed using $n = 2$ and $\tau = 26$. For this example, we create the CGSSN by partitioning the SSR domain into 100 rectangular regions as states, each with a unique symbol. The states visited through the SSR trajectory are highlighted in red. The temporal tracking of the states used creates the state sequence, which is then represented as the cycle graph. This example demonstrates how the periodic nature of the signal is captured by the cycle structure of



(a) (b)

Figure 2.6 Example demonstrating CGSSN formation procedure ($b = 10$) with the signal $x(t) = \sin(t)$ embedded into $\mathbb{R}^2$ space using an SSR and analysis using persistent homology with the unweighted shortest path distance. (a) Formation of the CGSSN from a time series signal and its delayed signal, (b) The distance matrix and associated persistence diagram using the unweighted shortest path distance.

the corresponding CGSSN.

We define a distance between nodes using the unweighted shortest path distance for this example due to its simplicity. The corresponding distance matrix and resulting persistence diagram are shown. The resulting persistence diagram shows that the periodic structure of the underlying time series and corresponding CGSSN is captured by the single point in the persistence diagram D_1 at coordinate $(1,12)$ with the loop structure being born at a filtration distance of 1 and filling in 12.

2.1.4 Results

This section shows that the CGSSN outperforms the previously used OPN for both noise robustness and dynamic state detection performance. We first begin in Section 2.1.4.1 where we provide a simple example highlighting improved dynamic state detection performance of the CGSSN over the OPN for a periodic and chaotic Rossler system simulation. We show these results using the persistent entropy summary statistic. The second result in Section 2.1.4.2 quantifies the dynamic state detection, of the OPN and CGSSN using lower dimensional embedding on 23 continuous dynamical systems with periodic and chaotic simulations. Lastly, in Section 2.1.4.5, we empirically investigate the noise robustness of the CGSSN compared to the OPN.

2.1.4.1 Dynamic State Detection for Rossler System

Our first result is from a study of the complex network topology of OPNs compared to CGSSNs. To demonstrate the difference and motivate why the CGSSN outperforms the OPN in terms of dynamic state detection, we use an $x(t)$ simulation of the Rossler system defined as

$$\frac{dx}{dt} = -y - z, \qquad \frac{dy}{dt} = x + ay, \qquad \frac{dz}{dt} = b + z(x - c). \tag{2.7}$$

We simulated Eq. (2.26) using the *scipy* odeint solver for $t \in [0, 1000]$ with only the last 230 seconds used to avoid transients. The signal was sampled at a rate of $f_s = 22$ Hz. For periodic dynamics we use system parameters of $[a,b,c] = [0.1, 0.2, 14]$ and for chaotic we set $a = 0.15$. These simulated signals are shown in Fig. 2.7. To create the OPNs for both signals, we used an embedding delay $\tau = 43$ selected using the multi-scale permutation entropy method and dimension $n = 7$. The corresponding networks are shown in the second column of Fig. 2.7. To form the CGSSNs we similarly chose $\tau = 43$, but used dimension $n = 4$ and $b = 12$ for partitioning the SSR with resulting networks shown in the third column.

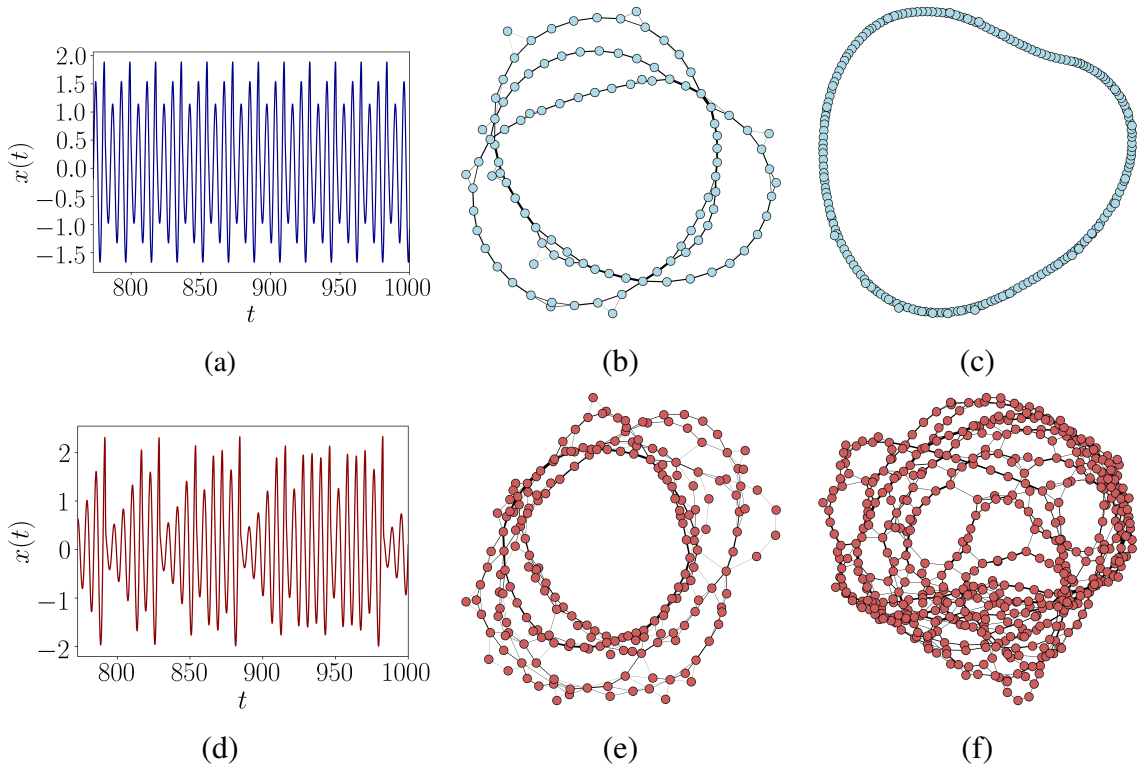


Figure 2.7 Transitional complex network topology comparison between OPN and CGSSN for the $x(t)$ simulation of the Rossler system described in Eq. (2.26). (a) Periodic Rossler Simulation $x(t)$, (b) Periodic OPN ($n = 7$) $E'(D_1) = 0.503$., (c) Periodic CGSSN ($n = 4$ and $b = 12$). $E'(D_1) = 0.026$, (d) Chaotic Rossler Simulation $x(t)$, (e) Chaotic OPN ($n = 7$). $E'(D_1) = 0.893$, (f) Chaotic CGSSN ($n = 4$ and $b = 12$). $E'(D_1) = 0.905$.

The resulting OPN and CGSSN from the Rossler system simulations of periodic and chaotic dynamics both capture the increasing complexity of the signal with the dynamic state change. For the periodic signal, the OPN show overarching large loops relating to the periodic nature of the SSR. However, the CGSSN better captures the periodic nature of the trajectory with only a single loop forming. This characteristic of the CGSSN is due to periodic flows never intersecting in the SSR if the signal is sampled at a high enough frequency, there is no or little additive noise, and an appropriately sized delay and dimension are selected. While correctly choosing the delay and dimension is not a trivial task, there is a broad literature on their selection for the SSR task. This work relies on the multi-scale permutation entropy method for selecting the delay and the false-nearest-neighbors algorithm [**?**] for selecting an appropriate SSR dimension. However, we found that increasing the dimension one higher than that suggested using false-nearest-neighbors more reliably formed a single loop structure in the CGSSN. Additionally, in Appendix 2.1.6 we demonstrate that for 23 dynamical systems, setting $b \geq 12$ resulted in only a single loop structure for periodic signals while minimizing the computational demand when using the CGSSN. As such, we set $b = 12$ unless otherwise stated.

For the chaotic $x(t)$, the OPN and CGSSN both summarize the topology of the attractor with both networks having a high degree of entanglement with nodes being highly intertwined. This is a typical characteristic of complex transitional networks formed from chaotic signals. Furthermore,

it should be noted that the CGSSN tends to be more entangled than its OPN counterpart, suggesting that the CGSSN better captures the increase in complexity of the chaotic signal.

To quantify how well the OPN and CGSSN capture the complexity of the signals, we rely on persistent entropy [**?**], which was previously adapted [51] to study the resulting persistence diagram using the unweighted shortest path distance of complex networks. The normalized persistent entropy [**?**, **?**] is defined as

$$E'(D) = \frac{-\sum_{x \in D} \frac{\text{pers}(x)}{\mathscr{L}(D)} \log_2 \left(\frac{(\text{x})}{\mathscr{L}(D)} \right)}{\log_2 \left(\mathscr{L}(D) \right)}, \tag{2.8}$$

where $\mathscr{L}(D) = \sum_{x \in D} \text{pers}(x)$ with $\text{pers}(x) = |b - d|$ as the lifetime or persistence of point $x = (b, d)$ in a persistence diagram D. For studying the complexity of transitional network we apply this score to the one-dimensional persistent diagram D_1, which measures the loop structures in the network. This score yields a value close to zero for networks with a single loop structure corresponding to periodic dynamics and a value close to one for chaotic dynamics with highly intertwined networks. For our example OPN and CGSSNs in Fig. 2.7 we get normalized persistent entropy scores of 0.503 and 0.893 for periodic and chaotic OPNs, respectively, and 0.026 and 0.905 for CGSSNs. These statistics show that the CGSSN outperforms the OPN with a significantly larger difference in the entropy values. This is mainly due to the CGSSN having a score near zero for periodic dynamics due to its general loop structure compared to the periodic OPN having several loops. This result comparing the OPN and CGSSN suggests that the CGSSN will outperform the OPN for the dynamic state detection task. With this single case under our belt, we turn our attention to an empirical study of this characteristic over more dynamical systems.

2.1.4.2 Empirical Testing of Dynamic State Detection for 23 Continuous Dynamical Systems

The previous example in Section 2.1.4.1 showed the improved dynamic state detection performance of the CGSSN over the OPN for a single example (Rössler System). However, we want to show that this improvement is present over various systems. To do this, we use 23 continuous dynamical systems listed in the Appendix 2.1.7 with details on the simulation method—each system was simulated for both periodic and chaotic dynamics.

For each periodic and chaotic signal, we calculate the resulting persistence diagram of the OPN and CGSSN using each of the distance methods (unweighted shortest path, shortest weighted path, weighted shortest path, and diffusion distance). We then compare the collection of persistence diagrams for a specific network type (OPN or CGSSN) and distance measure by calculating the bottleneck distance matrix between each persistence diagram. The bottleneck distance $d_{BN}(D, F)$ is a similarity measure between two persistence diagrams (D and F). It is calculated as the sup norm distance between the persistence diagrams, where the persistence diagrams are optimally matched with the distance between matched persistence points being at most d_{BN}. The bottleneck distance matrix $\mathbf{D}_{BN}$ is calculated by finding d_{BN} between all persistence diagrams.

The question we are trying to answer is if periodic and chaotic dynamics result in similar persistence diagrams across multiple systems. To answer this, we first use a lower-dimensional projection of $\mathbf{D}_{BN}$ by implementing the Multi-Dimensional Scaling (MDS) projection to two dimensions. To measure how well the dynamics delineate on the MDS projection, we use a Support Vector Machine (SVM) with a Radial Basis Function (RBF). Note that because the MDS does not allow for the mapping of previously unseen points, we cannot use this procedure for a proper

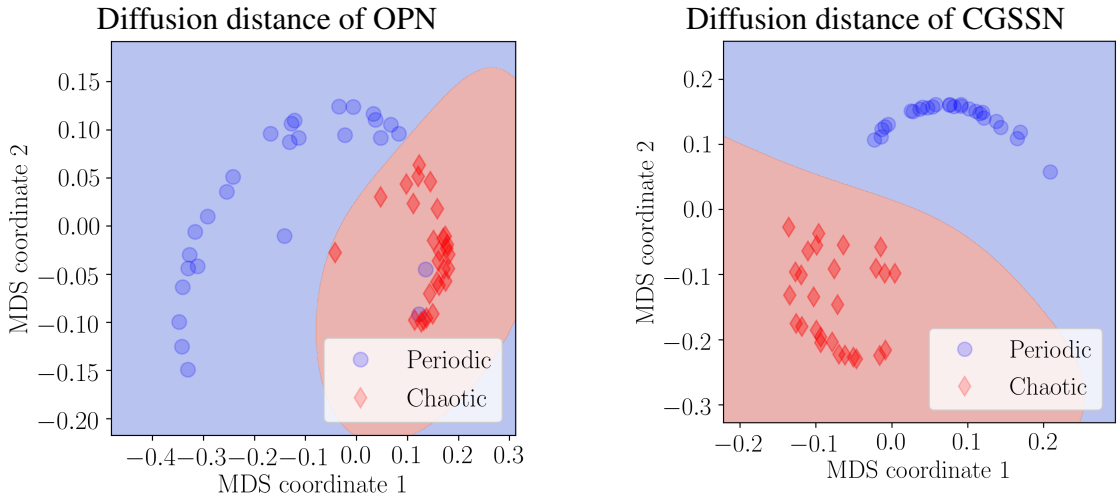


Figure 2.8 Two-dimensional MDS projection of the bottleneck distances between persistence diagrams of the chaotic and periodic dynamics with an SVM radial bias function kernel separation. This separation analysis was repeated for the OPNs and CGSSNs using the diffusion distance.

classification test as we cannot approximate training error. However, we can use this procedure to see if the the persistence diagrams of different classes are separated with respect to the bottleneck distance.

We fit the SVM using the default `SKLearn` SVM parameters package. The resulting separations for periodic and chaotic dynamics using the OPN (left) and CGSSN (right) are shown in Appendix 2.1.8. These separations are for the diffusion distance calculation as it provided the best results for both the OPN and CGSSN. However, we also include similar figures for other choices of distances in Appendix 2.1.8.

Figure 2.8 demonstrates the significant improvement in dynamic state detection of the CGSSN over the OPN. This is shown with the periodic and chaotic networks being clustered for the CGSSN (right of Fig. 2.8) with no overlap compared to the OPN (left of Fig. 2.8) having some overlap between periodic and chaotic dynamics. This is further shown with the SVM kernel being able to separate the periodic and chaotic regions for the CGSSN easily. To better compare all distance measures and complex network combinations, we quantify the performance of each SVM kernel using the accuracy of the separation. We repeated this accuracy calculation 100 times for each combination using 100 random seeds to generate the SVM kernels. The resulting average accuracies with standard deviation uncertainties are reported in Table 2.1.

Based on the results in Table 2.1, the CGSSN outperforms the OPN for all distance measures. Additionally, we found 100% separation accuracy for both the shortest weighted path and diffusion distances when combined with the CGSSN. We believe this performance improvement is due to the coarse-graining procedure capturing the SSR vector's amplitude information which is discarded when identifying permutations in the OPN.

2.1.4.3 *n*-Periodic Systems

Based on the state space embedding structure of a system, one may expect that for a 2 or 3-periodic system that the CGSSN may result in 2 and 3 loops respectively, but this is not the

Table 2.1 Accuracies for SVM seperation of MDS projections for dynamic state detection. Uncertainties are recorded as one standard deviation for random seeds 1–100.

Network	Distance	Average Separation Accuracy	Uncertainty
OPN	Shortest Unweighted Path Distance	80.7%	1.5%
OPN	Shortest Weighted Path Distance	88.9%	0.0%
OPN	Weighted Shortest Path Distance	88.9%	0.0%
OPN	Lazy Diffusion Distance	95.0%	0.9%
CGSSN	Shortest Unweighted Path Distance	98.1%	0.9%
CGSSN	Shortest Weighted Path Distance	100.0%	0.0%
CGSSN	Weighted Shortest Path Distance	98.1%	0.9%
CGSSN	Lazy Diffusion Distance	100.0%	0.0%

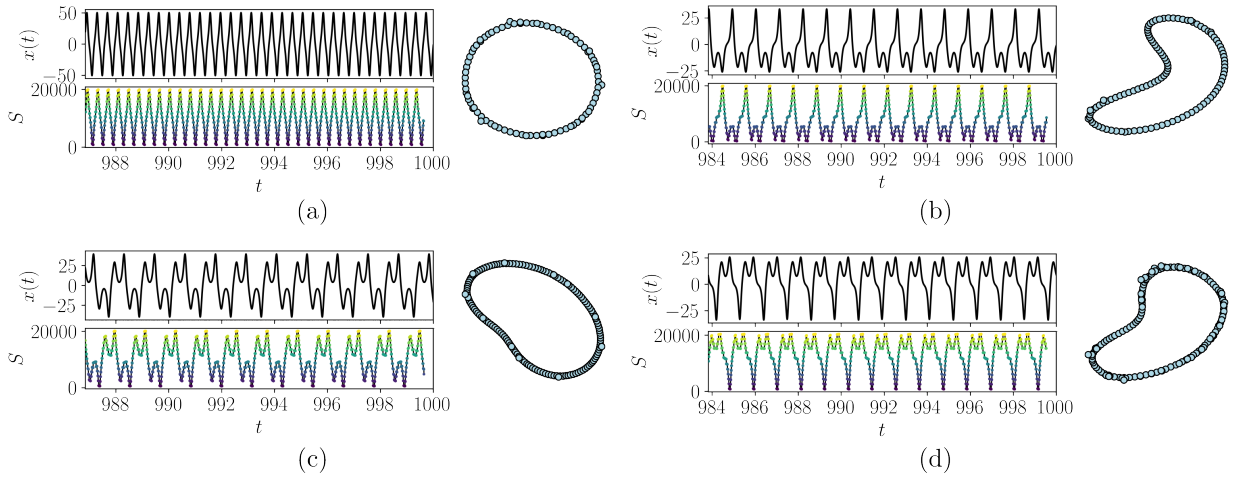


Figure 2.9 CGSSN results for four multi-periodic cases of the Lorenz system. The sequence of A's and B's below each image indicates the oribital sequence around the attractors A and B in the system. (a) AB $\rho = 350$, (b) AAB $\rho = 100.5$, (c) AABB $\rho = 160$, (d) ABBABB $\rho = 99.65$. As expected, all four cases result in a single loop CGSSN. These networks were generated using $n = 4$ and $b = 12$.

case. In general, for an n-periodic system, we expect the CGSSN to contain only a single loop, and so we caution the user that this method will likely not be able to differentiate differences in the periodicity. We demonstrate this nuance by showing CGSSN results on the Lorenz system for multi-periodic responses. Fig. 2.9 shows the corresponding CGSSNs for the Lorenz system varying the ρ parameter to obtain multi-periodic responses. The networks are labeled with a sequence of *A*'s and *B*'s where each letter corresponds to a loop in the trajectory around one of the attractors. For example *AAB* trajectory would be two loops around A and one around B before repeating the cycle. For all four cases shown, a single loop is obtained in the CGSSN even though the system exhibits multi-periodicity.

2.1.4.4 A Remark on Discrete Maps

As we demonstrated in Section. 2.1.4.2, the CGSSN method allows for efficient and accurate dynamic state detection over a range of continuous dynamical systems. Discrete maps are another

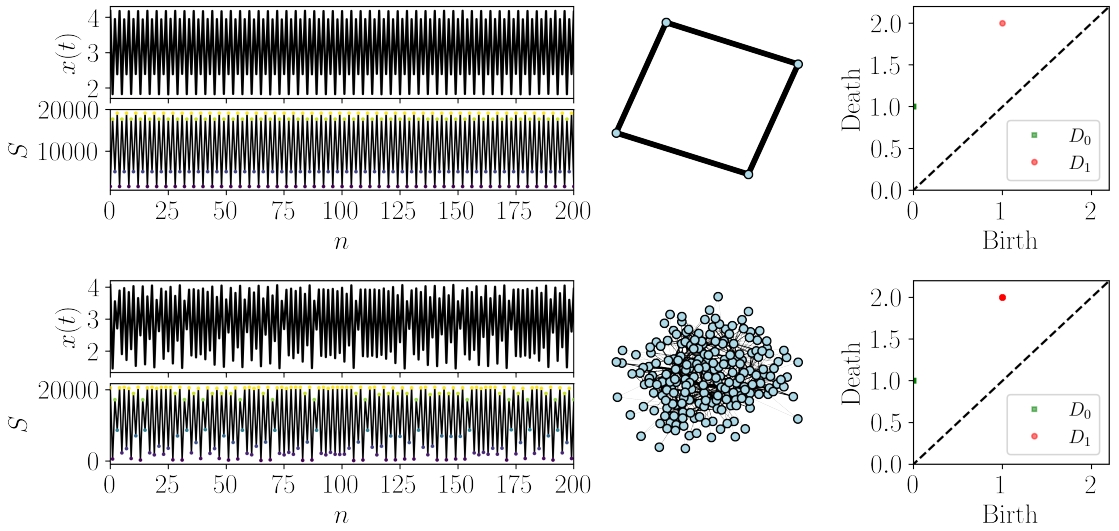


Figure 2.10 CGSSN results for the periodic (top row) and chaotic (bottom row) logistic map using the unweighted shortest path. The system responses are shown on the left along with the permutation sequence. The network representations are in the middle with the persistence diagrams on the right. Both networks exhibit the same persistence diagram due to the limited possible system states for the periodic case.

subset of dynamical systems where it would be useful to apply these tools; however, care must be taken for this type of system to ensure that the CGSSN is a suitable approach. This is because in discrete systems, there are typically far fewer states that the system can exhibit so in some cases the CGSSN may not contain any loops, but the response is still periodic leading to an incorrect classification in the model. To demonstrate, we show the CGSSNs for the periodic and chaotic logistic map in Fig. 2.10 where the unweighted shortest path distance was used to compute persistence. We see that the CGSSNs show vastly different structures where the periodic network contains a single loop and the chaotic network is tangled. However, the persistence diagrams for these networks appear to be equivalent because the networks were unweighted and all of the loops in the chaotic network are exactly the same size as the periodic case. Due to only having 4 possible states in the periodic logistic map here, the network loop does not provide enough of a difference to automatically classify it as either dynamic state. We note that the chaotic persistence diagram contains more loops than the periodic case here, but all are the same persistence lifetime. In the case of a continuous system where many more states are possible, these loops will be larger in size and the persistence diagram will reflect those differences allowing for classification of the dynamic state. In this case, when other distances are used such as the shortest weighted path, the resulting persistence diagrams have the forms that we expect for periodic and chaotic behaviors due to the weighting of the edges influencing the persistence lifetime of that loop.

In the case where the system being studied can exhibit many possible states in its periodic response, a single loop will form the CGSSN and the persistence diagram will show a persistence pair with a long lifetime. For example, we demonstrate this behavior on the 3 periodic linear congruential generator map in Fig. 2.11. The results in Figs. 2.10 and 2.11 demonstrate that this method should be used with caution on discrete systems and for systems with enough states that

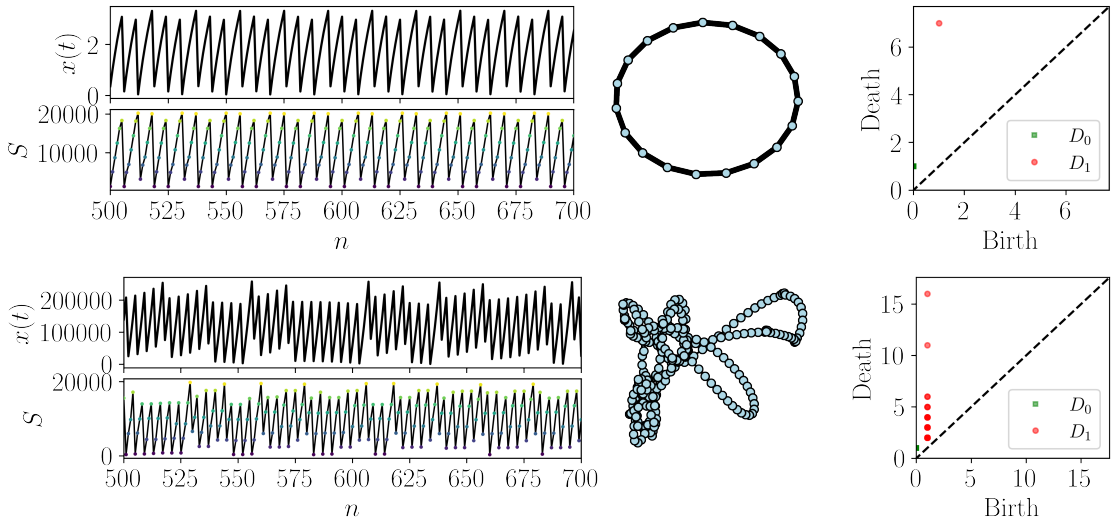


Figure 2.11 CGSSN results for the periodic (top row) and chaotic (bottom row) linear congruential generator map using the unweighted shortest path. The system responses are shown on the left along with the permutation sequence. The network representations are in the middle with the persistence diagrams on the right. Both networks exhibit the distinct persistence diagram structures due to the larger loop in the periodic network.

approach the behavior of a continuous system, the CGSSN persistence diagrams can provide a correct dynamic state detection.

2.1.4.5 Noise Sensitivity

One issue with ordinal partition networks is they are not exceptionally resilient to noise. Indeed, one can think of the ordinal partition network as being the 1-skeleton of the nerve of a particular closed cover of the state space, delineated by the hyperplanes $x_i \leq x_j$. Consequently, when noise is injected into the system, there are superfluous transitions when nearing one of these boundaries. For example, consider the signal and its embedding into $\mathbb{R}^3$ in Fig. 2.12.

This effect becomes even more prominent near an intersection of multiple hyperplanes. As the distance to the hyperdiagonal d^H becomes small, we see a significant increase in seemingly superfluous transitions between permutations π (highlighted in orange in Fig. 2.12). This issue is even more exaggerated when the embedded signal is consistently close to the hyperdiagonal, which results in network representations whose shape carries no information on the underlying dynamical system (e.g., see the signal and far-right OPN in Fig. 2.13). This is particularly detrimental when we attempt to include the weighting information, as the flips can skew the count for the number of times a boundary is crossed.

Certain network representations of time series are naturally more noise-robust than others. For example, Fig. 2.13 shows the OPN and CGSSN for the signal with and without noise. This example demonstrates that the CGSSN is the best choice for this signal with only minor changes in its shape, while the OPN loses all resemblance to the noise-free network.

Outside of this sensitivity to the hyperdiagonal, we also found that the CGSSN is more noise robust than the OPN for other signals. For example, in Fig. 2.14 we show the normalized persistent

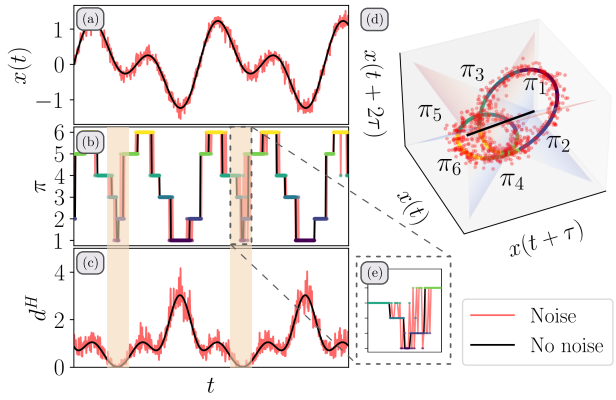


Figure 2.12 The three-dimensional state space reconstruction (d) from the signal $x(t)$ with and without additive noise (a) shows as the distance to the hyperdiagonal d^H (c) becomes small, undesired permutation transitions (b)—with zoomed-in section shown in (e)—occur as shown in the orange highlighted regions.

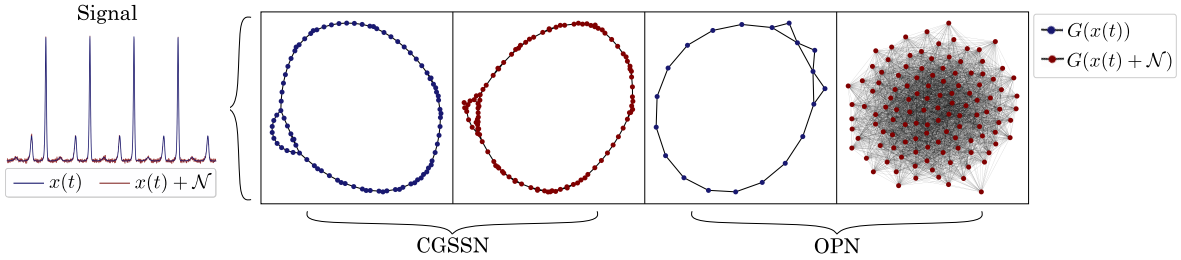


Figure 2.13 Example demonstrating the importance of choosing an appropriate network formation method when there is additive noise in the signal. The CGSSN retains the graph structure even with additive noise; in contrast, the OPN network loses all resemblance to the noise-free topological structure even with a small amount of additive noise. $x(t)$ is the signal, $\mathcal{N}$ is additive noise, and $G(x)$ is the graph representation of x.

entropy statistic from Eq. (2.8) calculated for the periodic and chaotic simulations of the Rössler system defined in Eq. (2.26) when additive noise is present in the signal. We incremented the additive noise using the Signal-to-Noise Ratio (SNR). The SNR (units of decibels) is defined as $\text{SNR} = 20\log_{10}(A_{\text{ signal}}/A_{textnoise})$, where $A_{\text{ signal}}$ and $A_{textnoise}$ are the root-mean-square amplitudes of the signal and additive noise, respectively. This result shows that for this signal the OPN network is only robust down to an SNR of approximately 32 dB of additive Gaussian noise, while the CGSSN is able to separate periodic from chaotic dynamics down to approximately 23 dB. We found similar results for the other 22 dynamical systems investigated in this work.

2.1.4.6 Experimental Results

To validate these tools, we apply them to experimental data collected from a base excited magnetic pendulum [**?**]. This system was shown to exhibit periodic and chaotic behavior under different parameters and the CGSSN persistence diagrams were generated for each case using all

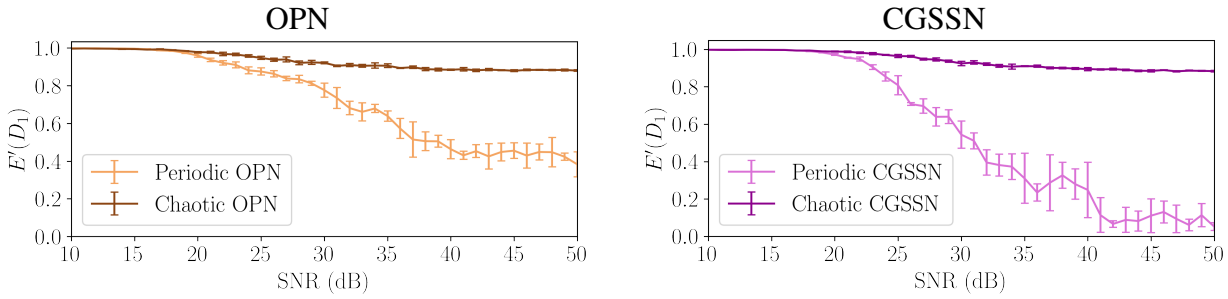


Figure 2.14 Noise robustness analysis of dynamic state detection using the summary statistic persistent entropy (see Eq. (2.8)) for OPN and CGSSN with increasing SNR on a periodic Rossler simulation from Eq. (2.26).

4 distance measures presented in this paper. Figure 2.15 shows the corresponding time series, permutation sequence, CGSSN, and persistence diagrams for the periodic response. We see that for all of the distance metrics, there is a clear singular cycle that forms with a significant persistence lifetime. Conversely, the same results are presented for the chaotic response in Fig. 2.16 where we see a drastically different distribution of persistence pairs corresponding to the high number of cycles present in the chaotic CGSSN. The results presented here are in agreement with our work in [**?**].

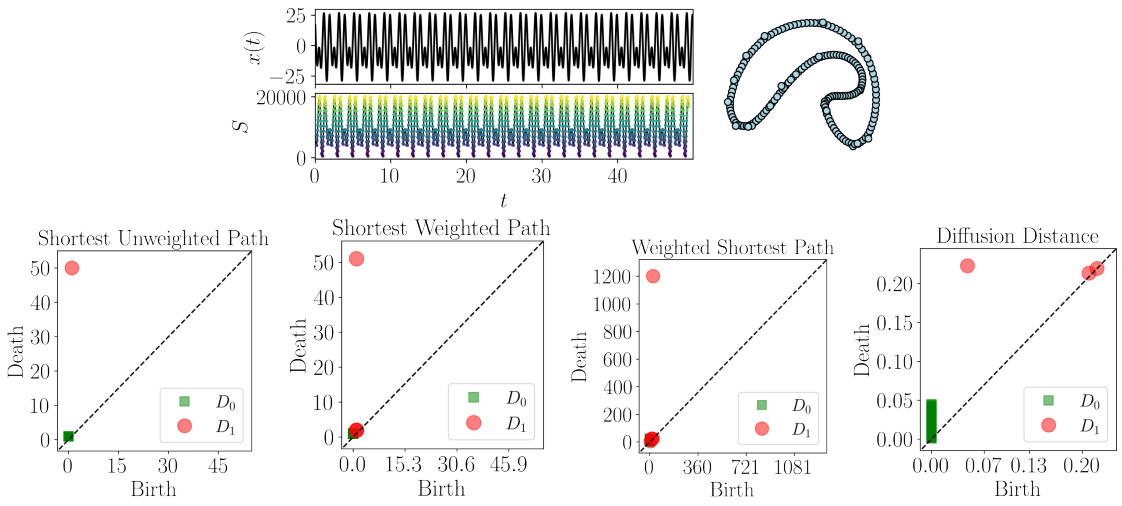


Figure 2.15 CGSSN results for the forced single magnetic pendulum under conditions that yield a periodic response. The top left images show the time series and permutation sequence and the top right shows the coarse grained state space network. The bottom row shows the corresponding persistence diagrams for the network under the distance metric in the title of each diagram.

2.1.5 Conclusion

In this work, we developed a novel framework for studying CGSSNs using persistent homology. We showed that the CGSSN outperformed the standard ordinal partition network in both noise robustness and dynamic state detection performance, with the CGSSN reaching 100% separation

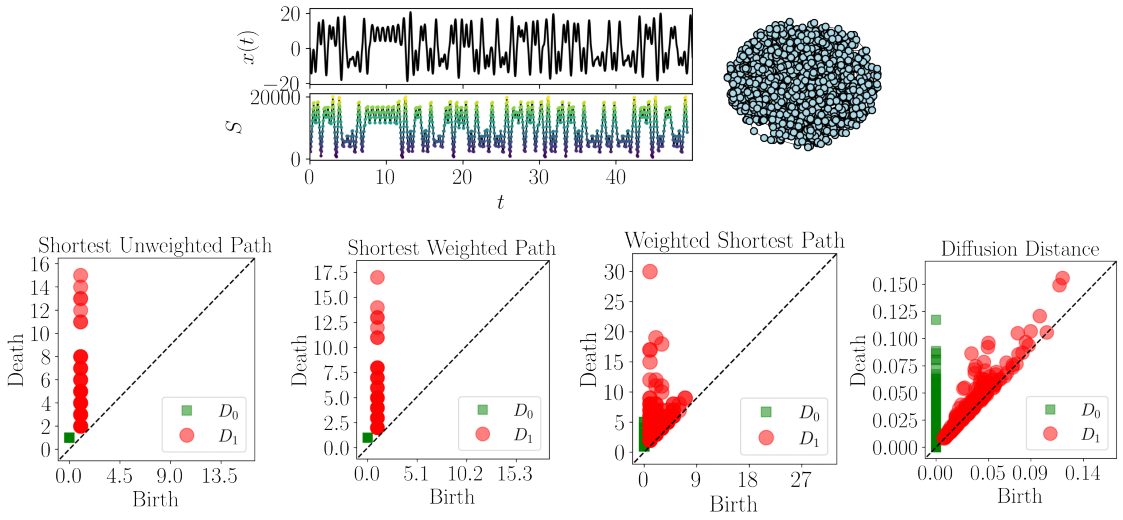


Figure 2.16 CGSSN results for the forced single magnetic pendulum under conditions that yield a chaotic response. The top left images show the time series and permutation sequence and the top right shows the coarse grained state space network. The bottom row shows the corresponding persistence diagrams for the network under the distance metric in the title of each diagram.

accuracy for dynamic state detection for 23 continuous dynamical systems. This is in comparison to the OPN, which could at most reach 95% accuracy. This approach was validated using data from a magnetic pendulum experiment to show that the topological structure for periodic and chaotic timseries are captured in the resulting persistence diagrams.

In this work, we only investigated the most straightforward construction of the CGSSN. Namely, the equal-sized hyper-cube tessellation cover of the SSR domain. Possible improvements to the CGSSN could be through a data-dependent adaptive cover algorithm. We also suspect that other choices of distances could provide improvements for the given pipeline.

Another future direction would be to prove a stability theorem for the CGSSN. That is, can we show that for a noisy version of a signal, the resulting CGSSN, and subsequently the computed persistence diagram, is similar to the ground truth. It would also be interesting to study how the CGSSN could serve to detect quasiperiodicity. We believe that the torus shape associated to the SSR of quasiperiodic signals could be captured using the CGSSN as it accounts for the signal amplitude.

2.1.6 Appendix — Coarse Graining Size Analysis

To determine the optimal binning size we we investigate how the structure of the resulting CGSSN changes as more states are used with b increasing. We considered $b \in [2, 20]$ as more than 20 bins per dimension becomes computationally expensive without increasing the performance (see Fig. 2.17). To summarize the shape of the network we use the maximum lifetime of one-dimensional features (loops) as $\max(L_1)$ and the normalized persistence entropy $E'(D_1)$ defined in Eq. (2.8) using the shortest unweighted path distance. The goal is to find a fine enough granularity (large enough b) that a periodic, noise-free signal will create a signal loop structure in the CGSSN. This loop structure should result with a persistent entropy of approximately zero. The idea behind this is based on a periodic attractor's SSR never intersecting if a suitably high dimension is selected.

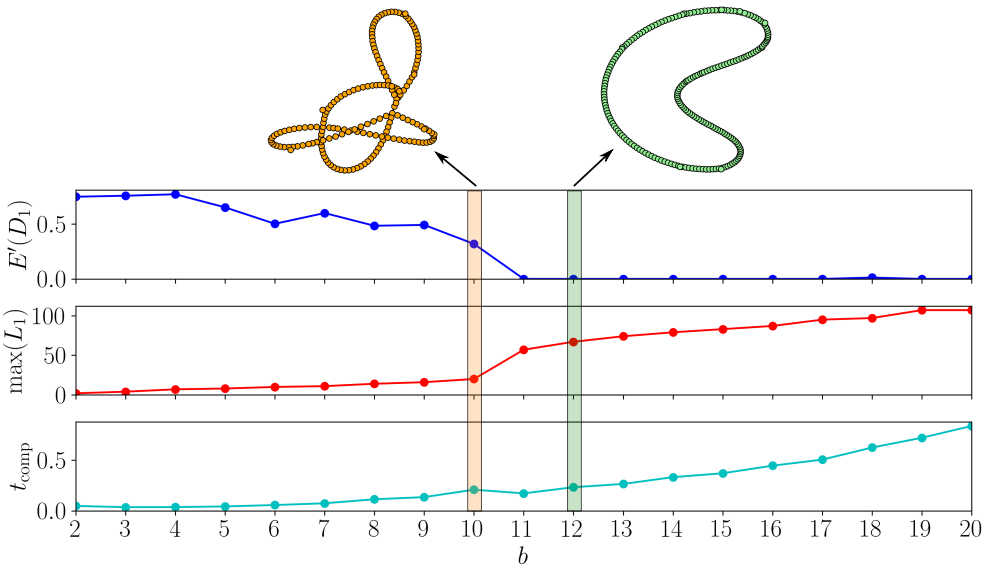


Figure 2.17 Normalized persistent entropy $E'(D_1)$, the maximum lifetime $\max(L_1)$, and computation time t_{comp} for the CGSSN formed with dimension $n = 4$ and $b \in [2, 20]$ for the Rossler system in Eq. (2.26) with example CGSSNs shown at $b = 10$ and $b = 12$.

We point the reader to our work in [69] for a comprehensive analysis to choosing a suitable embedding dimension for the problem. It was found that dimensions of $n = 4$ or 5 are suitable for most continuous systems. For the 23 dynamical systems selected a dimension $n = 4$ is greater than the dimension of the attractor and will be used unless otherwise stated. Let us first investigate a suitable number of bins b for the Rossler system defined in Eq. (2.26) with the $E'(D_1)$, $\max(L_1)$, and computation time t_{comp} calculated as b is increased from 2 to 20 shown in Fig. 2.17. This result show a sudden drop in $E'(D_1)$ and increase in $\max(L_1)$ from going from 10 to 11 bins. This is due the the granularity of the coarse-graining procedure being fine enough that the hypercubes do not capture multiple segments of the periodic flow. This is shown with the two CGSSNs at $b = 10$ and $b = 12$ where at $b = 10$ we have multiple intersections of the network while at $b = 12$ there are no intersections and we only have a single loop structure. Another characteristic is the exponentially increasing computation time t_{comp} as b increases. As such, we want to optimize the choice of b to capture the necessary complexity of the attractor while also minimizing the computation time. For this example a suitable $b = 12$ would be the best choice.

The next question we want to ask is if $b = 12$ is a good option for other dynamical systems. To test this we again calculate the $E'(D_1)$ and $\max(L_1)$ for $b \in [2, 20]$ for the 23 dynamical systems listed in Table 2.2. Figure 2.18 shows these statistics for all of the dynamical systems and it demonstrates that a choice of $b \in [11, 13]$ does work well for all of the dynamical systems with a drop in $E'(D_1)$. Based on this seemingly universal choice of b in this work we use $b = 12$ unless otherwise stated.

2.1.7 Appendix — Data

In this work we heavily rely on a 23 dynamical systems commonly used in dynamical systems analysis. All of these systems are continuous flow opposed to maps. The 23 systems are listed in

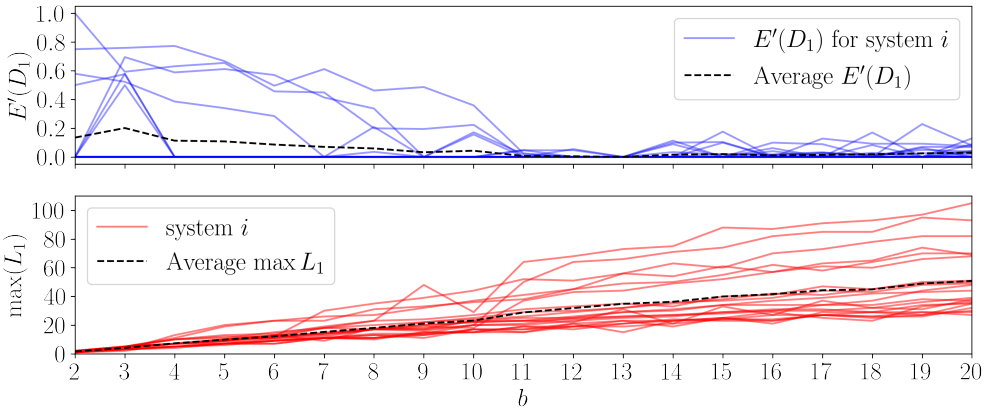


Figure 2.18 Binning size analysis using the normalized persistent entropy $E'(D_1)$ and maximum lifetime $\max(L_1)$ for 23 dynamical systems listed in Table 2.2 with $b \in [2,20]$.

Table 2.2 Continuous dynamical systems used in this work.

Autonomous Flows	**Driven Dissiptive Flows**
Lorenz	Driven Van der Pol Oscillator
Rossler	Shaw Van der Pol Oscillator
Double Pendulum	Forced Brusselator
Diffusionless Lorenz Attractor	Ueda Oscillator
Complex Butterfly	Duffing Van der Pol Oscillator
Chen's System	Base Excited Magnetic Pendulum
ACT Attractor	
Rabinovich Frabrikant Attractor	
Linear Feedback Rigid Body Motion System	
Moore Spiegel Oscillator	
Thomas Cyclically Symmetric Attractor	
Halvorsen's Cyclically Symmetric Attractor	
Burke Shaw Attractor	
Rucklidge Attractor	
WINDMI	
Simplest Cubic Chaotic Flow	

Table 2.2. The equations of motion for each systems can be found in the python topological signal processing package `Teaspoon` under the module *MakeData* `https://lizliz.github.io/teaspoon/`. Specifically, these systems are described in the dynamical systems function of the make data module [70].

Each system was solved to have a time delay $\tau = 50$, which was estimated from the multiscale permutation entropy method [69]. The signals were simulated for $750\tau/f_s$ seconds with only the last fifth of the signal used to avoid transients. It should be noted that we did not need to normalize the amplitude of the signal since the ordinal partition network is not dependent on the signal amplitude.

2.1.8 Additional Results

Here we provide the additional SVM projections to visualize the dynamic state detection performance of the shortest path distances: unweighted shortest path, shortest weighted path, and weighted shortest path. Table 2.1 provides the corresponding average accuracies.

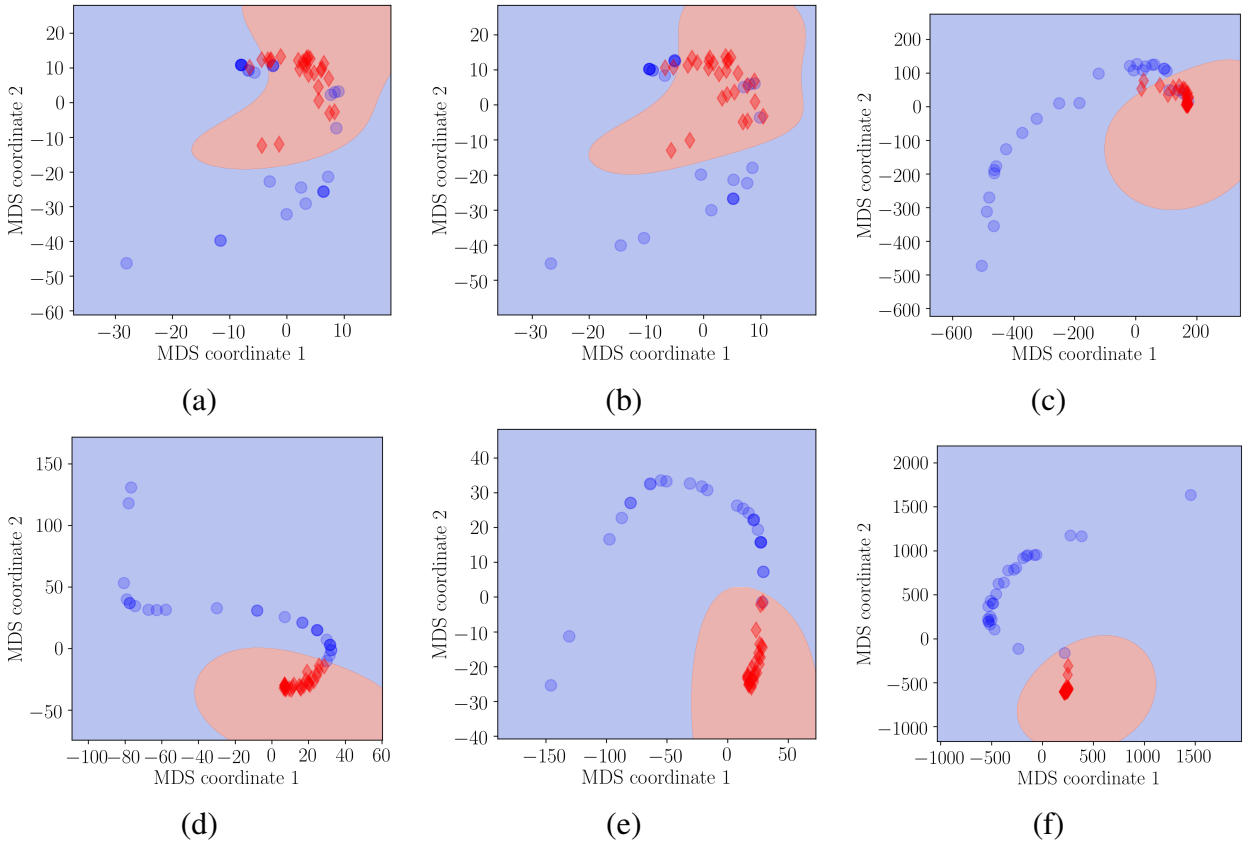


Figure 2.19 Two dimensional MDS projection of the bottleneck distances between persistence diagrams of the chaotic and periodic dynamics with an SVM radial bias function kernel separation. This separation analysis was repeated for the OPNs and CGSSNs using the unweighted shortest path, shortest weighted path, and weighted shortest path distances. (a) Unweighted shortest path distance of OPN, (b) Shortest weighted path distance of OPN, (c) Weighted shortest path distance of OPN, (d) Unweighted shortest path distance of CGSSN, (e) Shortest weighted path distance of CGSSN, (f) Weighted shortest path distance of CGSSN.

2.2 Timeseries Embedding Delay Estimation with TDA

Permutation Entropy (PE) is a powerful tool for quantifying the complexity of a signal which includes measuring the regularity of a time series. Additionally, outside of entropy and information theory, permutations have recently been leveraged as a graph representation, which opens the door for graph theory tools and analysis. Despite the successful application of permutations in a variety of scientific domains, permutations requires a judicious choice of the delay parameter τ and dimension n. However, n is typically selected within an accepted range giving optimal results for the majority of systems. Therefore, in this work we focus on choosing the delay parameter,

while giving some general guidance on the appropriate selection of n based on a statistical analysis of the permutation distribution. Selecting τ is often accomplished using trial and error guided by the expertise of domain scientists. However, in this paper, we show how persistent homology, a commonly used tool from Topological Data Analysis (TDA), provides methods for the automatic selection of τ. We evaluate the successful identification of a suitable τ from our TDA-based approach by comparing our results to both expert suggested parameters from published literature and optimized parameters (if possible) for a wide variety of dynamical systems.

Shannon entropy, which was introduced in 1948 [**?**], is a summary statistic measuring the regularity of a dataset. Since then, several new forms of entropy have been popularized for time series analysis. Some examples include approximate entropy [**?**], sample entropy [**?**], and permutation entropy (PE) [**?**]. While all of these methods measure the regularity of a sequence, PE does this through the motifs or permutations found within the signal. This allows for PE to be related to predictability [**?**, **?**], which is useful for detecting dynamic state changes. Similar to Shannon Entropy, PE [**?**] is quantified as the summation of the probabilities of a data type (see Eq. (2.9)), where the data types for PE are permutations (see Fig. 2.20), which we represent as π. Permutations have recently been used in other applications such as ordinal partition networks [**?**,51] and the conditional entropy of these networks [**?**]. The permutation parameters n and τ represent the permutation size and spacing, respectively. More specifically, τ is the embedding delay lag applied to the series and n is a natural number that describes the dimension of the permutation. In this study we focus on selecting τ using methods based in Topological Data Analysis (TDA) since the dimension is typically chosen in the range $3 < n \leq 7$ for most applications [**?**]. However, we still provide a novel and simple guidance on the automatic selection of n based on a statistical analysis of the permutations.

Currently, the most common method for selecting PE parameters is to adopt the values suggested by domain scientists. For example, Li et al. [**?**] suggest using $\tau = 3$ and $n = 3$ for electroencephalographic (EEG) data, Zhang and Liu [**?**] suggest $\tau = 3$ or 5 and $n \in [3,5]$ for logistic maps, and Frank et al. [**?**] suggest $\tau = 2$ or 3 and $n \in [3,7]$ for heart rate applications. One main disadvantage of using suggested parameter values for an application is the high dependence of PE on the sampling frequency. As an example, Popov et al. [**?**] showed the importance of considering the sampling frequency when selecting τ for an EEG signal. Another limitation is the need for application expertise in order to determine the needed parameters. This can hinder using PE in new applications that have not been sufficiently explored. Consequently, there is a need for an automatic, application-independent parameter selection algorithm for PE.

Several methods have been developed for estimating τ for phase space reconstruction via Takens embedding [**?**]. Some of which include mutual information [71], autocorrelation [**?**], and phase space methods [**?**]. There has also been recent work in determining if these methods are suitable for the delay parameter selection for permutations [69]. Outside of this, a general framework for selecting both n and τ was introduced in [**?**]. In this manuscript our goal is to determine other, TDA-based methods for selecting τ for PE and to draw a connection between permutations and state space reconstruction. Specifically, in Section 2.2.3.2, we relate permutations to state space reconstruction to provide a justification for using the lag parameter from the latter to select τ for the former. We then present a novel TDA-based tool for finding τ. For our approach we compute the 0-D sublevel set persistence in both time and frequency domains to obtain approximations of the maximum significant frequency. We then utilize Nyquist's sampling theorem to find an appropriate τ value.

To determine the viability of our methods, PE parameters are generated and compared to expert suggested values, optimal parameters based on maximizing the difference between permutation entropy for periodic and chaotic signals, and the delay corresponding to the first minima of mutual information. The method of mutual information was chosen as a basis for comparison based on its accuracy in selecting τ for PE as demonstrated in [69].

2.2.1 Permutation Entropy Example

Permutation entropy $H(n)$ for permutation dimension n is calculated according to [**?**] as

$$H(n) = -\sum p(\pi_i) \log p(\pi_i), \tag{2.9}$$

where $p(\pi_i)$ is the probability of a permutation π_i, and $H(n)$ has units of bits when the logarithm is of base 2. The permutation entropy parameters τ and n are used when selecting the permutation size: τ is the number of time steps between two consecutive points in a uniformly sub-sampled time series, and n is the permutation length or motif dimension. Using a real-valued data set X and a measurement of the set $x_i \in X$, we can define the vector $v_i = [x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+(n-1)\tau}]$, which has the permutation π_i. To better understand the possible permutations, consider an example with third degree ($n = 3$) permutations. This results in six possible motifs as shown in Fig. 2.20.

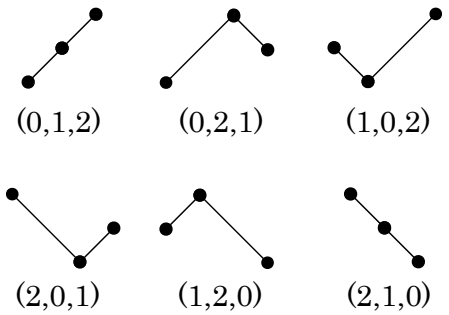


Figure 2.20 All possible permutation configurations (motifs) for n = 3, where $[\pi_1 \ldots \pi_6] = [(0,1,2) \ldots (2,1,0)]$.

Next, to further demonstrate PE with an example, consider the sequence $X = [4,7,9,10,6,11,3]$ with third order permutations $n = 3$ and time delay $\tau = 1$. The sequence can be broken down into the following permutations: two $(0,1,2)$ permutations, one $(1,0,2)$ permutation, and two $(1,2,0)$ permutations for a total of 5 permutations. Applying Eq. (2.9) yields

$$H(3) = -\frac{2}{5}\log\frac{2}{5} - \frac{2}{5}\log\frac{2}{5} - \frac{1}{5}\log\frac{1}{5} = 1.522 \,\text{bits}.$$

The permutation distribution can be visually understood by illustrating the probabilities of each permutation as separate bins. To accomplish this, Fig. 2.21 was created by taking the same series X (Fig. 2.21a) and placing the abundance of each permutation into its respective bin (Fig. 2.21b). PE is at a maximum when all $n!$ possible permutations are evenly distributed or, equivalently, when the permutations are equiprobable with $p(\pi_i) = p(\pi_1), p(\pi_2), \ldots, p(\pi_{n!}) = \frac{1}{n!}$. From this, the maximum permutation entropy $H_{\max}$ is quantified as

$$\begin{aligned} H_{\max}(n) &= -\sum p(\pi_i) \log p(\pi_i) \\ &= -\log\frac{1}{n!} = \log n!. \end{aligned} \tag{2.10}$$

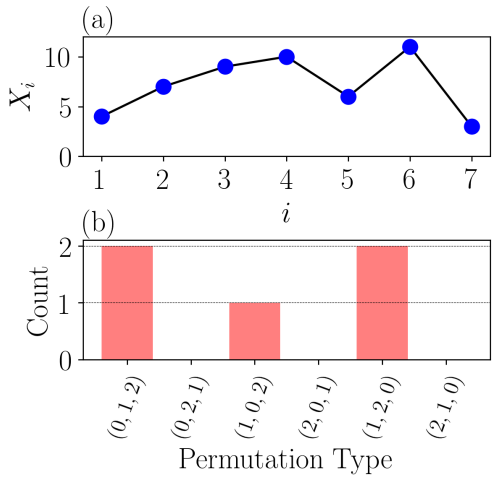


Figure 2.21 Abundance of each permutation from example data set X.

Applying Eq. (2.10) for $n = 3$ yields a maximum PE of approximately 2.585 bits. Using the maximum possible entropy $\log_2 n!$, the normalized permutation entropy is calculated as

$$h_n = -\frac{1}{\log_2 n!} \sum p(\pi_i) \log_2 p(\pi_i). \tag{2.11}$$

Applying Eq. (2.11) to the example series X results in $h_3 \approx 0.5888$.

2.2.2 An overview of tools from TDA

Our TDA-based approaches for finding the delay dimension employs two types of persistence applied to two different types of data. Specifically, in our approach we combine the 0-D sublevel persistence of one-dimensional time series with the z-score. This section provides a basic background of the topics needed to intuitively understand the subsequent analysis. More specifics can be found in [9–11, 13].

2.2.2.1 Simplicial complexes

An abstract k-simplex σ is defined as a set of $k+1$ indices where $\dim(\sigma) = k$. If we apply a geometric interpretation to a k-simplex, we can think of it as a set V of $k+1$ vertices. Using this interpretation, a 0-simplex is a point, a 1-simplex is an edge, a 2-simplex is a triangle, and higher dimensional versions can be similarly obtained.

A simplicial complex K is a set of simplices $\sigma \subseteq V$ such that for every $\sigma \in K$, all the faces of σ, i.e., all the lower dimensional component simplices $\sigma' \subset \sigma$ are also in K. For example, if a triangle (2-simplex) is in a simplicial complex K, then so are the edges of the triangle (1-simplices) as well as all the nodes in the triangle (0-simplices). The dimension of the resulting simplicial complex is given by the largest dimension of its simplices according to $\dim(K) = \max_{\sigma \in K} \dim(\sigma)$. The n-skeleton of a simplicial complex $K^{(n)}$ is the restriction of the latter to its simplices of degree at most n, i.e., $K^{(n)} = \{\sigma \in K \mid \dim(\sigma) \leq n\}$.

Given an undirected graph $G = (V, E)$ where V are the vertices and E are the edges, we can construct the clique (or flag) complex

$$K(G) = \{\sigma \subseteq V \mid uv \in E \text{ for all } u \neq v \in \sigma\}.$$

2.2.2.2 Homology

If we fix a simplicial complex K, then homology groups can be used to quantify the 1-dimensional topological features of the structure in different dimensions. 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, while the rank of $H_2(K)$ is the number of voids, and so on. The homology groups are constructed using linear transformations termed boundary operators.

To describe the boundary operators, we let $\{\alpha_\sigma\}$ be coefficients in a field F (in this paper we choose $F = \mathbb{Z}_2$). Since we are using the field $\mathbb{Z}_2$ we do not need to consider the orientation of the simplicial complex. Then $K^{(n)}$, the n-skeleton of K, can be used as a generating set of the F-vector space $\Delta_n(K)$. In this representation, any element of $\Delta_n(K)$ can be written as a finite formal sum $\sum_{\sigma \in K^{(n)}} \alpha_\sigma \sigma$ called an n-chain. Further, elements in $\Delta_n(K)$ are added by adding their coefficients. The group of all n-chains is the nth chain group $\Delta_n(K)$, which is a vector space. Given a simplicial complex K, the boundary map $\partial_n : \Delta_n(K) \to \Delta_{n-1}(K)$ is defined by

$$\partial_n([v_0, \ldots, v_n]) = \sum_{i=0}^{n} (-1)^i [v_0, \ldots, \hat{v}_i, \ldots, v_n],$$

where $\hat{v}_i$ denotes the absence of element v_i from the set. This linear transformation maps any n-simplex to the sum of its codimension 1 (codim-1) faces. The geometric interpretation of the boundary operator is that it yields the orientation-preserved boundary of a chain.

By combining boundary operators, we obtain the chain complex

$$\ldots \xrightarrow{\partial_{n+1}} \Delta_n(K) \xrightarrow{\partial_n} \ldots \xrightarrow{\partial_1} \Delta_1(K) \xrightarrow{\partial_0} 0,$$

where the composition of any two subsequent boundary operators is zero, i.e., $\partial_n \circ \partial_{n+1} = 0$. An n-chain $\alpha \in \Delta_n(K)$ is a cycle if $\partial_n(\alpha) = 0$; it is a boundary if there is an $n+1$-chain β such that $\partial_{n+1}(\beta) = \alpha$. Define the kernel of the boundary map ∂_n using $Z_n(K) = \{c \in \Delta_n(K) : \partial_n c = 0\}$, and the image of ∂_{n+1} $B_n(K) = \{c \in \Delta_n(K) : c = \partial_{n+1} c', c' \in \Delta_{n+1}(K)\}$. Consequently, we have $B_k(K) \subseteq Z_k(K)$. Therefore, we define the nth homology group of K as the quotient group $H_n(K) = Z_n(K)/B_n(K)$. In this paper, we only need 0-dimensional persistent homology, and we always assume homology with $\mathbb{Z}_2$ coefficients which removes the need to keep track of orientation. In the case of 0-dimensional homology, there is a unique class in $H_0(K)$ for each connected component of K. For 1-dimensional homology, there is one homology class in $H_1(K)$ for each *hole* in the complex and so on for higher dimensional invariants.

2.2.2.3 Filtration of a simplicial complex

Now, we are interested in studying the structure of a changing simplicial complex. We introduce a real-valued filtration function on the simplicies of K such that $f(\tau) \leq f(\sigma)$ for all $\tau \leq \sigma$ simplices in K. If we let $\{y_1 < y_2 < \ldots < y_\ell\}$ be the set of the sorted range of f for any $y \in \mathbb{R}$, then the filtration of K with respect to f is the ordered sequences of its subcomplexes

$$\emptyset \subseteq K(y_1) \subseteq K(y_2) \subseteq \cdots \subseteq K(y_\ell) = K.$$

The sublevel set of K corresponding to y is defined as

$$K(y) = \{\sigma \in K \mid f(\sigma) \leq y\}, \tag{2.12}$$

where each of the resulting $K(y)$ is a simplicial complex, and for any $y_1 \leq y_2$, we have $K(y_1) \subseteq K(y_2)$.

The filtration of K enables the investigation of the topological space under multiple scales of the output value of the filtration function f. In this paper we consider a filtration function which corresponds to 0D sublevel persistence applied to 1-D time series data.

0-D persistence applied to 1-D time series: Let χ be the time-ordered set of the critical values of a time series. Here, we can think of the simplicial complex $K = G(V,E)$ containing a number of vertices $|V|$ equal to the number of critical values in the time series and only the edges E that connect adjacent vertices. i.e., vertices $\{v_i \mid 1 \leq i \leq n\}$, and edges $\{v_i v_{i+1} \mid 1 \leq i \leq n-1\}$. Therefore, we have a one-to-one correspondence between the critical values in χ and the vertices of the simplicial complex V. We define the filtration function for every face σ in K according to

$$f(\sigma_i) = \begin{cases} \chi_i & \text{if } \sigma_i \in V, \\ \max(u,v) & \text{if } \sigma_i = uv \in E. \end{cases}$$

Using this filtration function in Eq. (2.12), we can define an ordered sequence of subcomplexes where $y \in [\min(\chi), \max(\chi)]$.

2.2.2.4 Persistent homology

Persistent homology is a tool from topological data analysis which can be used to quantify the shape of data. The main idea behind persistent homology is to watch how the homology changes over the course of a given filtration.

Fix a dimension n, then for a given filtration

$$K_1 \subseteq K_2 \subseteq \cdots \subseteq K_N$$

we have a sequence of maps on the homology

$$H_n(K_1) \to H_n(K_2) \to \cdots \to H_n(K_N).$$

We say that a class $[\alpha] \in H_n(K_i)$ is born at i if it is not in the image of the map $H_n(K_{i-1}) \to H_n(K_i)$. The same class dies at j if $[\alpha] \neq 0$ in $H_n(K_{j-1})$ but $[\alpha] = 0$ in $H_n(K_j)$.

This information can be used to construct a persistence diagram as follows. A class that is born at i and dies at j is represented by a point in $\mathbb{R}^2$ at (i, j). The collection of the points in the persistence diagram, therefore, give a summary of the topological features that persists over the defined filtration. See the example of Fig. 2.22 for $n = 0$ and time series data.

2.2.3 Permutation Delay

To form permutations from a time series a delay embedding is applied to a uniform subsampling of the original time series according to the embedding parameter τ. For example, the subsampled sequence X with elements $\{x_i : i \in \mathbb{N} \cup 0\}$ subject to the delay τ is defined as $X(\tau) = [x_0, x_\tau, x_{2\tau}, \ldots]$. Riedl et al. [**?**] showed that PE is sensitive to the time delay, which prompts the need for a robust method for determining an appropriate value for τ. For estimating the optimal τ, we will be investigating the following methods in the subsequent sections: Mutual Information (MI) in Section 2.2.3.1, and combining 0-D persistence with the frequency and time domains (Section

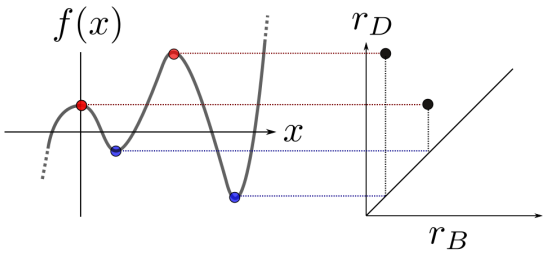


Figure 2.22 Example formulation of a persistence diagram based on 0-D sublevel sets.

2.2.3.3). We recognize, but do not investigate, some other commonly used methods for finding τ. These include the autocorrelation function [**?**] and the phase space expansion [**?**].

Before introducing the TDA-based methods, in Section 2.2.3.2 we elucidate a connection between the permutation delay parameter and the state space reconstruction delay parameter for Takens' embedding to justify the use of methods such as MI and the TDA-based methods.

2.2.3.1 Mutual Information

A common method for selecting the delay τ for state space reconstruction is through a measure of the mutual information between a time series $x(t)$ and its delayed version $x(t+\tau)$ [71]. Mutual information is a measurement of how much information is shared between two sequences, and it was first realized by Shannon et al. [**?**] as

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}, \tag{2.13}$$

where X and Y are separate sequences, $p(x)$ and $p(y)$ are the probability of the element x and y separately, and $p(x,y)$ is the joint probability of x and y. Fraser and Swinney [71] showed that for a chaotic time series the mutual information between the sequence $x(t)$ and $x(t+\tau)$ will decrease as τ increases until reaching a minimum. At this delay τ, the individual data points share minimum amount of information, thus indicating that the data points are sufficiently separated. While this delay value was specifically developed for phase space reconstruction from a single time series, we show in Section 2.2.3.2 how this delay parameter selection method is also appropriate for permutation entropy. Due to this relationship and the result in [69] suggesting the use of MI for PE delay selection, we benchmark the delays τ obtained from our methods against their MI counterparts.

2.2.3.2 Relating Permutations to Delay Reconstruction

The goal in this section is to relate the distribution of permutations formed from a given delay τ to the state space reconstruction with the same delay τ. This connection will show the time delay for both permutations and state space reconstruction are related.

Let us first start by describing the process for state space reconstruction and its similarity to permutations. As described by Takens' [**?**], we can reconstruct an attractor that is topologically equivalent to the original attractor of a dynamical system through embedding a 1-D signal into $\mathbb{R}^n$ by forming a cloud of delayed vectors as $v_i = [x(t_i), x(t_{i+\tau}), x(t_{i+2\tau}), \ldots, x(t_{i+(n-1)\tau})]$ for $i \in [0, L-n\tau]$, where L is the length of the discretely and uniformly sampled signal. Permutations are

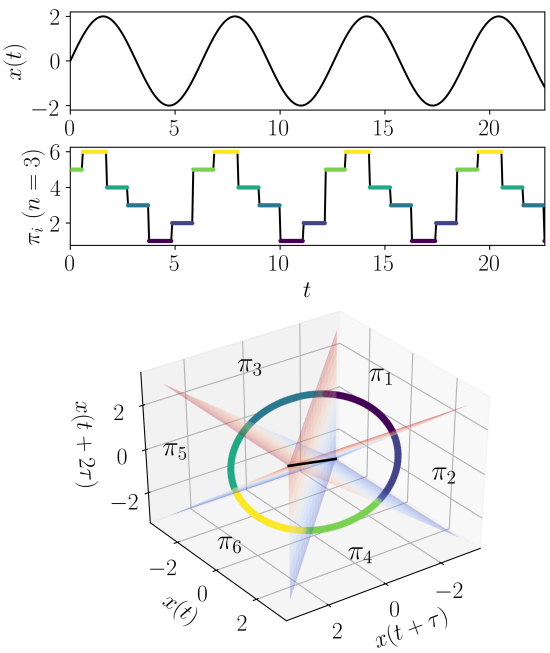


Figure 2.23 Example formation of a permutation sequence from the time series $x(t) = 2\sin(t)$ with sampling frequency $f_s = 20$ Hz, permutation dimension $n = 3$ and delay $\tau = 40$. The corresponding time-delay embedded vectors from $x(t)$ with the permutation binnings $(\pi_1, \ldots, \pi_6)$ in the state space are shown in the bottom figure.

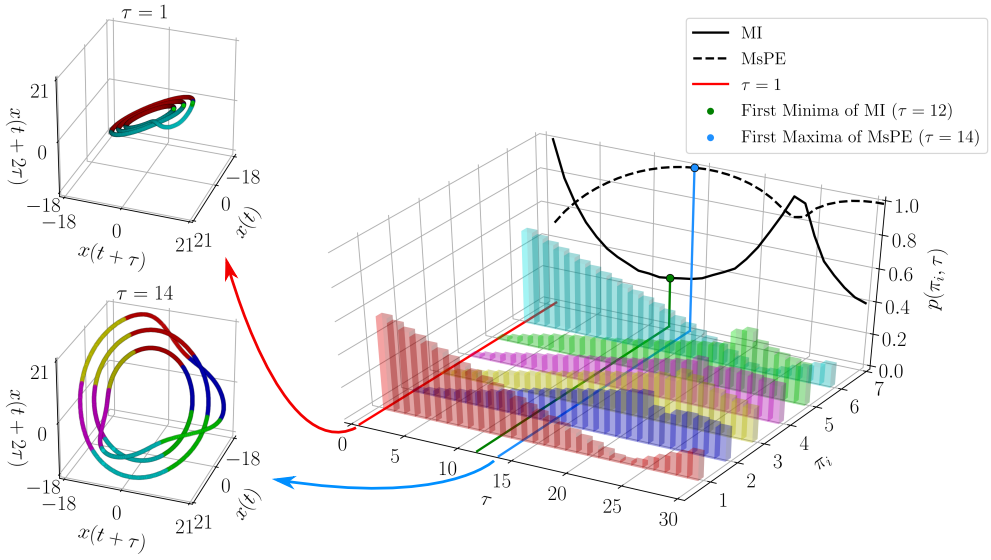


Figure 2.24 Example comparing first minima of mutual information and first maxima of multi-scale permutation entropy, which demonstrates the correspondence between the two. On the left are the $n = 3$ time delayed state space reconstructions with an inaccurately chosen $\tau = 1$ and appropriate $\tau = 14$. On the right shows the permutation distribution as τ increases and the associated multi-scale permutation entropy and mutual information plots.

formed in a very similar fashion where we take our vectors v_i and find their symbolic representation based on their ordinal ranking as explained in Section 2.2.1. The different permutation types can be viewed as an inequality-based binning of the $\mathbb{R}^n$ vector space of the reconstructed dynamics as shown in Fig. 2.23 for dimension $n = 3$. For example, consider the region in Fig. 2.23 labeled π_1. All points on the trajectory shown in this region have the property that $x(t+2\tau) > x(t+\tau) > x(t)$ meaning that points in this region are placed into the group π_1 where all nodes in the permutation are increasing. Similarly, the other sections of the trajectory are appropriately binned to the remaining permutations. This provides a first intuitive understanding of the connection between permutation and state space reconstruction; however, we need to determine if the optimal τ parameter used in v_i is related to the optimal delay in PE.

Takens' embedding theorem explains that, theoretically, any delay τ would be suitable for reconstructing the original topology of the attractor; however, this has the requirement of unrestricted signal length with no additive noise in the signal [**?**]. Since this is rarely a condition found in real-world signals, a τ is chosen to unfold the attractor such that noise has a minimal effect on the topology of the reconstructed dynamics.

Let us now explain what we mean by the correspondence between τ and the unfolding of the dynamics and what effect this has on the corresponding permutations. If the delay τ is too small (e.g. $\tau = 1$ for a continuous dynamical system with a high sampling rate) the delay embedded reconstructed attractor will be clustered around the hyper-diagonal in $\mathbb{R}^n$. Additionally, the corresponding permutations will be overwhelmingly dominated by the permutation types π_1 and $\pi_{n!}$ with these two permutations being of the all increasing and all decreasing ordinal patterns, respectively. The dominance of these two permutations for a delay τ that is too small was termed by Casals et al. [**?**] as the "redundancy effect." For an example of this see the permutation distribution and clustering about the hyper-diagonal in $\mathbb{R}^3$ in Fig. 2.24 when $\tau = 1$. We see that for $\tau = 1$, the trajectory is dominated by π_1 and π_6 permutations, and for this case each group makes up a probability of 50% of the embedded attractor. As the delay increases, the probabilities of the remaining groups begin to increase and the π_1 and π_6 probabilities decrease. These probabilities are generally calculated by the computing the fraction of the trajectory that is occupied by each ordinal pattern. This example is based on the x-solution to the periodic Rossler dynamical system as described in Section 2.2.7.2. As the delay increases beyond the redundancy effect, the reconstructed attractor begins to unfold to have a similar shape and topology as the true attractor. Correspondingly, as the delay increases the permutation distribution tends towards a more equiprobable distribution (See Fig. 2.24 at $\tau \approx 14$).

A way of summarizing the permutation probability distribution is actually through PE itself and more specifically the analysis of Multi-scale Permutation Entropy (MsPE). Riedl et al. [**?**] showed how after the redundancy effect there is a suitable delay for PE, which we related to the first maxima of the MsPE plot [69]. The MsPE plot for our periodic Rossler example is shown in Fig. 2.24. Let us also look at the MI plot as a comparison. The idea behind MI is that at the first minima of the mutual information between $x(t_i)$ and $x(t_{i+\tau})$ the delay τ accurately provides a suitable delay for state space reconstruction. A quick investigation of the MI function reveals a high degree of correlation between the MI function and the MsPE function with the first maxima of MsPE being approximately at the same τ as the first minima of MI. When the delay becomes significantly larger than the first minima of MI or maxima of MsPE, the permutation distribution begins to fluctuate as shown in Fig. 2.24. This effect was termed the "irrelevance effect" by Casdagli et al. [**?**]. This increasing of τ beyond the the first minima also correlates with what Kantz and Schreiber [34]

describe as the reconstruction filling an overly large space with the vectors already being independent. Additionally, at a minima beyond the first minima, Fraser and Swinney [71] showed how the reconstructed attractor shape will no longer qualitatively match the shape of the true state space.

In summary, we have shown a main heuristic result we need to move forward: tools for delay parameter selection for PE can be suitable for state space reconstruction and vice-versa. While we do not provide a proof that PE and state space reconstruction use the same τ, it has recently been shown that a connection between co-homology, information theory, and probability does exist [**?**], which strengthens our qualitative analysis of this connection. In the following sections we leverage tools from TDA to determine the optimal τ associated with the unfolding of the attractor.

2.2.3.3 Finding τ Using Sublevel Set Persistence

In this section our goal will be to leverage sublevel set persistence for the selection of τ for both state space reconstruction and permutation entropy. Specifically, our goal is to automate the frequency analysis method [**?**] for selecting τ for state space reconstruction by analyzing both the time and frequency domain of the signal using sublevel set persistence. Melosik and Marszalek [**?**] leveraged Shannon-Nyquist sampling criteria and used the maximum significant frequency $f_{\max}$ and the sampling frequency f_s to select an appropriate τ according to

$$\tau = \frac{f_s}{\alpha f_{\max}}, \tag{2.14}$$

where $\alpha \in [2,4]$. The value $\alpha = 2$ is associated to the Nyquist sampling rate, while $\alpha > 4$ produces an oversampling. Since this method was developed using the Nyquist sampling rate, it is applicable for continuous, band-limited signals. This frequency based approach was used to find suitable delays for the 0/1 test on chaos and heuristically compare the Lorenz attractor and its time-delay reconstruction [**?**]. The heuristic comparison showed that this frequency approach actually provided more accurate delay parameter selections for state space reconstruction than the mutual information function when trying to replicate the shape of the attractor. Unfortunately, a major drawback of this method is the non-trivial selection of $f_{\max}$. In Melosik's and Marszalek's original work [**?**] the maximum frequency was manually selected using a normalized Fast Fourier Transform (FFT) cutoff of approximately 0.01, which does not address the possibility of additive noise.

In [69] the maximum "significant" frequency was approximated in a time series using the FFT and defining a power spectrum cutoff based on the statistics of additive noise in the FFT. An issue with this method for non-linear time series is that the Fourier spectrum does not easily yield itself to selecting the maximum "significant" frequency for chaotic time series even with an appropriately selected cutoff to ignore additive noise. Additionally, the method was only developed for Gaussian White Noise (GWN) contamination of the original time series.

The following sections improve the selection of the maximum significant frequency using two novel methods based on 0-D sublevel set persistence. We chose to use 0-D sublevel set persistence due to its computational efficiency and stability for true peak selection [**?**, 72]. The first method is based on a time domain analysis of the sublevel set lifetimes (see Section 2.2.3.4), and the second implements a frequency domain analysis using sublevel set persistence and the modified z-score (see Section 2.2.3.5).

2.2.3.4 Time Domain Approach

The first approach we implement for estimating the maximum significant frequency of a signal is based on a time domain analysis of the sublevel set persistence. This process uses the time ordered lifetimes from the sublevel set persistence diagram. Time ordered lifetimes and a cutoff separating the sublevel sets associated with noise were previously introduced in [**?**]. Here we use those methods and results to find the time t_B in which all the significant sublevel sets are born. Figure 2.25 shows an example time order lifetimes plot where the time between two adjacent lifetimes is defined as T_{B_i}. If we use T_{B_i} as an approximation of a period in the time series, then

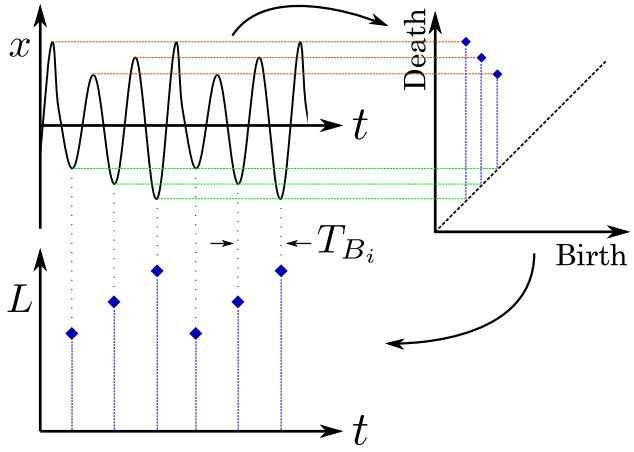


Figure 2.25 Example demonstrating process from time series x (periodic Rossler system) to sublevel set persistence diagram to time ordered lifetimes on the bottom left. Additionally, on the bottom left shows a sample time periodic between sublevel sets as T_{B_i}.

we can calculate the associated frequencies as $f_i = 1/T_{B_i}$ Hz. If we then look at the distribution of f_i, the maximum "significant" frequency can be approximated using the 75% quantile of the distribution of the frequencies as $f_{\max} \approx Q_{75}(f)$. This quantile allows for a few outlier frequencies to occur without having a significant effect on the estimate of the maximum significant frequency.

Applying this method to the periodic Rossler system described in Eq. (2.26) results in $\tau = 23$ with the corresponding state space reconstruction for $n = 2$ shown in Fig. 2.26. This suggested delay is larger than that of mutual information ($\tau = 16$), but relatively close. We also see in Fig. 2.26 that the attractor appears to be circular. Embedding this signal using the suggested delay from [**?**] resulted in an attractor that was more elliptical in shape which is not ideal for attractor reconstruction. This suggests that the time-domain analysis for selecting the maximum frequency and corresponding delay functions can automatically suggest an appropriate delay for permutation entropy and state space reconstruction resulting in a more optimal unfolding of the attractor in both periodic and chaotic cases.

2.2.3.5 Fourier Spectrum Approach

In this section we present a novel TDA based approach for finding the noise floor in the Fourier spectrum for selecting the maximum significant frequency $f_{\max}$ to be used for selecting τ for PE through Eq. (2.14). Specifically, we show how the 0-dimensional sublevel set persistence, a tool from TDA discussed in Section 2.2.2, can be used to find the significant lifetimes and the

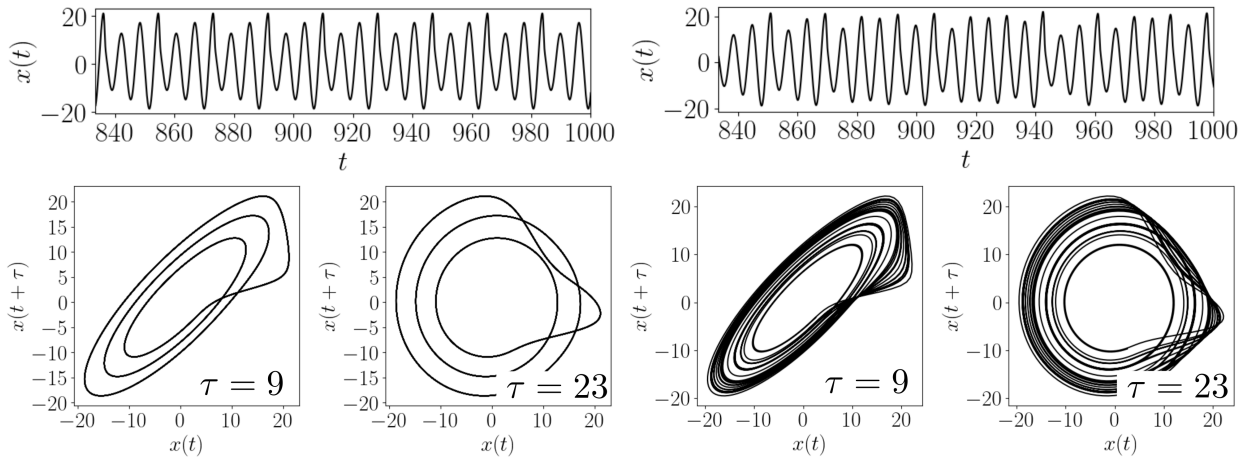


Figure 2.26 Example demonstrating the time delay $\tau = 23$ result for the periodic and chaotic Rossler example time series shown in the top figures and the resulting $n = 2$ Takens' embeddings. The bottom right figures show the embeddings using the delays from [**?**] ($\tau = 9$) and the sublevel persistence method ($\tau = 23$) for both periodic (left) and chaotic (right) signals.

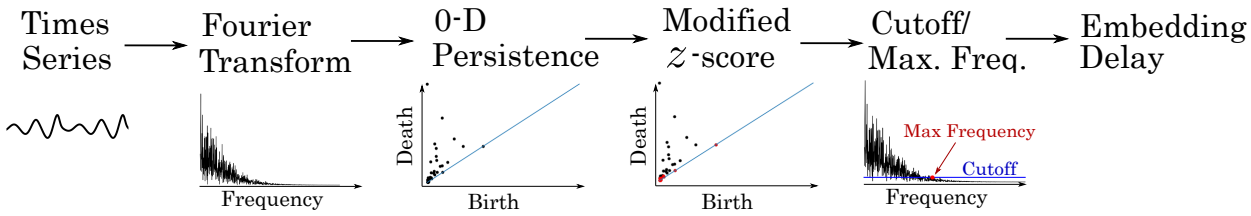


Figure 2.27 Overview of procedure for finding maximum significant frequency using 0-dimensional sublevel set persistence and the modified z-score for a signal contaminated with noise.

associated frequencies in the frequency spectrum. Although it would be ideal to separate the significant lifetimes based on propagating the FFT of a random process into the persistence space, this task is not trivial. There have been studies on pushing forward probability distributions into the persistence domain [**?**, **?**, **?**], but it is difficult to obtain a theoretical cutoff value in persistence space; therefore, we instead separate the noise lifetimes from significant lifetimes through the use of the modified z-score. This separation allows us to find the noise floor and the maximum significant frequency via a cutoff. This process for finding the cutoff and associated maximum frequency is illustrated in Fig. 2.27. The following paragraphs give an overview of the modified z-score and cutoff analysis.

Modified z-score The modified z-score z_m is essential to understanding the techniques used for isolating noise from a signal [**?**]. The standard score, commonly known as the z-score, uses the mean and the standard deviation of a data set to find an associated z-score for each data point and is defined as

$$z = \frac{x - \mu}{\sigma}, \tag{2.15}$$

where x is a data point, μ is the mean, and σ is the standard deviation of the data set, respectively. The z-score value is commonly used to identify outliers in the data set by rejecting points that are above a set threshold, which is set in terms of how many standard deviations away from the mean are acceptable. Unfortunately, the z-score is susceptible to outliers itself with both the mean and the standard deviation not being robust against outliers [**?**]. This led Hampel [**?**] to develop the modified z-score as an outlier detection method that is robust to outliers. The logic behind the modified z-score or median absolute deviation (MAD) method is grounded on the use of the median instead of the mean. The MAD is calculated as

$$\text{MAD} = \text{median}(|\mathbf{x} - \tilde{x}|), \tag{2.16}$$

where $\mathbf{x}$ is an array of data points and $\tilde{x}$ is its median. The MAD is substituted for the standard deviation in Eq. (2.15). To improve the modified z-score, Iglewicz and Hoaglin [**?**] suggested to additionally substitute the mean with the median. The resulting equation for the modified z-score is then

$$z_m = 0.6745 \frac{\mathbf{x} - \tilde{x}}{\text{MAD}}, \tag{2.17}$$

where the value 0.6745 was suggested in [**?**]. We can now use the modified z-score z_m for evaluating the "significance" of each point in the sublevel set persistence diagram of the Fourier spectrum. A threshold for separating noise in the persistence domain is discussed in the following paragraph.

Threshold and Cutoff Analysis To determine the noise floor in the normalized Fast Fourier Transform (FFT) spectrum, we compute the 0-dimensional persistence of the FFT. This provides relatively short lifetimes for the noise, while the prominent peaks, which represent the actual signal, have comparatively long lifetimes or high persistence. To separate the noise from the outliers we calculate the modified z-score for the lifetimes in the persistence diagram. We can then determine if the lifetime is associated to noise or signal based on a z_m cutoff as D, where we can label a lifetime as significant (an outlier) if $z_m > D$. Iglewicz and Hoaglin [**?**] suggest a z_m threshold of $D = 3.5$ based on an analysis of 10,000 random-normal observations. However, we apply both the FFT and 0-D sublevel set persistence to the original signal so we need to determine if this cutoff is also suitable for data that has been processed in this way. To do this we used a signal of 10,000 random-normal observations and applied FFT. We then calculated the 0-D sublevel set persistence and computed the modified z-score z_m using the resulting lifetimes. For an accurate cutoff we would expect to label all of the lifetimes as noise with $z_m < D$ since each signal is composed of pure noise. As shown in Fig. 2.28, a threshold of approximately $D = 4.8$ labels all of the lifetimes as noise. This threshold was rounded up to 5 for simplicity. We can now simply define a cutoff based on the labeling of each lifetime from the modified z-score with Cutoff $= \max(lifetime_{noise})$. We emphasize that this threshold is only appropriate for additive Gaussian white noise and a different cutoff may need to be obtained with a different noise distribution.

We can now find the maximum significant frequency $f_{\max}$ as the highest frequency in the Fourier spectrum with an amplitude greater than the specified cutoff. For this method to accurately function, it is required that there is some additive noise in the time series. This is due to the fact that our pipeline in Fig. 2.27 does not distinguish between signals with and without noise so if a noise-free signal is used, the outliers will drastically alter the results and the cutoff will not be accurate. Adding noise to the signal is meant to increase the contrast between outliers and the true

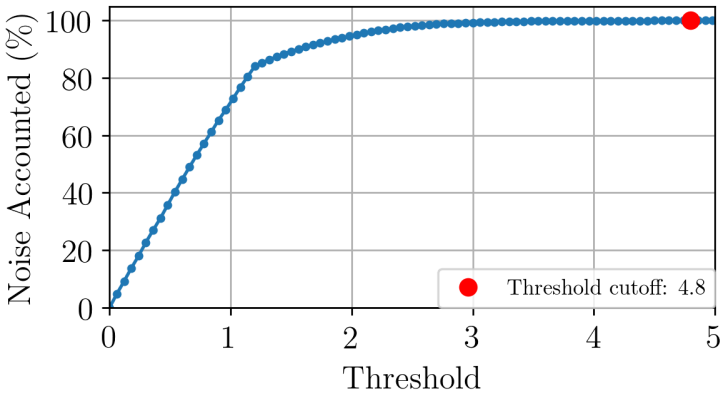


Figure 2.28 Percent of the persistence points from 0-D sublevel set persistence of the FFT of GWN using the modified z-score with the provided threshold ranging from 0 to 5.

signal to allow for a proper cutoff to be obtained. Further, it is uncommon to have a signal without noise in real world situations. To accommodate this for our simulations, additive Gaussian noise with Signal-to-Noise Ratio of 30 dB is added to the time series before calculating the FFT. If we apply this method to the example periodic Rossler system time series we find a suggested delay of $\tau = 15$. Which is close to the delay estimated using mutual information. W will further investigate its accuracy on several other systems in Section 2.2.5 to make more general conclusions on the functionality of this method for selecting τ.

2.2.4 Permutation Dimension

In this section we will show that, contrary to the delay selection, the dimension for permutation entropy is not related to that of Takens' embedding. Additionally, we will provide a simple method for selecting an appropriate permutation dimension based on the permutation distribution.

Permutation entropy is often used to differentiate between the complexity of a time series when there is a dynamic state change (e.g. periodic compared to chaotic), so the dimension should be chosen such that it is large enough to capture these changes. To accomplish this we suggest that permutations of the time series do not occupy all of the possible permutations, but rather only a fraction of the permutations when an appropriate delay is selected. This criteria is set so that a change can be captured by an increase/decrease in the number of permutations and their associated probabilities. Because of this, we suggest a dimension where, at most, only 50% of the permutations are used. However, it may be more suitable to select a dimension where a lower percentage is used (e.g. 10%).

To begin this method for determining if the dimension is high enough to capture the time series complexity we will define N_π as the number of permutation types where the probability of that permutation type is significant. Specifically, we will consider the probability of that permutation to be significant if the number of occurrences of permutation π is greater than 10% of the maximum number of occurrences of any permutation type of dimension n. The permutation delay τ was selected from the expert suggested values provided in [**?**, 69]. We can now express our needed dimension as the ratio and inequality

$$\frac{N_\pi}{n!} \leq R, \tag{2.18}$$

where $R = 0.50$ for the suggested maximum 50% criteria.

To compare this dimension to the standard Takens' embedding tools for selecting n we will implement four examples:

$$\begin{aligned} x_1(t) &= \frac{t}{10} \\ x_2(t) &= \sin(t) \\ x_3(t) &= \sin(t) + \sin(\pi t) \\ x_4(t) &\sim \mathcal{N}(\mu = 0, \sigma^2 = 1), \end{aligned} \tag{2.19}$$

where $t \in [0, 100]$ with a sampling rate of 20 Hz and $\mathcal{N}$ is Gaussian additive noise. By applying Eq. (2.18) to the time series in Eq. (2.19), we can suggest dimensions of 2 for $x_1(t)$, 4 for $x_2(t)$, 6 for $x_3(t)$, and 7 for $x_4(t)$ as shown in Fig. 2.29.

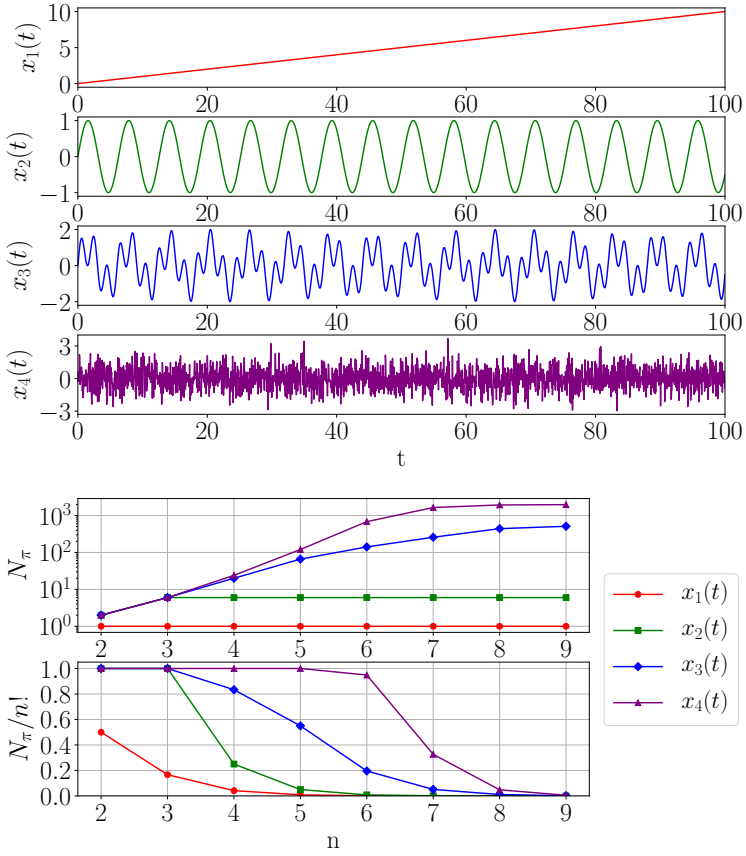


Figure 2.29 Percent of permutations used $R = N_\pi/n!$ for each example time series (see Eq. (2.19)) as the dimension is incremented.

In comparison to Takens' embedding, for time series $x_2(t)$ dimension $n = 2$ would be sufficient, but if this was used for permutation entropy, no increase in complexity could be detected. Additionally, this result suggests an upper bound on the dimension for permutation entropy as $n \approx 9$ as the ratio in Eq. (2.18) is approximately 0 for dimensions $n > 9$. As a rule of thumb from this result, a dimension of 8 would be suitable for almost all applications, but it would be optimal to minimize

the dimension to reduce the computation time of PE. In Section 2.2.5 we will show the resulting suggested dimensions using this method for a wide variety of dynamical systems.

2.2.5 Results

This section provides the results of the parameter selection methods. First, in Section 2.2.5.1, we calculate the delay parameter for a wide variety of dynamical systems and data sets using mutual information and the the automatic TDA-based methods described in this manuscript. Unfortunately, the optimal parameters cannot be decided based on a simple entropy value comparison since there is no direct equivalence between PE and other entropy approximations of a signal such as Kolmogorov-Sinai (KS) entropy with only a bounding between the two as KS $\leq PE$ [**?**]. Therefore, to determine the accuracy of the automatically selected PE parameters we implement two other methods of comparison. The first is a comparison to expert suggested parameters for a wide variety of systems (see Section 2.2.5.1). The second approach is a comparison to optimal parameters based on having a significant difference between the PE of two different states for each system. Of course the second method has the requirement that we have a system model or data set with two different states for comparison, which is not typically the case, but does allow for an approximation of optimal PE parameters for these systems. These comparisons are discussed in Section 2.2.5.1.

The second half of the results, in Sections 2.2.5.5 and 2.2.5.6, is based on analyzing the robustness of the automatic TDA-based PE parameter selection methods to additive noise and signal length, respectively.

2.2.5.1 Parameter Value Comparison for Common Dynamical systems

To determine a range of approximately optimal PE parameters we will quantify the difference between PE values for a wide range of delays and dimensions with the difference for a given τ and n calculated as

$$\Delta h_n(\tau) = h_n^{(\text{Ch.})}(\tau) - h_n^{(\text{Pe.})}(\tau), \tag{2.20}$$

where the superscripts Ch. and Pe. represent the PE calculation on the chaotic and periodic time series for the given dynamical system. The specific parameters used to generate periodic and chaotic responses for each system are described in Appendix 2.2.7. If we apply Eq. (2.20) to the Rossler system for $\tau \in [1,15]$ and $n \in [3,10]$ we find that $\Delta h_n(\tau)$ is significant when $\tau \in [9,15]$ and $n \in [6,10]$ as shown in Fig. 2.30. However, as mentioned previously in Section 2.2.4, dimensions greater than 8 can be computationally expensive. We consider this range where $\Delta h_n(\tau)$ is relatively large as the range of optimal PE parameters to be compared to. We repeated this process for finding the optimal parameter ranges for PE using a similar procedure to this Rossler example as shown in Table 2.3.

To verify our TDA-based methods for determining τ, Table 2.3 compares our results to the values from a wide variety of systems for both the first minima of the mutual information function and from expert suggestions, including several listed by Riedl et al. [**?**]. The table also shows the resulting permutation dimensions suggested from the permutation statistics as described in Section 2.2.4 for both $R = 0.1$ and $R = 0.5$ from Eq. (2.18). For these systems we have also included, where applicable, the delay and dimension parameter estimates for both periodic and chaotic responses to validate each method's robustness to chaos and non-linearity. However, for the medical data section we instead included a healthy/control and unhealthy (arrhythmia for ECG and seizure for EEG) as a substitute for a periodic and chaotic response, respectively. A detailed description of

Cat.	system	State	Delay Sublevel Set Pers. t	Delay Sublevel Set Pers. f	MI	Dim. R 0.5	Dim. R 0.1	Expert Suggested Parameters ø	Expert Suggested Parameters n	Expert Suggested Parameters Ref.	Opt. Param. Range ø	Opt. Param. Range n
Noise Models	Gauss.	-	1	1	3	7	8	1	3-6	[?]	-	-
	Uniform	-	1	1	3	7	8	-	-	-	-	-
	Rayleigh	-	1	1	2	7	8	-	-	-	-	-
	Expon.	-	1	1	2	7	8	-	-	-	-	-
Cont. Flows	Lorenz	Per.	11	7	5	5	6	10	5-7	[?]	8-11	5-10
		Cha.	11	8	11	5	7					
	Rossler	Per.	23	15	16	5	6	9	6	[?]	15-23	6-10
		Cha.	23	15	18	5	6					
	Bi-direct. Rossler	Per.	16	9	14	5	6	15	6-7	[?]	11-22	6-10
		Cha.	16	13	16	5	6					
	Mackey Glass	Per.	2	4	4	5	6	10	4-8	[?]	6-12	4-8
		Cha.	9	3	6	5	7					
	Chua Circuit	Per.	17	11	14	5	6	20	5	[?]	16-24	5-10
		Cha.	17	21	17	5	7					
	Coupled Ross.-Lor.	Per.	4	7	2	4	6	8	3-10	[?]	4-8	4-9
		Cha.	4	5	8	5	7					
	Double Pendul.	Per.	67	44	7	4	5	-	-	-	7-47	5-10
		Cha.	41	28	47	6	7					
Period. Funct.	Periodic	-	13	24	16	4	5	15	4	[?]	-	-
	Quasi	-	25	49	26	6	7	-	-	-	-	-
Maps	Logistic	Per.	1	1	3	4	5	1-5	4-7	[?]	1-4	3-6
		Cha.	1	1	16	4	6					
	Henon	Per.	1	1	3	4	5	1-2	2-16	[?]	1-5	5-8
		Cha.	1	1	16	6	7					
Med. Data	ECG	Cont.	22	7	17	5	6	10-32	3-7	[?]	6-23	5-7
		Arrh.	15	6	15	5	6					
	EEG	Cont.	1	3	6	8	8	1-3	3-7	[?]	2-6	4-7
		Seiz.	12	4	10	5	7					

Table 2.3 A comparison between the calculated and suggested values for the delay parameter τ. The shaded (red) cells highlight the methods that failed to provide a close match to the suggested delay.

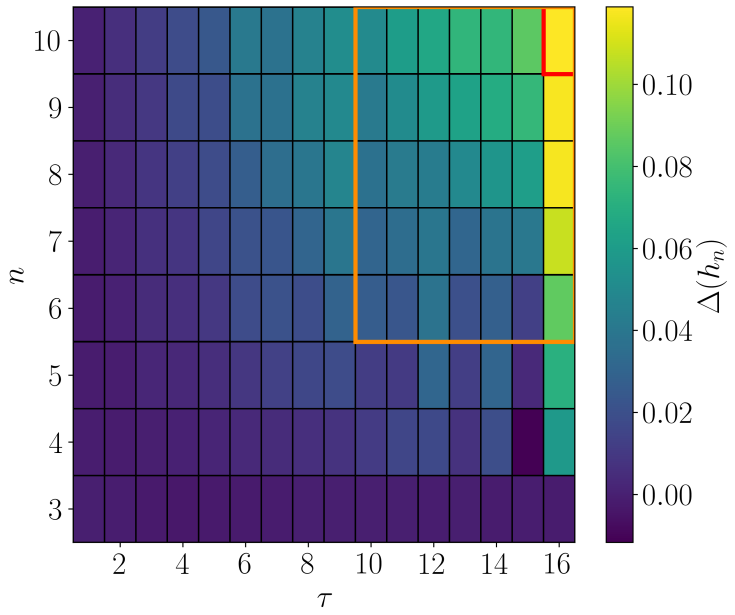


Figure 2.30 Example showing difference in PE (see Eq. (2.20)) for periodic and chaotic dynamic states of the Rossler system for a wide range of PE parameters.

each dynamical system or data set used, including parameters for periodic and chaotic responses, is provided in the Appendix.

In Table 2.3 we have highlighted the methods that failed to provide an accurate delay τ in red. This does not mean the embedding is not optimal it simply means it fell outside of the range of expert suggested delays. We will now go through the methods and highlight the advantages and drawbacks as well as provide general suggestions for which method to use based on the category.

Noise Models: We only have one expert suggestion of parameters for the noise models category, which is for Gaussian white noise (Gauss.) as $\tau = 1$ and $n \in [3,6]$. In regards to the delay, all TDA based methods show an accurate selection of $\tau = 1$, however the suggestion of $\tau = 3$ from Mutual Information (MI) is slightly higher than suggested. We found that the expert suggested dimensions of 3 to 6 is significantly lower than the minimum dimension suggested by our permutation statistics method of $n = 7$. As mentioned in Section 2.2.4, we believe it is necessary to have the number of permutations used to be at least less than 50% of all the permutations available, which corresponds to a dimension $n = 7$ for Gaussian noise. Additionally, as shown in Fig. 2.29, if a dimension of even $n = 6$ is used approximately 95% of the permutations are already in use for Gaussian noise, which would make dynamic state changes difficult or statistically insignificant if there is a complexity increase. From this logic we can then conclude that a suitable dimension should actually be at least $n = 7$ if any increase in the time series complexity is expected. If only decreases in complexity are expected, then a dimension of $n = 6$ may be suitable.

Continuous Flows: The next category is of continuous flows described by systems of non-linear differential equations. As shown in Table 2.3, both the time domain analysis via sublevel set persistence and mutual information provide accurate delay suggestions for many of the examples. We can also conclude that the frequency domain analysis using sublevel set persistence often provided delays that were too small. In regards to the dimension, the suggested dimensions from the permu-

tation statistics agreed with the delay suggested by experts for all of the continuous flow systems. This suggests that our method for selecting a dimension for permutation entropy in Section 2.2.4 is accurate for simulations of continuous differential equations.

Periodic Functions: For periodic functions, including a simple sinuisodal function (periodic) and two incommensurate sinuisoidal functions (quasiperiodic), our results in Table 2.3 show that all methods, including mutual information, provide accurate selections of τ except the Fourier spectrum analysis via sublevel sets. This method results in a significantly larger suggestion for τ. In regards to the dimension selection, our results using the permutation statistics method described in Section 2.2.4 agree with the expert suggested minimum dimension of $n = 4$. For the quasiperiodic function tested, there is no reference or expert suggested delay; however, a discrepancy was observed in the obtained time delays where the frequency domain methods yielded a delay of 49 and the time domain and mutual information approaches resulted in delays of about 25. Therefore the results for the quasiperiodic system are inconclusive.

Maps: When selecting the delay parameter for permutations and takens' embedding for maps we found that all of the topological methods suggested accurate delay parameters, while the standard mutual information method selected overly large delay parameters when the maps are chaotic. Therefore, we suggest the use of one of the topological methods when estimating the delay parameter for maps. For the permutation dimension we found a suggested dimension $n \in [4,7]$, in comparison to the expected suggested dimension ranging from 2 to 16. While the range suggested from the permutations statistics as described in Section 2.2.4 falls within the range suggested by experts, their range is too broad. Specifically, a dimension greater than 9 can be computationally cumbersome, and a dimension lower than 4 would not show significant differences for dynamic state changes. Therefore, we suggest the use of our narrower range of dimension from $n \in [5,6]$ for maps, which agrees with our optimal PE parameter range.

Medical Data: The medical data used in this study inherently has some degree of additive noise, which provides a first glimpse into the noise robustness of the delay parameter selection methods investigated. However, a more thorough investigation will be provided in Section 2.2.5.5. From our analysis, we disagree with the expert suggested delay $\tau \in [1,3]$, but rather suggest the delay selected from either mutual information or the time domain analysis of sublevel set persistence. The general selection for delays between 1 and 3 does not account for the large variation in possible sampling rates. If a small delay is used in conjunction with a high sampling rate, an inaccurate delay could be selected resulting in indistinguishable permutation entropy values as the dynamic state changes. In regards, to the permutation dimension n, we believe that a more appropriate dimension, in comparison to the values suggested by experts, should range between 5 and 7 for medical data applications.

2.2.5.2 A Note on Forecasting Performance

In Table 2.3, we show that a significant proportion of the obtained embedding parameters align with expert suggested values. However, some of the systems resulted in parameters outside of these suggested ranges. This section aims to demonstrate that using the embedding parameters from our TDA-based methods can still provide accurate time series forecasting. To perform the forecasting, an AutoRegressive (AR) model was used to learn from simulation data and fit a model

of the form,

$$x_t = \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t, \tag{2.21}$$

where φ_i are learned coefficients from the simulation data, p is the order of the model and ε_t is Gaussian white noise [**?**]. Note also that more sophisticated forecasting models also exist such as the AutoRegressive Moving Average (ARMA) model [**?**], but for this analysis we consider the simplest such case to demonstrate the accuracy of the estimated embedding parameters and in practice these complex models were found to learn significantly slower compared to the AR model. We assume that we only have access to one state of the system and embed the signal using the appropriate delay and dimension. The embedded signal is then regarded as the ground truth because in practice we do not have access to the remaining states, and Takens' theorem only guarantees a topological equivalence in the reconstructed attractor.

We then fit the autoregressive model to the time series choosing a sufficiently large model order and embed the forecasted signal over a range of delays. These reconstructed attractors are then compared with the embedded attractor from the simulation using the sum of the Root Mean Square Errors (RMSE) across all embedded states of the system and plotted with respect to the embedding delay τ similar to what is done in [34]. If the embedding delay is sufficient for forecasting, we expect that its error will be reasonably close to the error when using the expert suggested delay.

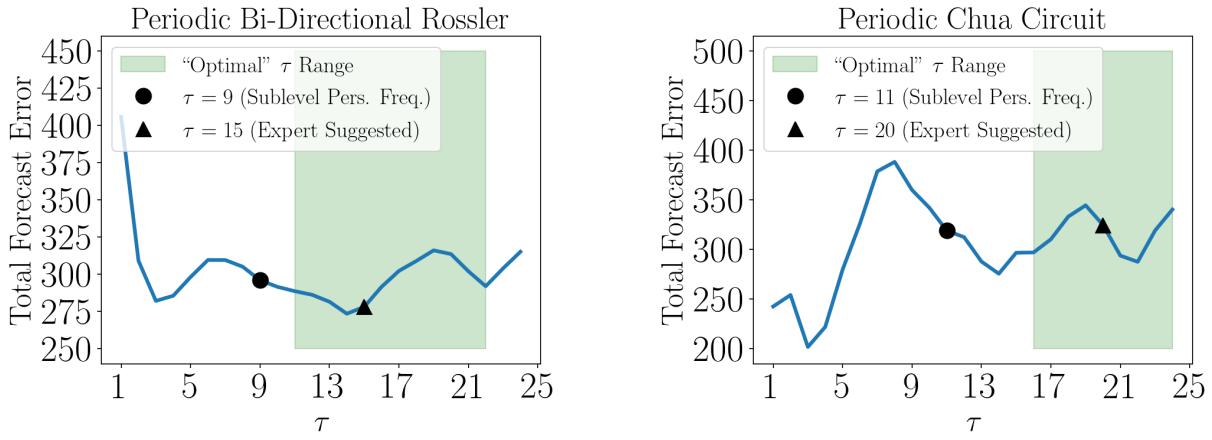
2.2.5.3 Periodic Bi-Directional Rossler System

Using the sublevel persistence method in the frequency domain, the delay for the periodic bi-directional Rossler system was computed to be $\tau = 9$ while the expert suggested delay is $\tau = 15$. This system was simulated using the specifications in Sec. 2.2.7.4 and the AR model was used to forecast the embedded system states between $t = 930$ to $t = 980$. The AR model was trained on simulation data at an initial condition of $x_1 = -0.4$, $y_1 = 0.6$, $z_1 = 5.8$, $x_2 = 0.8$, $y_2 = -2.0$, $z_2 = -4.0$, and the model was tested by changing x_1 to 1.0 and y_1 to 0.4. This process was performed for τ values between 1 and 25 to generate a RMSE error plot with respect to the system delay and a dimension of $n = 6$ was used for embedding. The forecast error for this system is shown in Fig. 2.31(a) where we see that the error for the two delays of interest are relatively close with the sublevel persistence method giving an approximately 10% larger forecast error compared to the expert suggested value. This implies that our method can provide embedding parameters that give accurate forecasting results for periodic signals. We also see that the forecast error for some small delays can be low so it is important to emphasize that this plot does not provide a measure of the optimal delay for attractor reconstruction.

2.2.5.4 Periodic Chua Circuit

The optimal delay using sublevel persistence with the frequency domain method for the periodic chua circuit was determined to be $\tau = 11$ while the expert suggested delay is $\tau = 20$. Autoregressive model forecasting was also applied to this system to demonstrate that our methods provide parameters that result in similar forecasting errors compared to expert suggested parameters. We trained a 300 term AR model on simulation data as in Sec. 2.2.7.5. The model was used to forecast the system states between $t = 138$ to $t = 180$ with training initial conditions of $x = 1.0$, $y = 0.0$, $z = 0.0$, and testing initial conditions were set by changing $y = 1.0$ and $z = 1.0$. A dimension of $n = 7$ was used for all simulations of this system and the delay was varied between 1 and 25. The

RMSE errors for this system are shown in Fig. 2.31(b) where we see that for this specific initial condition, the total error for the expert suggested parameter forecast is nearly identical to the result from our method, and because both errors are close this gives further evidence that our method is robust to forecasting.



(a) (b)

Figure 2.31 RMSE forecast error plot for the bi-directional periodic rossler system (a) and periodic chua circuit (b) with respect to the embedding delay using an AR forecast model with 300 terms to estimate the embedded system states. The emphasized values and ranges in the plot are from Table 2.3.

2.2.5.5 Robustness to Additive Noise

To determine the noise robustness of the delay parameter selection methods investigated in this work we will use an example time series. Specifically, we will use the x solution to the periodic Rossler system as described in Section 2.2.7.2. Note that the simulation parameters for these results differ from those in [**?**]. We will use additive Gasussian noise $\mathcal{N}(\mu = 0, \sigma^2)$, where σ is determined from the Signal-to-Noise Ratio (SNR). The SNR is a measurement of how much noise there is in the signal with units of decibels (dB) and is calculated as

$$\text{SNR}_{\text{dB}} = 20\log_{10}\left(\frac{A_{\text{signal}}}{A_{\text{noise}}}\right), \tag{2.22}$$

where A_{signal} and A_{noise} are the Root-Mean-Square (RMS) amplitudes of the signal and additive noise, respectively. If we manipulate Eq. (2.22) we can solve for A_{noise} as

$$A_{\text{noise}} = A_{\text{signal}} 10^{-\frac{\text{SNR}_{\text{dB}}}{20}}. \tag{2.23}$$

Because $x(t)$ is a discrete sampling from a continous system with $t = [t_1, t_2, \ldots, t_N]$, we calculate A_{signal} as

$$A_{\text{signal}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[x(t_i) - \bar{x}]^2}, \tag{2.24}$$

where $\bar{x}$ is the mean of x and is subtracted from $x(t)$ to center the signal about zero. with A_{noise} calculated, we set the additive noise standard deviation as $\sigma = A_{\text{noise}}$.

We applied a sweep of the SNR from 1 to 40 in increments of 1 with each SNR being repeated for 30 unique realizations of the noise distributed as $\varepsilon \sim \mathcal{N}(0, A^2_{\text{noise}})$. For each realization of $x(t) + \varepsilon$ the delay parameters were calculated using all 3 methods: sublevel set persistence of the frequency domain $\tau_{\text{SL}_\text{f}}$, sublevel set persistence of the time domain $\tau_{\text{SL}_\text{t}}$, and mutual information τ_{MI}. The mean and standard deviation of the 30 trials at each SNR were calculated for each method as shown in Fig. 2.32. In Fig. 2.32, it is clear that the methods are robust to noise down to a SNR of

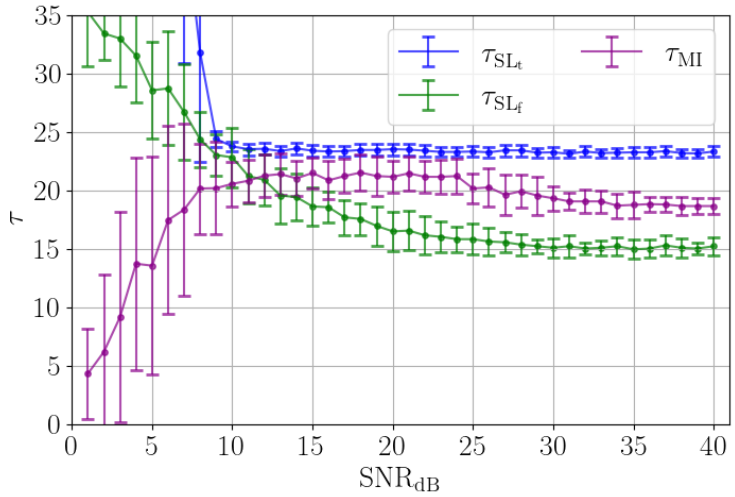


Figure 2.32 Noise robustness analysis of the delay parameter selection using the Rossler system with incrementing additive noise. The mean and standard deviation as error bars of the delay parameters from 30 trials at each SNR were calculated using sublevel set persistence of the frequency domain $\tau_{\text{SL}_\text{f}}$, sublevel set persistence of the time domain $\tau_{\text{SL}_\text{t}}$, and mutual information τ_{MI}.

approximately 10 dB. While this does show a limit for the sublevel set persistence methods, SNR values below 10 dB indicate extremely noisy signals.

2.2.5.6 Robustness to Signal Length

A common issue with signal processing and time series analysis methods is their limited functionality with smaller sets of data available, which has been used to analyze the sensitivity of the delay parameter selection [**?**]. Here we will investigate the limitations of these methods in the face of short time series. We will do this analysis by incrementing the length of the time series with the PE parameters calculated at each increment. For our analysis we will again use the Rossler system as described in Section 2.2.7.2. Specifically, we increment the length of the signal from $L = 75$ to 1000 in steps of 25 (see Fig. 2.33). However, if this type of analysis is not available for the data set being analyzed, for time series analysis applications it is commonly suggested to have a data length of $L = 4000$ for continuous dynamical systems and and $L = 500$ for maps [**?**].

In Fig. 2.33 we see that all of the methods reach a larger value of τ, in comparison to the expert suggested $\tau = 9$ when the time series contains at least 300 data points. However, we see that the time domain method seems to be more robust to signal length compared to mutual information

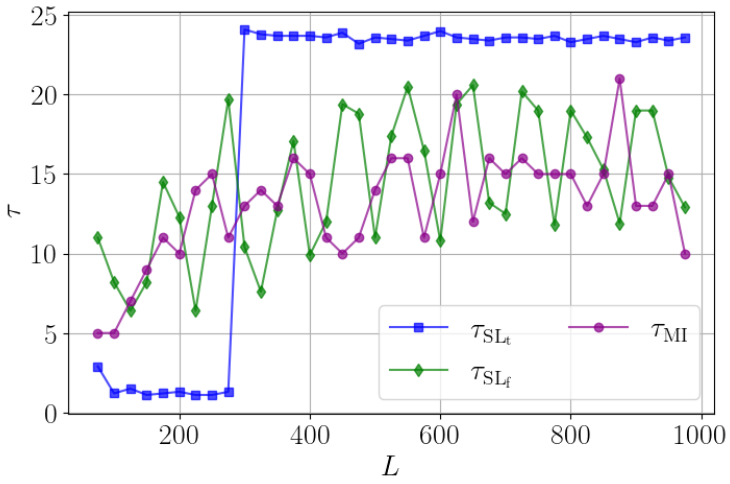


Figure 2.33 Signal length robustness analysis of the delay parameter selection using the Rossler system with incrementing signal length from 75 to 1000 in steps of 25. The delay parameters were calculated at each L using set persistence of the frequency domain τ_{SL_f}, sublevel set persistence of the time domain τ_{SL_t} and mutual information τ_{MI}.

and the frequency domain method. An important note to make is that this result is not general for all continuous dynamical systems. The required length of the signal is going to vary significantly depending on the sampling rate of the time series. To determine a general requirement for the methods we repeated this analysis method for all of the systems shown in Table 2.3. Our result from this analysis found that, in general, a signal of $L \geq 15\tau$ allows selecting an appropriate PE and state space reconstruction delay τ using the TDA-based methods described in this manuscript.

2.2.6 Conclusion

We described a novel TDA-based approach for automatically determining the PE delay parameter τ given a sufficiently sampled/over-sampled time series. We investigated the sublevel set persistence in both the time and frequency domains to determine the maximum significant frequency, which was then used to estimate an appropriate delay based on the Shannon-Nyquist sampling criteria [**?**].

In regards to the permutation dimension n, in Section 2.2.4 we developed a simple statistical analysis method for selecting an appropriate permutation dimension n based on the need for only a portion of the permutations to be used in the time series to capture complexity changes. This method also revealed that the permutation dimension and Takens' embedding dimension are not necessarily related and tools for the Takens' embedding dimension cannot generally be used for the permutation dimension.

To determine the accuracy of these methods, the resulting delays were compared to each of the standard MI approach of Fraser and Swinney [71], expert suggested parameters of various categories of dynamical systems and data sets, and optimal parameters calculated if the dynamical system model allowed for various differing dynamical states to be simulated. This result showed that the sublevel set persistence of the time domain method provided the most accurate delays for all of the systems analyzed except for EEG data. However, due to the differing sampling rates

for different EEG data sets, the expert suggested delay of 1–3 may not be accurate for the EEG data investigated in this work. We do not recommend using the sublevel set persistence of the frequency domain as this method consistently provided delays that were too small for continuous flow differential equations and too large for periodic functions. Our noise robustness analysis revealed that the methods we developed here were robust to additive noise up to the SNR values as low as 10 dB. We also analyzed the robustness of the methods to signal length and found they provide an accurate delay even for short time series with the general suggestion of signal length $L > 15\tau$.

In comparison to the expert suggested dimensions and the optimal dimensions from comparing chaotic and periodic signals, our dimension parameter selection method accurately provided similar results for nearly all of the systems investigated in this work. Additionally, the range suggested using our method was more precise than the dimensions suggested by experts giving the user a more definite answer to an appropriate permutation dimension as these results can vary based on simulation parameters. We hypothesize that using techniques such as a weighted average to combine the parameters from the different methods based on knowledge of the system could lead to even more optimal results. However, this is a topic for future research.

2.2.7 Summary of the used data and models

2.2.7.1 Lorenz System

The Lorenz system used is defined as

$$\frac{dx}{dt} = \sigma(y-x),\ \frac{dy}{dt} = x(\rho - z) - y,\ \frac{dz}{dt} = xy - \beta z. \tag{2.25}$$

The Lorenz system had a sampling rate of 100 Hz. This system was solved for 100 seconds and the last 20 seconds were used. For a periodic response parameters $\sigma = 10.0$, $\beta = 8.0/3.0$, and $\rho = 100$ were used. For a chaotic response parameters $\sigma = 10.0$, $\beta = 8.0/3.0$, and $\rho = 105$ were used.

2.2.7.2 Rössler System

The Rössler system used was defined as

$$\frac{dx}{dt} = -y - z,\ \frac{dy}{dt} = x + ay,\ \frac{dz}{dt} = b + z(x-c), \tag{2.26}$$

with parameters of $a = 0.10$, $b = 0.20$ for periodic and $b = 0.1$ for chaotic, and $c = 14$, which was solved for 1000 seconds with a sampling rate of 15 Hz. Only the last 2500 data points of the solution were used.

2.2.7.3 Coupled Rössler-Lorenz System

The coupled Lorenz-Rössler system is defined as

$$\begin{aligned}
\frac{dx_1}{dt} &= -y_1 - z_1 + k_1(x_2 - x_1), \\
\frac{dy_1}{dt} &= x_1 + ay_1 + k_2(y_2 - y_1), \\
\frac{dz_1}{dt} &= b_2 + z_1(x_1 - c_2) + k_3(z_2 - z_1), \\
\frac{dx_2}{dt} &= \sigma(y_2 - x_2), \\
\frac{dy_2}{dt} &= \lambda x_2 - y_2 - x_2 z_2, \\
\frac{dz_2}{dt} &= x_2 y_2 - b_1 z_2,
\end{aligned} \tag{2.27}$$

where $b_1 = 8/3$, $b_2 = 0.2$, $c_2 = 5.7$, $k_1 = 0.1$, $k_2 = 0.1$, $k_3 = 0.1$, $\lambda = 28$, $\sigma = 10$, and $a = 0.25$ for a periodic response and $a = 0.51$ for a chaotic response. This system was simulated at a frequency of 50 Hz for 500 seconds with the last 30 seconds used.

2.2.7.4 Bi-Directional Coupled Rössler System

The Bi-directional Rössler system is defined as

$$\begin{aligned}
\frac{dx_1}{dt} &= -w_1 y_1 - z_1 + k(x_2 - x_1), \\
\frac{dy_1}{dt} &= w_1 x_1 + 0.165 y_1, \\
\frac{dz_1}{dt} &= 0.2 + z_1(x_1 - 10), \\
\frac{dx_2}{dt} &= -w_2 y_2 - z2 + k(x_1 - x_2), \\
\frac{dy_2}{dt} &= w_2 x_2 + 0.165 y_2, \\
\frac{dz_2}{dt} &= 0.2 + z_2(x_2 - 10),
\end{aligned} \tag{2.28}$$

with $w_1 = 0.99$, $w_2 = 0.95$, $k = 0.25$ for periodic and $k = 0.3$ for chaotic. This was solved for 1000 seconds with a sampling rate of 10 Hz. Only the last 150 seconds of the solution were used.

2.2.7.5 Chua Circuit

Chua's circuit is based on a non-linear circuit and is described as

$$\begin{aligned}
\frac{dx}{dt} &= \alpha(y - f(x)), \\
\frac{dy}{dt} &= \gamma(x - y + z), \\
\frac{dz}{dt} &= -\beta y,
\end{aligned} \tag{2.29}$$

where $f(x)$ is based on a non-linear resistor model defined as

$$f(x) = m_1 x + \frac{1}{2}(m_0 + m_1)\left[|x+1| - |x-1|\right]. \tag{2.30}$$

The system parmeters were set to $\beta = 27$, $\gamma = 1$, $m_0 = -3/7$, $m_1 = 3/7$, and $\alpha = 10.8$ for a periodic response and $\alpha = 12.8$ for a chaotic response. The system was simulated for 200 seconds at a rate of 50 Hz and the last 80 seconds were used.

2.2.7.6 Mackey-Glass Delayed Differential Equation

The Mackey-Glass Delayed Differential Equation is defined as

$$x(t) = -\gamma x(t) + \beta \frac{x(t-\tau)}{1 + x(t-\tau)^n} \tag{2.31}$$

with $\tau = 2$, $\beta = 2$, $\gamma = 1$, $n = 7.75$ for periodic and $n = 9.65$ for chaotic. This was solved for 400 seconds with a sampling rate of 50 Hz. The solution was then downsampled to 5 Hz and the last 200 seconds were used.

2.2.7.7 Periodic Sinusoidal Function

The sinusoidal function is defined as

$$x(t) = \sin(2\pi t) \tag{2.32}$$

This was solved for 40 seconds with a sampling rate of 50 Hz.

2.2.7.8 Quasiperiodic Function

This function is generated using two incommensurate periodic functions as

$$x(t) = \sin(\pi t) + \sin(t). \tag{2.33}$$

This was sampled such that $t \in [0, 100]$ at a rate of 50 Hz.

2.2.7.9 EEG Data

The EEG signal was taken from andrzejak et al. [**?**]. Specifically, the first 5000 data points from the EEG data of a healthy patient from set A (file Z-093) was used and the first 5000 data points of a patient experiencing a seizure from set E (file S-056) was used.

2.2.7.10 ECG Data

The Electrocardoagram (ECG) data was taken from SciPy's misc.electrocardiogram data set. This ECG data was originally provided by the MIT-BIH Arrhythmia Database [**?**]. We used data points 3000 to 5500 during normal sinus rhythm and 8500 to 11000 during arrhythmia.

2.2.7.11 Logistic Map

The logistic map was generated as

$$x_{n+1} = r x_n (1 - x_n), \tag{2.34}$$

with $x_0 = 0.5$ and $r = 3.95$. Equation 2.34 was solved for the first 500 data points.

2.2.7.12 Hénon Map

The Hénon map was solved as

$$\begin{aligned} x_{n+1} &= 1 - ax_n^2 + y_n, \\ y_{n+1} &= bx_n, \end{aligned} \tag{2.35}$$

where $b = 0.3$, $x_0 = 0.1$, $y_0 = 0.3$, and $a = 1.4$. This system was solved for the first 500 data points of the x-solution.

2.2.7.13 Double Pendulum

The double pendulum is a staple benchtop experiment for investigated chaos in a mechanical system. A point-mass double pendulum's equations of motion are defined as

$$\begin{aligned} \frac{d\theta_1}{dt} &= \omega_1, \\ \frac{d\theta_2}{dt} &= \omega_2, \\ \frac{d\omega_1}{dt} &= \frac{N_1}{\ell_1(2m_1 + m_2 - m_2\cos(2\theta_1 - 2\theta_2))}, \\ \frac{d\omega_2}{dt} &= \frac{N_2}{\ell_2(2m_1 + m_2 - m_2\cos(2\theta_1 - 2\theta_2))}, \\ N_1 &= -g(2m_1 + m_2)\sin(\theta_1) - m_2 h\sin(\theta_1 - 2\theta_2) \\ &\quad - 2\sin(\theta_1 - \theta_2)m_2(\omega_2^2\ell_2 + \omega_1^2\ell_1\cos(\theta_1 - \theta_2)), \\ N_2 &= 2\sin(\theta_1 - \theta_2)(\omega_1^2\ell_1(m_1 + m_2) \\ &\quad + g(m_2 + m_2)\cos(\theta_1) + \omega_2^2\ell_2 m_2\cos(\theta_1 - \theta_2)), \end{aligned} \tag{2.36}$$

where the system parameters $g = 9.81$ m/s^2, $m_1 = 1$ kg, $m_2 = 1$ kg, $\ell_1 = 1$ m, and $\ell_2 = 1$ m. The system was solved for 200 seconds at a rate of 100 Hz and only the last 30 seconds were used with initial conditions $[\theta_1, \theta_2, \omega_1, \omega_2] = [0.4\,\text{rad}, 0.6\,\text{rad}, 1, 1]$ for periodic and $[\theta_1, \theta_2, \omega_1, \omega_2] = [0, 3\,\text{rad}, 0, 0]$ for chaotic. This system will have different dynamic states based on the initial conditions, which can vary from periodic, quasiperiodic, and chaotic.

CHAPTER 3

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 from [**?**] are presented in Section 3.1 and the pattern shapes are characterized from [**?**] in Section 3.2.

3.1 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.

3.1.1 Introduction

Extracting information from surface images is an important field of research with many applications such as medical imaging [**?**], remote sensing [**?**, **?**], 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' geom-

etry 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 [**?**] as well as fatigue resistance [**?**,**?**,**?**]. In additive manufacturing, surface texture is further used to understand and optimize the process [**?**, **?**, **?**].

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 [**?**]. For example, it has been shown that surface textures can be analyzed to identify the occurrence of chatter in a machining process [**?**, **?**, **?**, **?**, **?**, **?**]. Surface texture analysis has also been used to monitor and indicate tool wear in a machining process [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**], detect surface defects such as cracks and scratches [**?**, **?**, **?**, **?**, **?**], and for quantifying surface roughness of a part [**?**,**?**,**?**]. 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 [**?**, **?**, **?**, **?**] and can increase the ultimate and yield strengths [**?**, **?**]. 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 [**?**]. 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 [**?**]. Models have also been developed to describe the relationships between the system parameters and the resulting textures for this process [**?**]. Another example of a process that exploits surface texture for improving mechanical properties is piezo vibration striking treatment (PVST) [**?**], see Section 3.1.1.2. 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 analyzing these images ranging from statistical techniques to wavelet transform approaches [**?**, **?**, **?**]. The classical approaches can be grouped into four categories that are summarized in Fig. 3.1. For statistical methods, the gray level co-occurrence matrix (GLCM) is usually of interest in which a matrix is obtained containing information on the probabilities that adjacent pixels would have the same intensity [**?**]. Statistical measures are then computed on this matrix leading to quantification of broad features such as *smoothness*, *coarseness* and *regularity* of a texture [**?**, **?**].

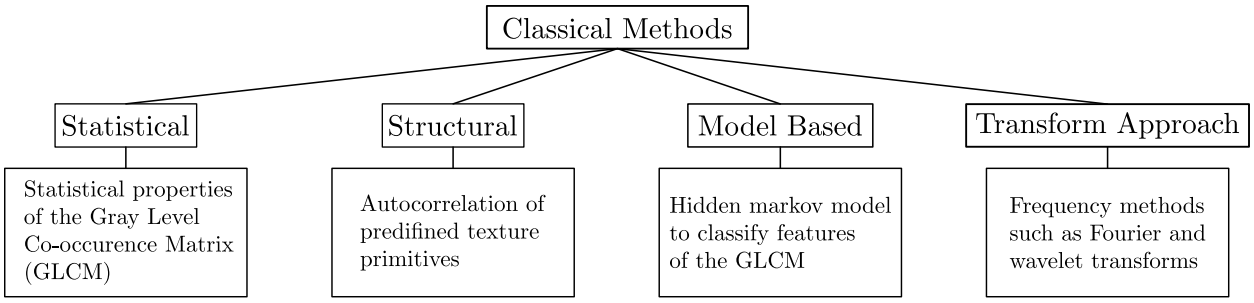


Figure 3.1 Block diagram summarizing the classical texture analysis methods and their basic descriptions.

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 [**?**]. Statistical quantities such as the image autocorrelation function provide information about the sizing of the primitives and a quantification of the texture periodicity [**?**]. 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 [**?**, **?**].

The final two methods are model based approaches and transform approaches. Model based methods utilize statistical models such as a Hidden Markov Model [**?**] 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 [**?**]. However, with transform methods, relative positioning of the texture features is lost in the process and further analysis is required to obtain this information [**?**]. 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.

3.1.1.1 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 3.2 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 [**?**] to classify surface patterns formed by the indentation centers in PVST processes [**?**] (second box in Fig. 3.2), quantifying the consistency of indentation depths (third box in Fig. 3.2), and characterizing generalized radii of indentation shapes, e.g., the profile of the indenter at different heights (last box in Fig. 3.2) 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 [**?**]. 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 [**?**], provide a quantitative approach for characterizing surfaces from texture-producing processes such as PVST.

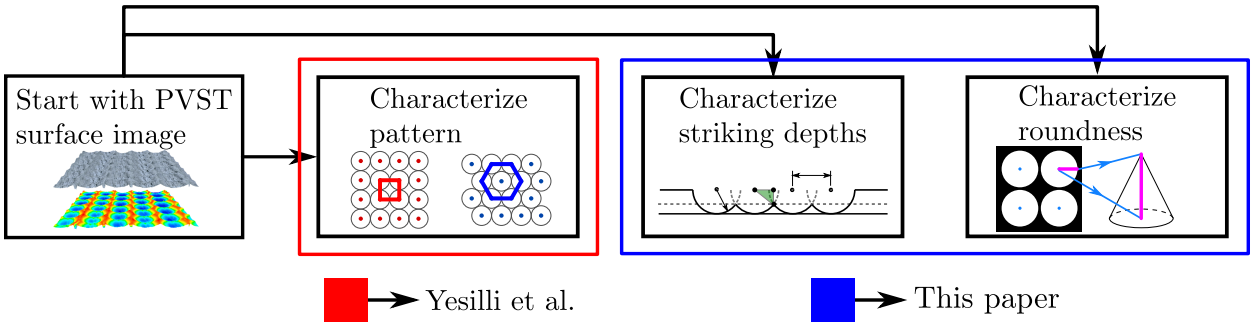


Figure 3.2 An overview flow chart for PVST texture characterization. Starting with a PVST image, three main features can be classified (depth, roundness, and pattern).

We start by describing the PVST process in 3.1.1.2. We then introduce the relevant TDA background followed by derivations for the theoretical expressions used for quantifying texture features in Section 3.1.2. The results are presented in Section 3.1.3, and the concluding remarks are listed in Section 3.1.5. Finally, CAD simulation of PVST patterns, a feature score noise analysis, and surface slope and angularity estimation are included in the appendices.

3.1.1.2 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. 3.3 demonstrates how the PVST process generates a texture on the surface as a result of the process parameters. We see that

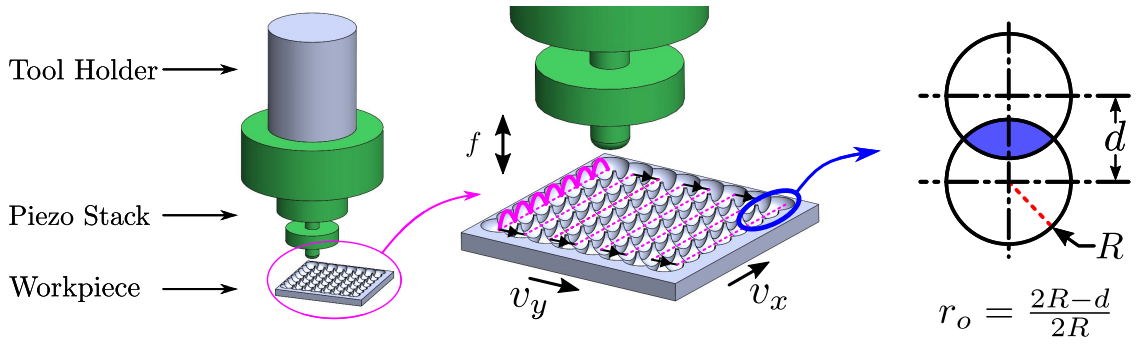


Figure 3.3 PVST diagram showing the mechanics of the PVST process and how the texture can be controlled using the frequency f, the in plane scanning speeds v_x and v_y, and the overlap ratio r_o.

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.

3.1.2 Background and Theory

Section 3.1.2.1 provides a brief background on persistent homology, the main tool from TDA that we use in this work. Sections 3.1.2.2 and 3.1.2.3 show the derivations for the expressions that will be used to score the strike depths and roundness, respectively, of the PVST surfaces. Section 3.1.2.3 shows how the process knowledge was applied to locate the strike minima and obtain the surface reference height. Section 3.1.2.4 describes how our approach can be generalized to other tool shapes.

3.1.2.1 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 [**?**] 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 [**?**]. 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. 3.4 (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. 3.4 (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. 3.4 (e-g) shows three different level sets for the full surface with corresponding binary images in Figs. 3.4 (a-c). Starting with Fig. 3.4 (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. 3.4 (h). Figure 3.4 (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. 3.4 (b) at $T = 0.6$. A second loop is born at 0.602 shown in Fig. 3.4 (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. 3.4 (h). As T reaches its highest value at 1 (Fig. 3.4 (h)), the full persistence diagram is obtained. The loop on the right side is visually larger than the left one in Fig. 3.4 (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 [72]. 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 [72] states that for two continuous well-behaved functions, f and g, the bottleneck distance satisfies,

$$d_B(D(f), D(g)) \leq ||f - g||_\infty, \tag{3.1}$$

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 (3.1) 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.

3.1.2.2 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. 3.5. 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

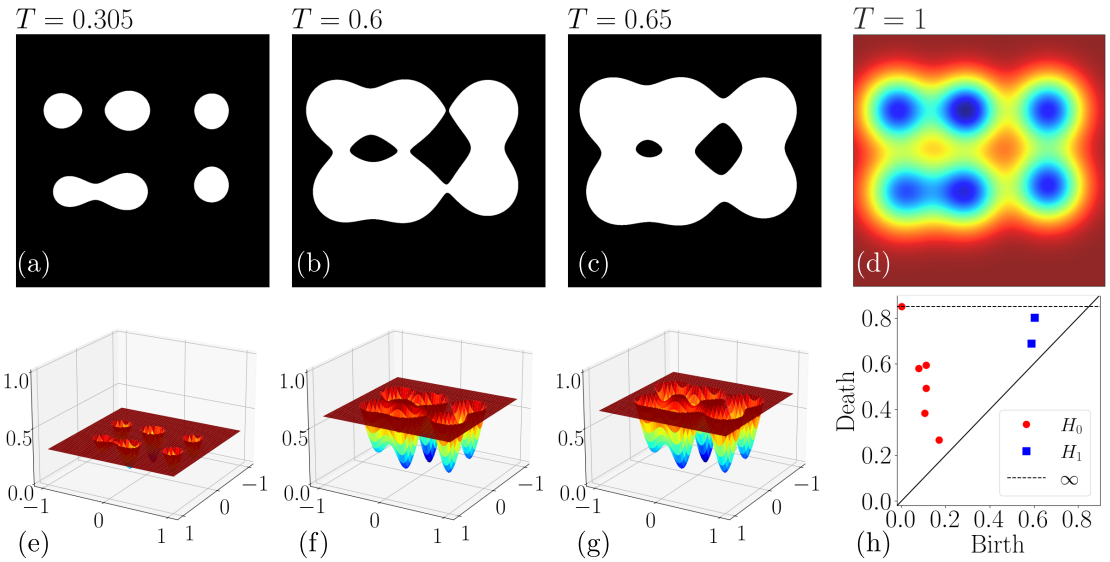
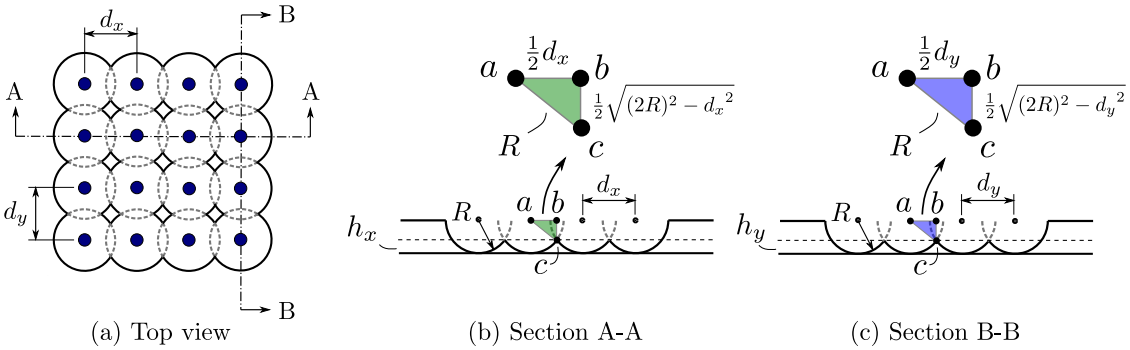


Figure 3.4 Sublevel persistent homology example. (a-c) shows the surface level set represented as a binary image at 3 threshold heights, (d) shows the full surface image (e-g) shows the corresponding 3D surface plots for the thresholded images and (h) shows the full sublevel set persistence diagram for the surface.



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

Figure 3.5 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,

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 ratio is defined by

$$r_x = \frac{2R - d_x}{2R}, \tag{3.2}$$

where r_x is the overlap ratio in the x direction. Using the geometric expressions in Fig. 3.5, h_x measured from the maximum depth of the impact can be computed using

$$h_x = \frac{1}{2}\left(2R - \sqrt{(2R)^2 - {d_x}^2}\right), \tag{3.3}$$

where h_x is the height at which all of the impact **rows** merge. Combining Eq. (3.2) and Eq. (3.3) to eliminate d_x gives an expression for the height h_x in terms of the impact radius and the overlap ratio

$$h_x = R\left(1 - \sqrt{(2 - r_x)r_x}\right). \tag{3.4}$$

Similar expressions can be obtained for the vertical direction by replacing x with y. In order to normalize Eq. (3.4), 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. (3.4) 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

$$\overline{h} = 1 - \sqrt{(2 - r)r}, \tag{3.5}$$

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. (3.5) 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. 3.6, 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

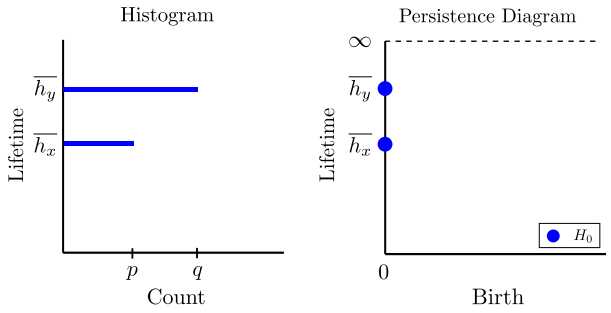


Figure 3.6 Theoretical sublevel persistence diagram and histogram for the striking depths for an arbitrary $p \times q$ grid with critical heights h_x and h_y.

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 3.1 shows the expected lifetime of the PVST strikes for different overlap ratios using Eq. (3.5). See Section 3.1.6 for CAD-based simulations used to confirm the theoretical derivations.

Table 3.1 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

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 counterparts. 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 [**?**]. Once the two distributions are obtained, we compute the Earth Movers Distance between them to quantify the differences between the distributions [**?**]. 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

$$\mathrm{EMD}(u,v) = \inf_{\pi \in \Gamma(u,v)} \mathbb{E}_{(x,y)\sim\pi}[|x-y|], \tag{3.6}$$

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 [**?**].

It should be noted that Eq. (3.6) 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. (3.6)

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

$$\bar{D} = 1 - \text{EMD}, \tag{3.7}$$

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.

3.1.2.3 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

$$x = \frac{n_p w}{P}, \tag{3.8}$$

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

$$R = \frac{v_x}{120 f}. \tag{3.9}$$

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

$$v_x = 3000(1 - r), \tag{3.10}$$

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 (3.9), we obtain an expression for the nominal circle radius in terms of the overlap ratio,

$$R = \frac{1}{4}(1-r), \tag{3.11}$$

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. 3.7 (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

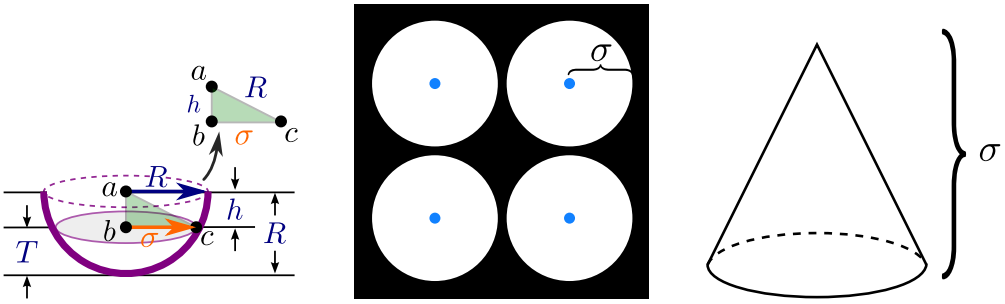
$$h^2 + \sigma^2 = R^2. \tag{3.12}$$

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

$$\sigma = \sqrt{(2R-T)T}, \tag{3.13}$$

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 ($T < \bar{h}$): Consider the PVST grid with no overlap shown in Fig. 3.7 (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. 3.7 (c)). As the distance from the edge of the circle increases, so does the height of the cones which can be understood from Fig. 3.7 (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



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

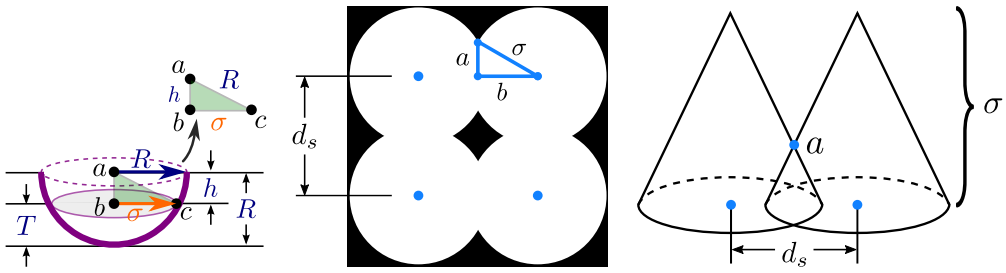
Figure 3.7 Converting between a binarized image and its corresponding distance transform geometry. (a) shows the strike geometry used for converting threshold heights to the radius of the strike at that height, (b) shows the thresholded image at a height below the critical height $T < \bar{h}$ (no overlap), (c) shows the cone geometry resulting from the distance transform of the binarized image and how the strike radius σ appears in each form.

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.

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. 3.8 (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,

$$\bar{h} = R(1 - \sqrt{(2-r)r}), \tag{3.14}$$

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 case from Section 3.1.2.3 is used, whereas if $\varepsilon \geq 1$, the circles will overlap and a more general relationship needs to be considered. In Fig. 3.8 (a) we show a binarized image where $\varepsilon > 1$ with strike overlap. When this thresholded image is distance transformed, a result similar to Fig. 3.8 (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



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

Figure 3.8 Converting between a binarized image and its corresponding distance transform geometry. (a) shows the thresholded image at a height below the critical height $T \geq \bar{h}$ **(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.

obtain an expression for a, we consider the triangle shown in Fig. 3.8 (b) and apply the Pythagorean theorem

$$a = \sqrt{\sigma^2 - b^2}. \tag{3.15}$$

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

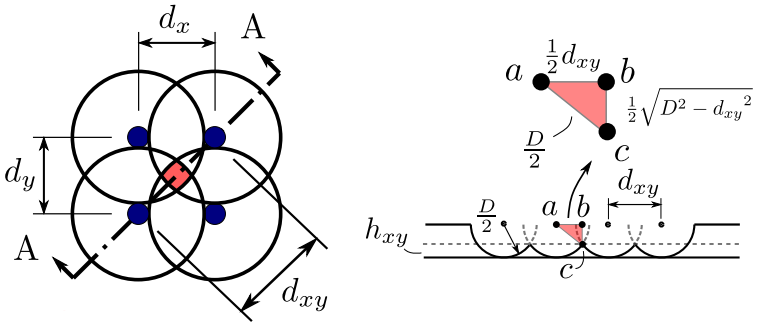
definition of the overlap ratio. At this point it is important to note that this expression depends on the full nominal radius R and should not be written in terms of σ because d_s remains invariant for all threshold heights. Observe that, $d_s = 2b$ from Fig. 3.8 (b) due to the circle position remaining constant. Applying the definition of d_s to the result for b we obtain an expression for b in terms of known parameters

$$b = R(1-r). \tag{3.16}$$

Substituting b into Eq. (3.15) gives the critical distance

$$a = \sqrt{\sigma^2 - R^2(1-r)^2}. \tag{3.17}$$

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. 3.9. We see that



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

Figure 3.9 Nominal PVST grid with a high overlap ratio to demonstrate diagonal cross section overlap height.

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 can be computed using

$$d_{xy} = \sqrt{d_x^2 + d_y^2}, \tag{3.18}$$

or if the grid is square

$$d_{xy} = 2\sqrt{2}R(1-r), \tag{3.19}$$

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},

$$h_{xy} = R(1 - \sqrt{1 - 2(1-r)^2}). \tag{3.20}$$

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,

$$d_{xy} > 2R. \tag{3.21}$$

Here, if we assume equality, and substitute Eq. (3.19), 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:

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 3.1.6 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. 3.10 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. 3.11 (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

(a) Binarized Image (b) Distance Transform

Figure 3.10 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.

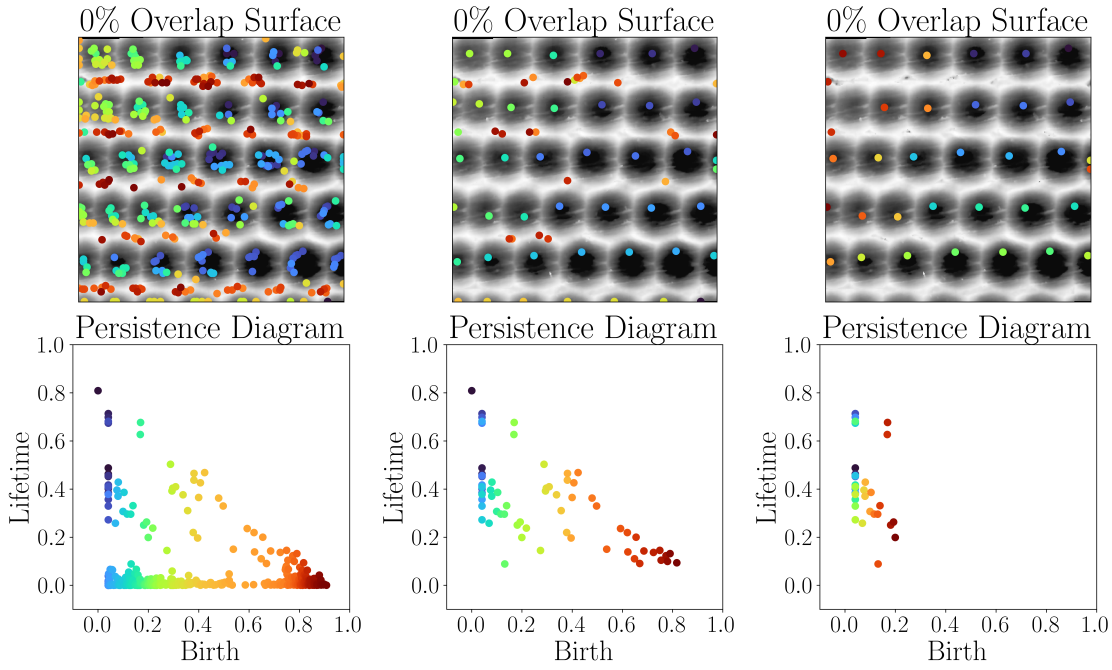
persistence pairs by computing a histogram of the lifetimes, and thresholding the lifetime above any point that contained a bar height larger than the number of desired features. This threshold resulted in the persistence diagram shown in Fig. 3.11 (b). It is clear that the features removed up to this 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. 3.11 (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. 3.12. 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 3.1.8.

Roundness Score: To quantify the feature roundness, the images needed to be thresholded at 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. (3.22),

$$\bar{R}_G = \frac{1}{1-h_r} \int_0^1 \mathrm{EMD}(T)\, \mathrm{d}T, \tag{3.22}$$

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

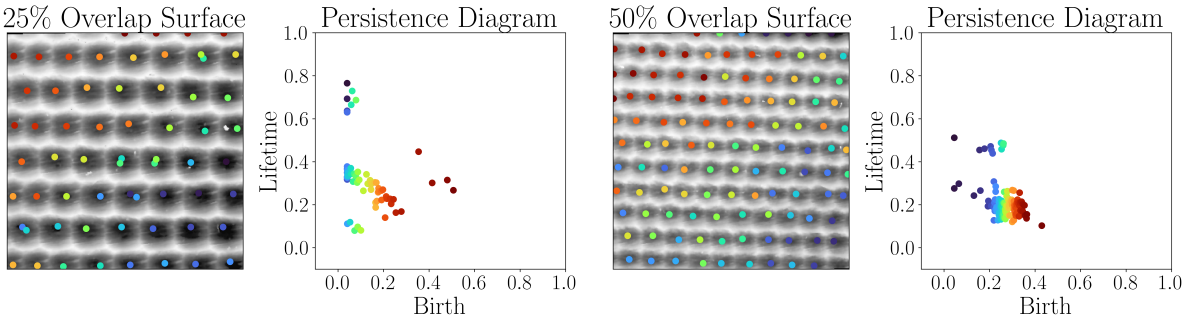


(a) Original Simplified Surface Persistence Diagram.

(b) Low Lifetime Filtered Persistence Diagram.

(c) Final Filtered Persistence Diagram.

Figure 3.11 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) 25% Overlap Filtered Persistence Diagram. (b) 50% Overlap Filtered Persistence Diagram.

Figure 3.12 Persistence diagram (PD) filtering on the simplified surface to locate strike minima. (a) 25% Overlap filtered PD, (b) 50% overlap filtered PD.

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

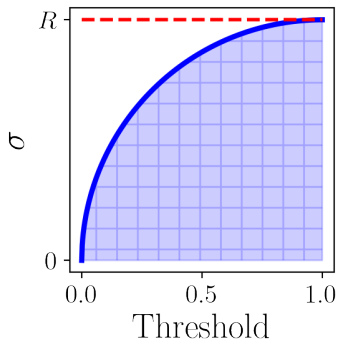


Figure 3.13 Plot of σ as a function of the image threshold height to demonstrate the worst case earth movers distance plot.

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. (3.13) 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. 3.13.

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

$$\bar{R} = 1 - \frac{4\bar{R}_G}{\pi R} \tag{3.23}$$

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 3.1.7 for a quantification of the noise robustness of the depth and roundness scores.

3.1.2.4 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 [**?**]. 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 complexities in the analysis, but we provide two potential avenues for doing this. The first method offered is to apply the techniques in Appendix 3.1.6 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 [**?**] described by

$$\rho(x,y) = \sum_{i=1}^{n} \alpha_i ||\vec{x} - \vec{b}_i||_p, \tag{3.24}$$

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. 3.14.

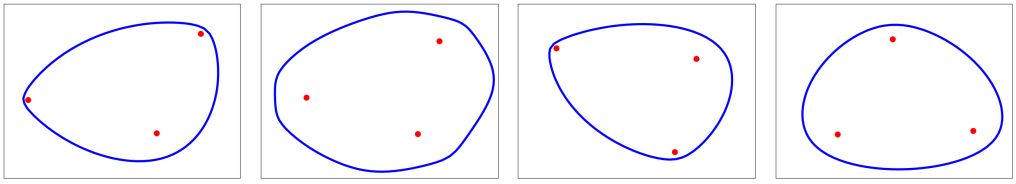


Figure 3.14 Example plots of generalized conic sections to demonstrate different tool shape configurations for generalizing the results in this paper. Red points are the focal points of the conic, and the blue curve represents the cross section of the impact tool.

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

3.1.3.1 Strike Depth Results

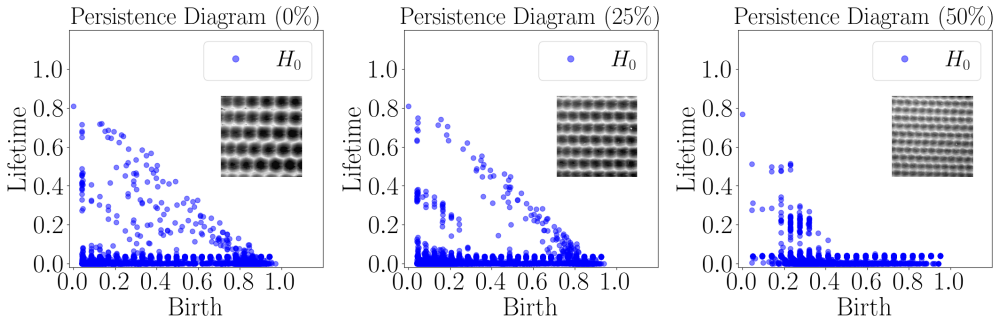
Sublevel set persistence was applied to the PVST images shown in Fig. 3.15 with the corresponding persistence diagrams adjacent to each image. We see a significant portion of the persistence pairs have negligible lifetime and are likely a result of noise in the images. The noise was removed from these 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

Table 3.2 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%

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. 3.11.

Applying these operations to the diagrams in Fig. 3.15, the filtered persistence diagrams in Fig. 3.16 were obtained. The corresponding computed depth scores are shown in Table 3.2, 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.



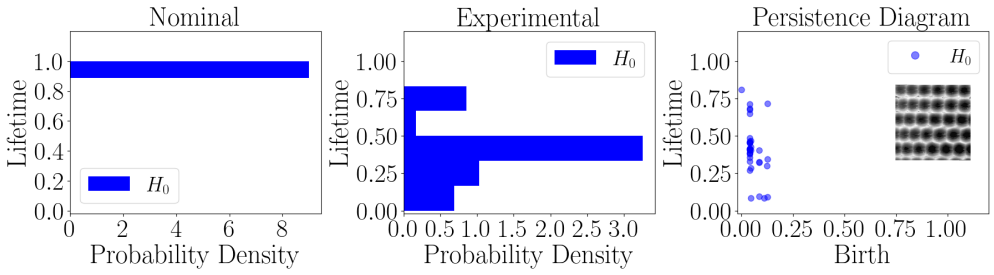
(a) 0% Overlap (b) 25% Overlap (c) 50% Overlap

Figure 3.15 Unaltered striking **depth** persistence diagrams (a) 0% Overlap PVST Image, (b) 25% Overlap PVST Image, (c) 50% Overlap PVST Image

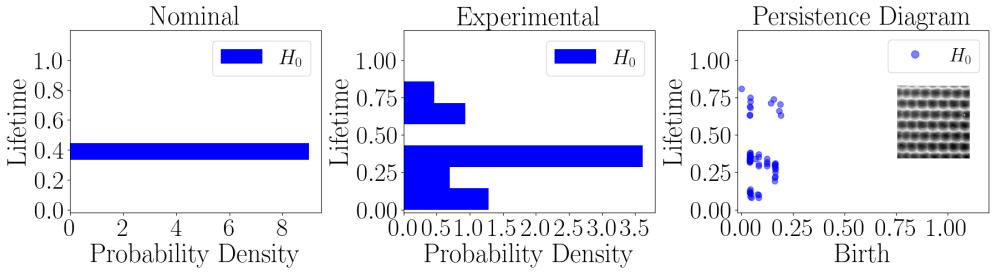
3.1.3.2 Strike Roundness Results

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 3.17 shows the thresholded and distance transformed images at various heights as an example.

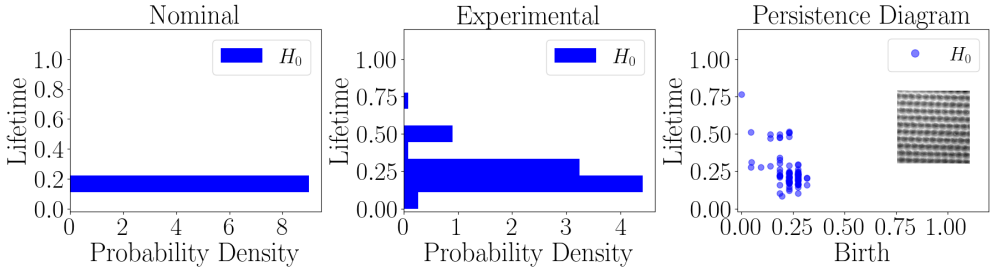
The persistence lifetime histograms were used to compute the earth movers distance between the nominal and experimental distributions which provided a score at each height in the image and therefore information about the roundness over the entire depth of the strikes. Thresholding was started at the reference point ($T = 0$) found from the filtered persistence diagrams. Histograms such as the ones in Fig. 3.18 were generated at each height to visually compare the theoretical distribution of persistence lifetimes to the experimental distribution. This process resulted in an earth movers distance distribution with respect to threshold height as shown in Fig. 3.19. 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. 3.19. 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



(a) 0% Overlap



(b) 25% Overlap



(c) 50% Overlap

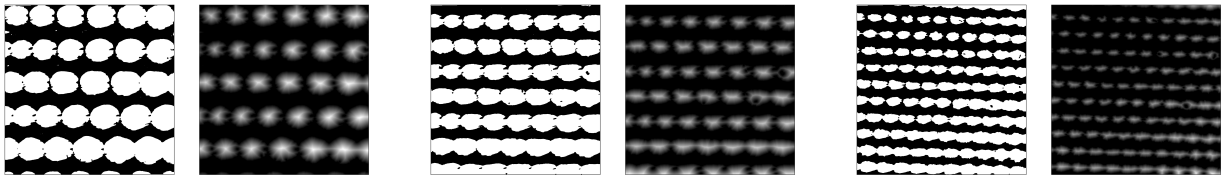
Figure 3.16 Noise filtered **depth** persistence diagrams and histograms for each overlap ratio.

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. (3.23) because the strikes were nominally spherical. The computed scores are shown in Table 3.3 where a higher score corresponds to the roundness distribution being closer to nominal.

Table 3.3 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.



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

Figure 3.17 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.

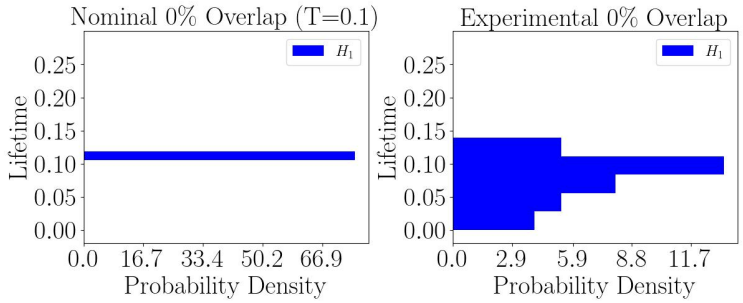


Figure 3.18 Roundness lifetime histogram example at a threshold height of 0.1 from the 0% overlap image.

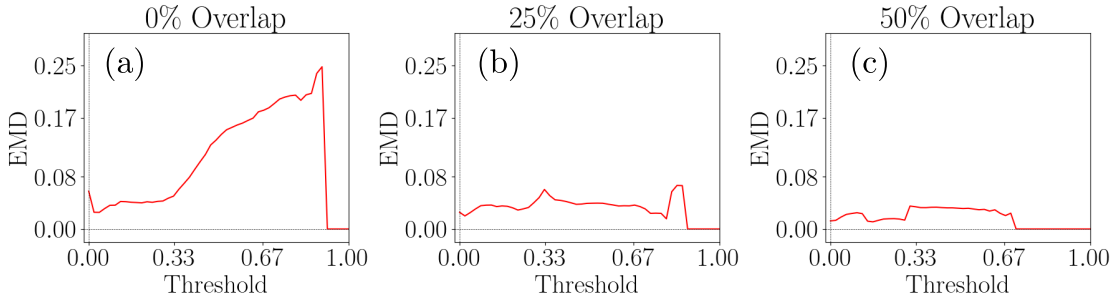


Figure 3.19 Earth movers distance between the experimental and theoretical roundness lifetime distributions as a function of threshold height for each overlap ratio.

3.1.4 Analysis Software

The software used for this analysis was implemented as a texture analysis module in the *teaspoon* python library for topological signal processing [52]. 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 [**?**].

To generate the nominal images, the process presented in Section 3.1.6 was followed using SolidWorks to model the surfaces and the algorithm in [**?**] to convert the CAD model to a point

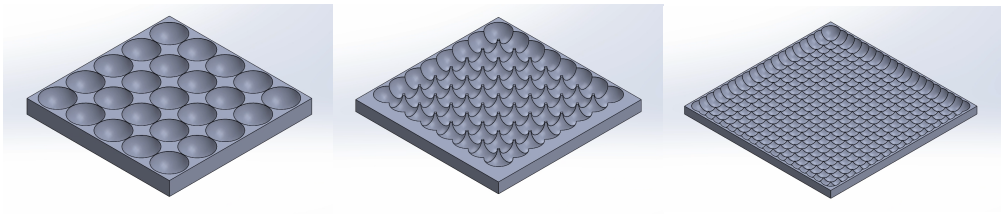
cloud that could be converted to an image.

3.1.5 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 treatment 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.

3.1.6 Appendix — Verifying Theoretical Results

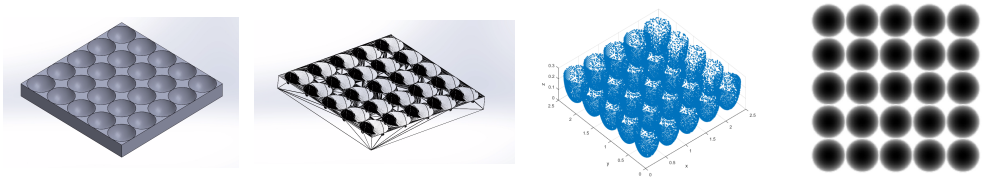
In order to verify the expressions in Section 3.1.2.2, we manufactured gray scale images consisting of perfect PVST strikes in the expected patterns, and computed sublevel persistence to determine whether the results are consistent with the expressions. CAD models were created to model the expected surfaces for 0, 25, and 50% overlap ratios as shown in Fig. 3.20. 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 3.20 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. 3.21 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.



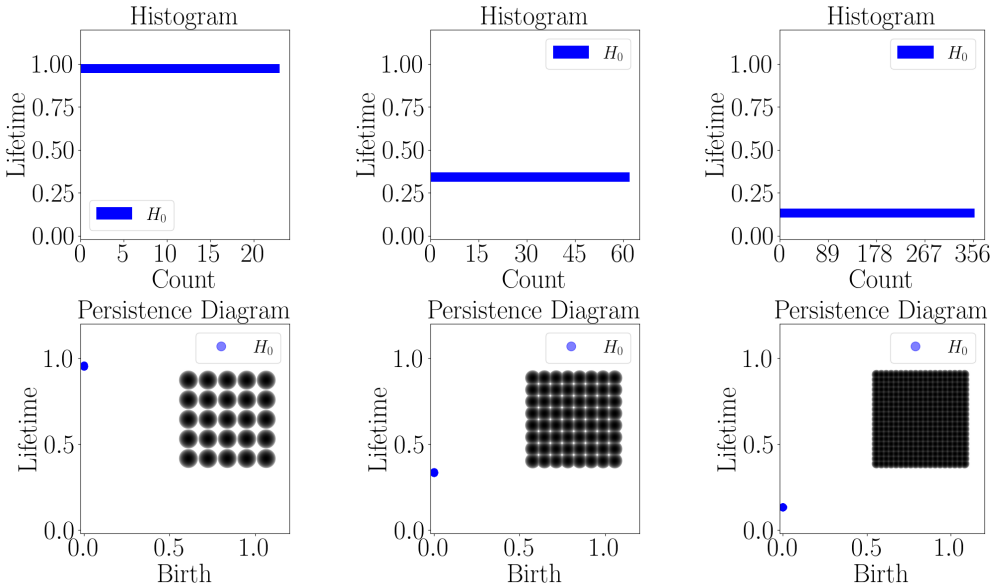
(a) CAD Model (b) STL Conversion (c) Point Cloud (d) Grayscale Image

Figure 3.21 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.

A Matlab script was implemented to convert the high resolution STL files into point clouds. Note that the point cloud shown in Fig. 3.21c was only plotting one point per 75 points for viewing clarity. After converting the model to a point cloud, the algorithm in [**?**] was used to convert the point cloud to a gray scale image and a bilinear interpolation created a smooth image as shown in Fig. 3.21d.

3.1.6.1 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. 3.22. Table 3.4 shows a comparison of the expected



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

Figure 3.22 Nominal CAD surface striking depth persistence diagrams and histograms for each overlap ratio.

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 3.4 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%**

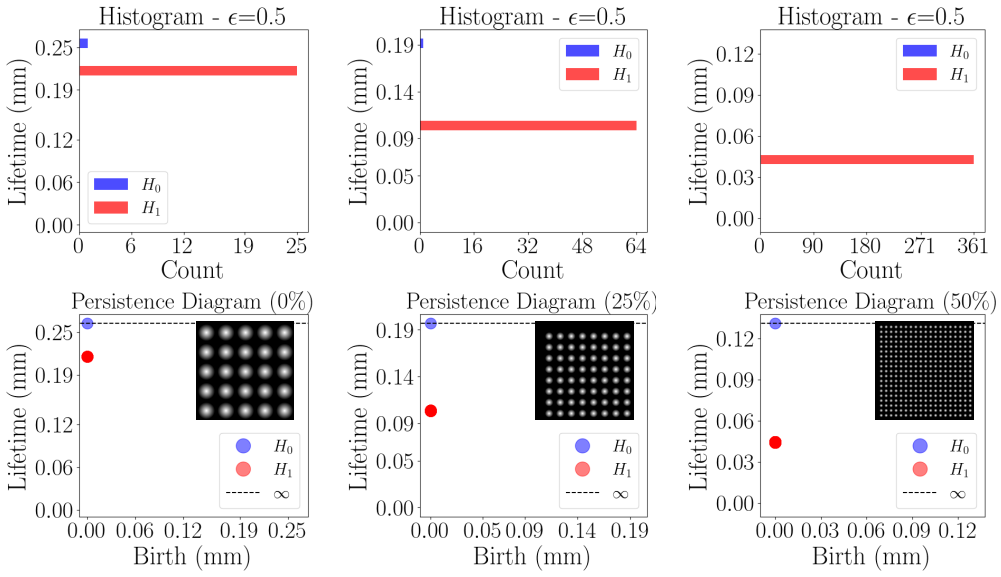
3.1.6.2 Strike Roundness

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 ($\varepsilon = 0.5$)

$$T = 0.5h, \tag{3.25}$$

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. 3.23. 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. 3.23. We see that the 1D persistence shows the expected number of



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

Figure 3.23 CAD model distance transformed image (strike roundness) persistence diagrams for $\varepsilon = 0.5$ ($T < \bar{h}$) at each overlap ratio.

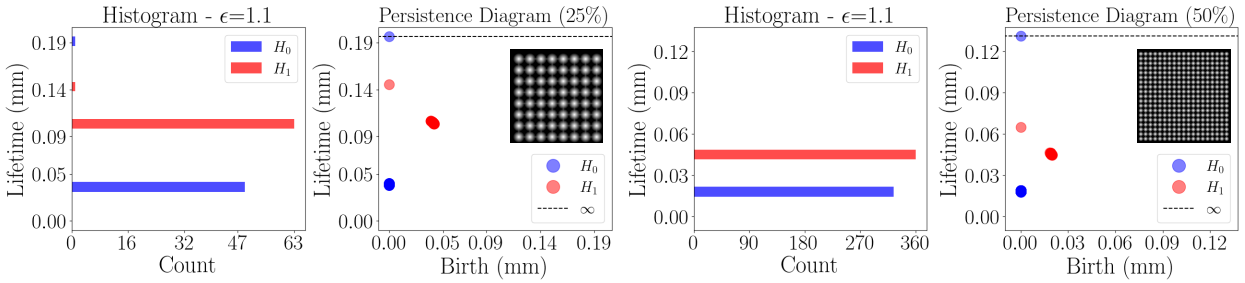
loops that are born at time 0 and die at various heights. Using Eq. (3.8), we have converted the pixel distances into distances in mm using $W = 2.55$ mm and $P = 5000$. The results in Table 3.5 are

Table 3.5 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%**

exceedingly 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 3.24 shows the thresholded images at this height. The corresponding persistence diagrams for $\varepsilon = 1.1$ are also shown in Fig. 3.24. 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 3.6. It was clear that the results were nearly identical to the theoretical results which verifies the expressions from Section 3.1.2.3.



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

Figure 3.24 CAD model distance transformed image (strike roundness) persistence diagrams for $\varepsilon = 1.1$ ($T > \overline{h}$) at each overlap ratio.

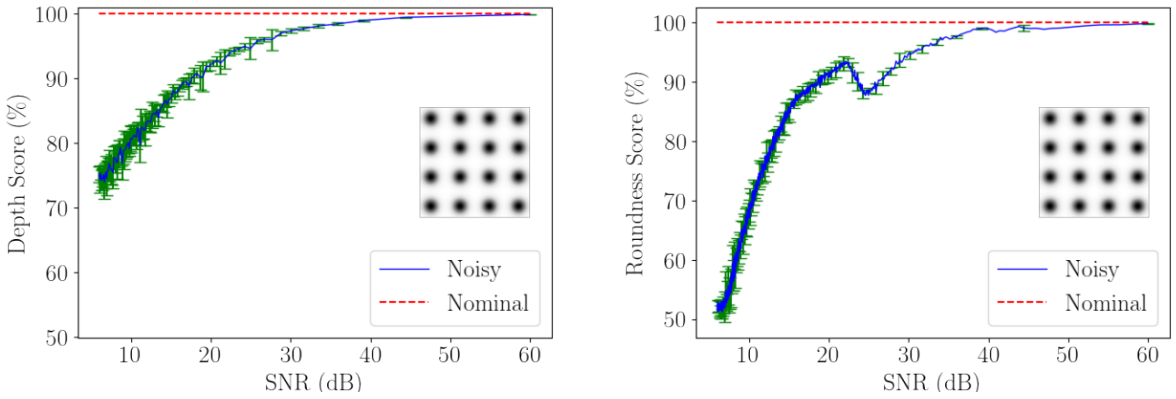
Table 3.6 Comparison of CAD model persistence results and theoretical strike **roundness** for each overlap ratio ($\varepsilon = 1.1$).

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%**

3.1.7 Appendix — Score Noise Study

3.1.7.1 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. 3.25 (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. 3.25 (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 3.25 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.

3.1.7.2 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. 3.25 (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.

3.1.8 Appendix — 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. 3.11 (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. 3.11 (c). For this image, the resulting plane has the form,

$$z = 0.1508 - 0.0003315 i_x - 0.0001764 i_y \tag{3.26}$$

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 3.7. Note that the image coordinate system was used for these slopes so the negative y direction points toward the top of the image.

Table 3.7 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

3.2 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.

3.2.1 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 [**?**]. A key quality in manufacturing is the surface texture which is directly related to surface roughness [**?**,**?**,**?**] and to the tactile feel of the resulting products which is quantified by tactile roughness [**?**, **?**]. 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 [**?**], and fatigue resistance [**?**, **?**, **?**].

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) [**?**]. In PVST, a tool is used to impact the surface and a scanning strategy is applied to treat the whole surface, see Fig. 3.26. 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 [**?**], 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 segmentation. An emerging tool that has shown promise for quantifying texture is from the field of

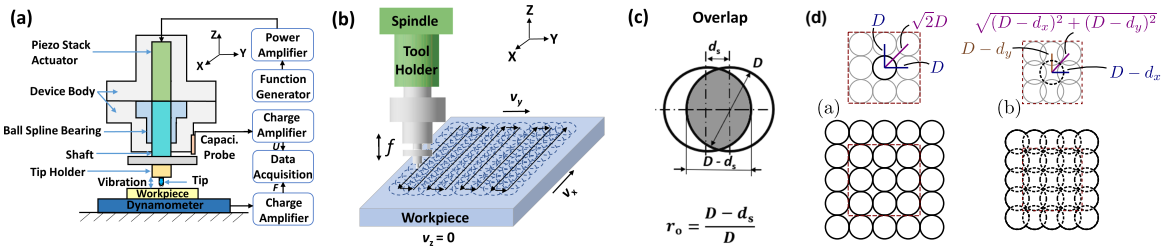


Figure 3.26 Schematics of PVST: (a) PVST; (b) Striking process in PVST; (c) Overlap of indentations; (d) Grid geometry. PVST nominal grid for (d1) x and y overlap $d_x = d_y = 0$, and d2) for x overlap $d_x \geq 0$ and y overlap $d_y \geq 0$. The striking tool diameter is D, and the figure shows a scanning 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 $d_x = d_y = d_s$.

Topological Data Analysis (TDA). More specifically, topological measures were used to quantify order of nearly hexagonal lattices [**?**] which represent nanoscale pattern formation on a solid surface that can result from broad ion beam erosion [**?**]. The input data in [**?**] 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 [**?**], 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 3.2.2 provides background for PVST. Section 3.2.3 explains the experimental setup and how the experimental data is collected. Section 3.2.4 outlines the processing performed on each image to obtain the point cloud data. Section 3.2.5 discusses the TDA-based approach proposed in this study. Section 3.2.6 compares the results of the analysis to a perfect square lattice. Section 3.2.9 includes the concluding remarks.

3.2.2 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 [**?**], Surface Mechanical Attrition Treatment [**?**], High-frequency Mechanical Impact Treatment [**?**], and Ultrasonic Nanocrystal Surface Modification [**?**], 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 [**?**, **?**]. The schematics of PVST are shown in Fig. 3.26. 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. 3.26a, 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. 3.26b-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.

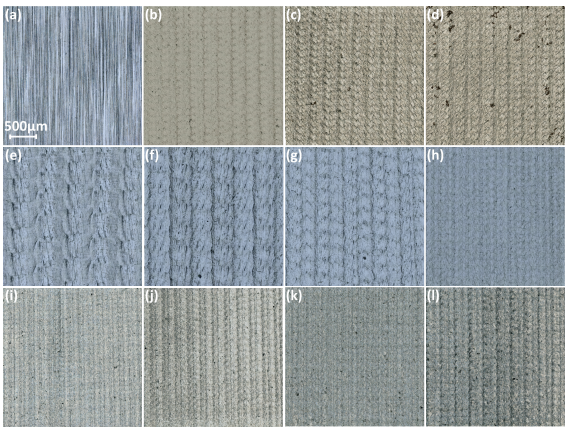


Figure 3.27 Various surface textures under different PVST conditions: (a) initial workpiece surface; (b) — (d) different Z values; (e) — (h) different overlap ratios; (i) — (l) different driving voltages.

3.2.3 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. 3.8).

Table 3.8 Various PVST conditions. The first column represents the samples in Fig. 3.27.

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. 3.28a. 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. 3.28b 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. 3.27. 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.

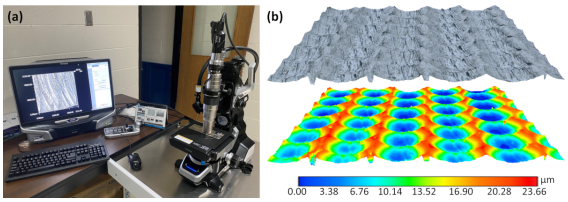


Figure 3.28 (a) KEYENCE digital microscope (b) illustrations of scanned surface texture.

3.2.4 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. 3.29.

3.2.4.1 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 [**?**]. 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.

3.2.4.2 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. 3.30 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.

3.2.4.3 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. 3.30. The reason for manually choosing a guess for the centers' locations, instead of using the nominal locations, is shown in Fig. 3.28b. 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. 3.28b) 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. 3.28b 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.

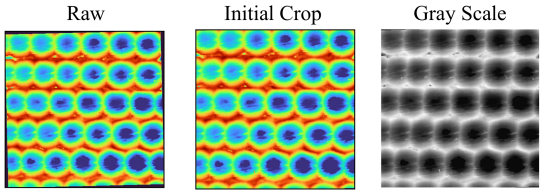


Figure 3.29 Preprocessing of a sample raw image.

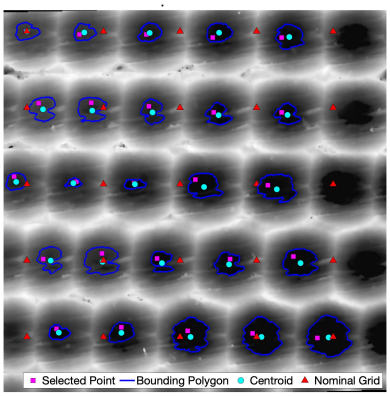


Figure 3.30 Locating Nominal and True Lattice Centers

3.2.5 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 [**?**, **?**, 9, 11, 13, 14].

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

3.2.5.1 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. 3.31a 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.

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. 3.31a shows that for $d = 0$ we have 36 distinct components that emerge or are *born*, Fig. 3.31b 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. 3.31f (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. 3.31 shows that the grid has interesting geometric features characterized by

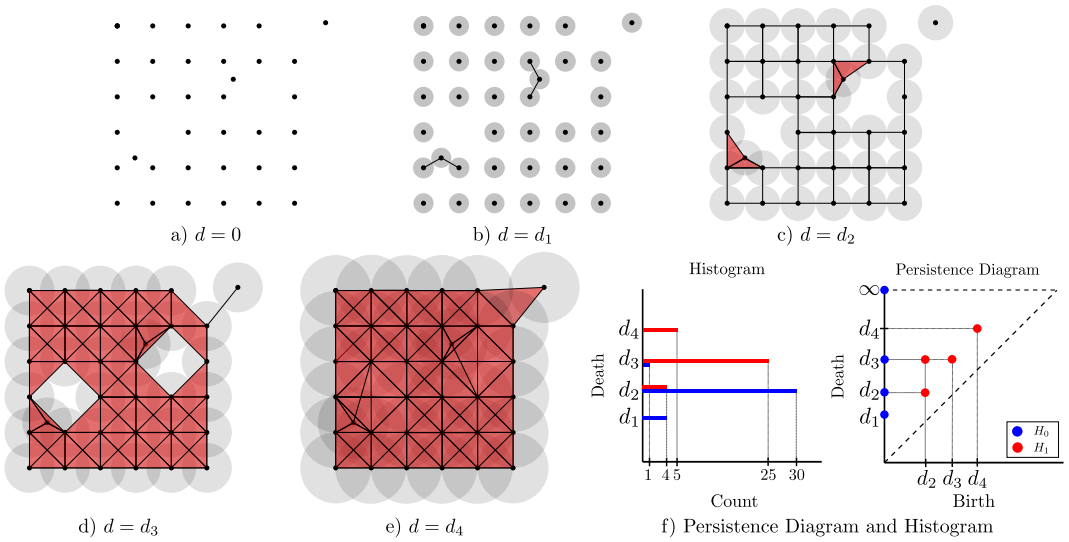


Figure 3.31 Persistence Example

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. 3.31f (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. 3.31b, 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. 3.31c 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. 3.31f. 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.

3.2.5.2 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. 3.32 and the connectivity parameter d was varied until the all of the rectangles were included in the complex.

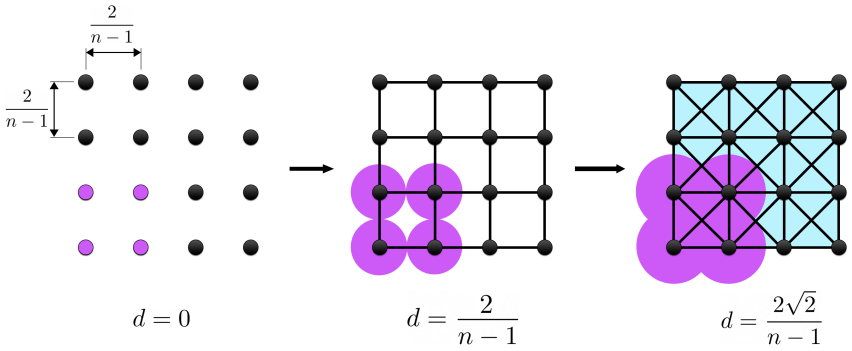


Figure 3.32 An illustration of the key values of the diameter d at which features are born and die for a perfect Square Lattice. The expanding disks (purple) are only shown for four points to better visualize the edges and the triangles that get added at each shown d value.

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 [**?**] that for both perfect square and perfect hexagonal lattices, the variance in the 0D persistence (lifetimes) is zero, which we express as

$$Var(H_0) = 0. \tag{3.27}$$

This expression can be used for measuring the deviation from a square/hexagonal lattice in the presence of a non-zero variance [**?**]. Note that the overlap ratios for the PVST images are 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. 3.33.

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 [**?**] so to normalize this measure, it was multiplied by a factor of 4.

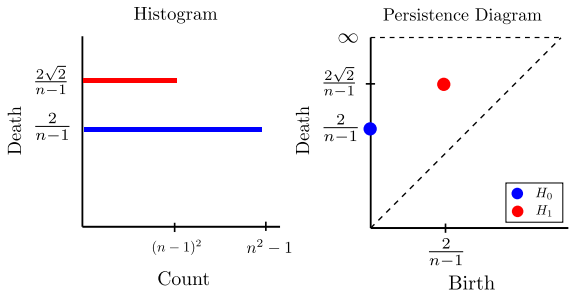


Figure 3.33 Theoretical Persistence Diagram for a perfect square lattice.

1-D Persistence For the 1-D persistence in a Vietoris–Rips complex, the presence of loops is of interest. Figure 3.32 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

$$H_1 = \frac{2\sqrt{2}}{n-1} - \frac{2}{n-1}. \tag{3.28}$$

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. (3.29) 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 [**?**].

$$\sum H_1 = 2(\sqrt{2}-1)(n-1). \tag{3.29}$$

The 1-D persistence diagram for the rectangular lattice is shown in Fig. 3.33. 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 [**?**]. 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. (3.29). Because the H_1 sum is nonzero for a square lattice, the CPH score approach presented in [**?**] 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

shown in equations (3.30a) and (3.30b).

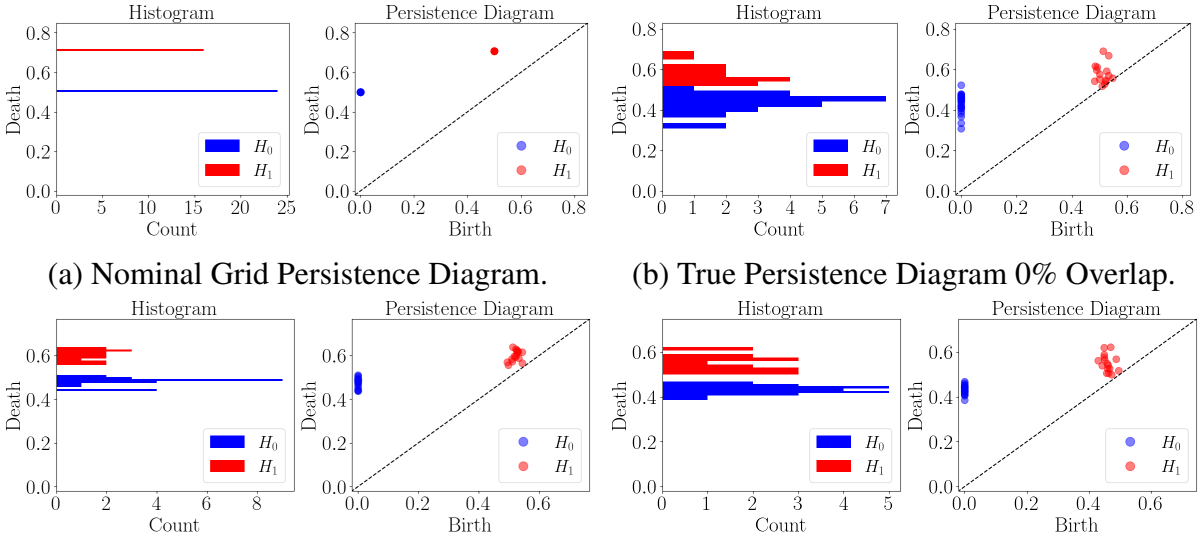
$$\overline{H_0} = 4Var(H_0), \tag{3.30a}$$

$$\overline{H_1} = \frac{1}{2(\sqrt{2}-1)(n-1)} \Sigma H_1, \tag{3.30b}$$

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.

3.2.6 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 3.2.5.2 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 3.34 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.

3.2.7 Persistence Diagrams

The generated point clouds from Section 3.2.4 were passed into Ripser, a python library used to generate persistence diagrams from point cloud data with a Vietoris-Rips Complex [**?**]. First, nominally generated grid point clouds were passed to Ripser to verify the expressions obtained in Section 3.2.5.2. The same persistence diagram resulted for all three of the grids shown in Fig. 3.34a, 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. 3.32 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. 3.34b—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.

3.2.8 Measures of Order

Equations (3.30a) and (3.30b) 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 3.9. 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 [**?**] 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 3.9 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

3.2.9 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 4

MODELING

This chapter presents my work on a time delay model for metabolic oscillations in yeast cells published in [**?**]. 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.

4.1 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.

4.1.1 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 [**?**], yeast populations in low growth conditions exhibit metabolic cycling (MC) [**?**, **?**] also known as respiratory cycling [**?**]. While traditionally described as a result of carbon limitation, limitations by other essential nutrients like phosphate [**?**] or ammonium, ethanol, glucose, and sulfur [**?**, **?**] 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 [**?**].

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 [**?**]. 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 [**?**, **?**, **?**]. Using experimental techniques ranging from micorarray analysis [**?**, **?**] to short-life luciferase fluorescent reporters [**?**], 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 [**?**]. 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 [**?**].

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 [**?**].

There were many different hypotheses centered on chemical signals that may mediate metabolic synchrony. In particular, Murray et. al. [**?**] proposed acetaldehyde and sulphate, Henson [**?**] and Sohn and Kuriyama [**?**] hydrogen sulphide, while Adams et. al. [**?**]found that Gts1 protein plays a key stabilizing role. Finally, Muller at. al. [**?**] 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 [**?**, **?**]. Finally, paper [**?**] 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 [**?**] 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 [**?**]. 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 [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**]. 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 4.1.2 we introduce a single protein model and extract theoretical results such as the fixed points and its linear stability behavior. Section 4.1.3 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 4.1.4 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 4.1.5 where the numerical methods are applied and the stability of the system is characterized in a sub-

set of the overall parameter space. We then apply the same methods to the three protein system in Section 4.1.6. Finally, we give concluding remarks in Section 4.1.7.

4.1.2 Theory — Single Protein

4.1.2.1 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 [**?**]. 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

$$\dot{p}(t) = B\mu(t-\tau) - Dp(t), \tag{4.1}$$

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:

$$\mu(t) = f(p(t))R(t), \tag{4.2}$$

with Hill function

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

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

$$R(t) = R_T - A\int_{t-\tau}^{t} \mu(s)ds, \tag{4.3}$$

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. (4.3) leads to

$$\dot{R}(t) = A\left(\mu(t-\tau) - \mu(t)\right). \tag{4.4}$$

We note that this differentiation step is only valid for constant delays and if variable delays are used, Eq. (4.3) must be used for analysis. Putting the equations together, we have the model

$$\begin{aligned} \dot{p}(t) &= Bf(p(t-\tau))R(t-\tau) - Dp(t), \\ \dot{R}(t) &= A\left(f(p(t-\tau))R(t-\tau) - f(p(t))R(t)\right). \end{aligned} \tag{4.5}$$

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.

4.1.2.2 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. (4.5). This process yields one equilibrium condition (Eq. (4.6)) because the equation for $\dot{R}(t)$ in Eq. (4.5) is satisfied for all constant p and R.

$$Dp^* = Bf(p^*)R^* \tag{4.6}$$

The other equilibrium conditions are obtained from Eq. (4.3) by assuming that the integrand is constant when equilibrium has been reached. This yields Eq. (4.7).

$$R^* = \frac{R_T}{1+A\tau f(p^*)} \tag{4.7}$$

We then substitute Eq. (4.7) into Eq. (4.6) to obtain,

$$p^* = \frac{Bf(p^*)R_T}{D(1+A\tau f(p^*))}. \tag{4.8}$$

Finally, inserting the definition of $f(p^*)$, we get a polynomial expression that must be satisfied for the equilibrium points as shown in Eq. (4.9).

$$(1+A\tau)p^{*n+1} - \frac{BR_T}{D}p^{*n} + \kappa^n p^* = 0 \tag{4.9}$$

Any solution to Eq. (4.9) can then be plugged in to Eq. (4.7) 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^*).

$$(1+A\tau)p^{*n} - \frac{BR_T}{D}p^{*n-1} + \kappa^n = 0, \tag{4.10a}$$

$$R^* = \frac{R_T}{1+A\tau f(p^*)}. \tag{4.10b}$$

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. (4.9) for $n = 2$ yielded three equilibrium points:

$$\begin{aligned}(p^*, R^*) &= (p_{\text{trivial}}, R_{\text{trivial}}),\\ (p^*, R^*) &= (p_{\text{middle}}, R_{\text{middle}}),\\ (p^*, R^*) &= (p_{\text{top}}, R_{\text{top}}),\end{aligned}$$

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:

$$R_T < \sqrt{\frac{4D^2\kappa^2(1+A\tau)}{B^2}}. \tag{4.11}$$

This boundary given by the equality of Eq. (4.11) is plotted in the stability diagrams in Sec. 4.1.5 as a red curve with triangles. So any parameters that satisfy (4.11) 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.

4.1.2.3 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 [39]. We linearize (4.5) by computing the Jacobian matrices of the present and delayed states about an equilibrium 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,

$$\dot{\vec{x}}(t) = \vec{g}(\vec{x}, \vec{x}_\tau), \tag{4.12}$$

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.

$$\mathbf{G}_1 = \frac{\partial \vec{g}_1}{\partial x} = \begin{bmatrix} \frac{\partial \vec{g}_{11}}{\partial x_1} & \frac{\partial \vec{g}_{11}}{\partial x_2} \\ \frac{\partial \vec{g}_{12}}{\partial x_1} & \frac{\partial \vec{g}_{12}}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -D & 0 \\ -Af'(x_1)x_2 & -Af(x_1) \end{bmatrix}, \tag{4.13}$$

and

$$\mathbf{G}_2 = \frac{\partial \vec{g}_2}{\partial x_\tau} = \begin{bmatrix} \frac{\partial \vec{g}_{21}}{\partial x_{1\tau}} & \frac{\partial \vec{g}_{21}}{\partial x_{2\tau}} \\ \frac{\partial \vec{g}_{22}}{\partial x_{1\tau}} & \frac{\partial \vec{g}_{22}}{\partial x_{2\tau}} \end{bmatrix} = \begin{bmatrix} Bf'(x_{1\tau})x_{2\tau} & Bf(x_{1\tau}) \\ Af'(x_{1\tau})x_{2\tau} & Af(x_{1\tau}) \end{bmatrix}, \tag{4.14}$$

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

$$\dot{\vec{x}} \approx \begin{bmatrix} -D & 0 \\ -Af'(x_1)x_2 & -Af(x_1) \end{bmatrix}\Big|_q (\vec{x} - \vec{q}) + \begin{bmatrix} Bf'(x_{1\tau})x_{2\tau} & Bf(x_{1\tau}) \\ Af'(x_{1\tau})x_{2\tau} & Af(x_{1\tau}) \end{bmatrix}\Big|_q (\vec{x}_\tau - \vec{q}). \tag{4.15}$$

If a change of variables, $\vec{y} = \vec{x} - \vec{q}$ is implemented, with $\vec{y}_\tau = \vec{x}_\tau - \vec{q}$, Eq. (4.15) simplifies to Eq. (4.16) effectively moving the equilibrium point to the origin.

$$\dot{\vec{y}} \approx \begin{bmatrix} -D & 0 \\ -Af'(p^*)R^* & -Af(p^*) \end{bmatrix} \vec{y} + \begin{bmatrix} Bf'(p^*)R^* & Bf(p^*) \\ Af'(p^*)R^* & Af(p^*) \end{bmatrix} \vec{y}_\tau. \tag{4.16}$$

We can write Eq. (4.16) in a simplified form as,

$$\dot{\vec{y}} \approx \mathbf{G}_1(q)\vec{y} + \mathbf{G}_2(q)\vec{y}_\tau. \tag{4.17}$$

We will use this linearized system to evaluate the stability of the equilibrium points of the system. Note that Eq. (4.17) was derived only from the DDE system Eq. (4.5). In addition, the constraint Eq. (4.3) 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 (4.17), which leads to the elementary ODE system

$$\dot{\vec{y}} \approx \begin{bmatrix} -D & 0 \\ 0 & 0 \end{bmatrix} \vec{y}, \tag{4.18}$$

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 (4.3), 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 [**?**].

4.1.3 Theory — Three Protein Model

4.1.3.1 Model

We extend the single protein model in Eq. (4.5) 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. (4.19).

$$\begin{aligned}
\dot{p_1}(t) &= B_1 f(p_2(t-\tau_1)) f(p_3(t-\tau_1)) R(t-\tau_1) - D_1 p_1, \\
\dot{p_2}(t) &= B_2 f(p_1(t-\tau_2)) R(t-\tau_2) - D_2 p_2, \\
\dot{p_3}(t) &= B_3 f(p_1(t-\tau_3)) R(t-\tau_3) - D_3 p_3, \\
\dot{R}(t) &= A(\mu_1(t-\tau_1) + \mu_2(t-\tau_2) + \mu_2(t-\tau_3) - \mu_1(t) - 2\mu_2(t)),
\end{aligned} \tag{4.19}$$

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,

$$\begin{aligned}
R_T = R(t) + A \Bigg(&\int_{t-\tau_1}^{t} f(p_2(s)) f(p_3(s)) R(s) ds + \int_{t-\tau_2}^{t} f(p_1(s)) R(s) ds + \\
&\int_{t-\tau_3}^{t} f(p_1(s)) R(s) ds \Bigg).
\end{aligned} \tag{4.20}$$

4.1.3.2 Equilibrium Points

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

$$\begin{aligned}
D_1 p_1^* &= B_1 f(p_2^*) f(p_3^*) R^*, \\
D_2 p_2^* &= B_2 f(p_1^*) R^*, \\
D_3 p_3^* &= B_3 f(p_1^*) R^*,
\end{aligned} \tag{4.21}$$

where (p_1^*, p_2^*, p_3^*,R^*) is the equilibrium point. Similarly to the single protein system, the $\dot{R}$ expression in Eq. (4.19) is always satisfied at equilibrium, but Eq. (4.20) yields the fourth and final equilibrium condition:

$$R_T = R^* + A\left[f(p_2^*)f(p_3^*)R^*\tau_1 + f(p_1^*)R^*(\tau_2+\tau_3)\right]. \tag{4.22}$$

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. (4.21) all depend directly on R^*. We eliminate R^* by solving Eq. (4.22) for R^*,

$$R^* = \frac{R_T}{1 + A\left(f(p_2^*)f(p_3^*)\tau_1 + f(p_1^*)(\tau_2+\tau_3)\right)}, \tag{4.23}$$

and substituting R^* in the three Eqs. (4.21), along with using the definition of $f(x)$. These steps result in three multivariate polynomial equilibrium equations shown in Eqs. (4.24),(4.25) and (4.26).

$$\begin{aligned} &D_1 p_1^*(\kappa^n + p_1^{*n})(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}) + AD_1 p_1^* p_2^{*n} p_3^{*n}(\kappa^n + p_1^{*n})\tau_1 \\ &\quad + AD_1 p_1^{*(n+1)}(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n})(\tau_2+\tau_3) = B_1 R_T p_2^{*n} p_3^{*n}(\kappa^n + p_1^{*n}), \end{aligned} \tag{4.24}$$

$$\begin{aligned} &D_2 p_2^*(\kappa^n + p_1^{*n})(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}) + AD_2 p_2^{*(n+1)} p_3^{*n}(\kappa^n + p_1^{*n})\tau_1 \\ &\quad + AD_2 p_1^{*n} p_2^*(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n})(\tau_2+\tau_3) = B_2 R_T p_1^{*n}(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}), \end{aligned} \tag{4.25}$$

$$\begin{aligned} &D_3 p_3^*(\kappa^n + p_1^{*n})(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}) + AD_3 p_2^{*n} p_3^{*(n+1)}(\kappa^n + p_1^{*n})\tau_1 \\ &\quad + AD_3 p_1^{*n} p_3^*(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n})(\tau_2+\tau_3) = B_3 R_T p_1^{*n}(\kappa^n + p_2^{*n})(\kappa^n + p_3^{*n}). \end{aligned} \tag{4.26}$$

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. (4.23). Due to the complexity of these equations, we solve them numerically using the variable precision (VPA) solver in `Matlab` [**?**]. Details for how these equations were solved are outlined in Sec. 4.1.6.

4.1.3.3 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 4.1.4.1. We will linearize the system about the equilibrium point, $\vec{q} = \left[p_1^*\ p_2^*\ p_3^*\ R^*\right]^T$ from the solution to Eqs. (4.24),(4.25), and (4.26). 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,

$$\vec{x} = \begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \\ R(t) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \ \vec{x}_{\tau_i} = \begin{bmatrix} p_1(t-\tau_i) \\ p_2(t-\tau_i) \\ p_3(t-\tau_i) \\ R(t-\tau_i) \end{bmatrix} = \begin{bmatrix} x_{1\tau_i} \\ x_{2\tau_i} \\ x_{3\tau_i} \\ x_{4\tau_i} \end{bmatrix}, \tag{4.27}$$

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

$$\dot{\vec{x}} = \vec{g}(\vec{x}, \vec{x}_{\tau_1}, \vec{x}_{\tau_2}, \vec{x}_{\tau_3}), \tag{4.28}$$

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,

$$\dot{\vec{x}} \approx \mathbf{G}_1(\vec{q})(\vec{x} - \vec{q}) + \sum_{i=2}^{4} \mathbf{G}_i(q)(\vec{x}_{\tau_i} - \vec{q}), \tag{4.29}$$

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.

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

$$\mathbf{G}_2(\vec{q}) = \begin{bmatrix} 0 & B_1 f'(p_2^*)f(p_3^*)R^* & B_1 f(p_2^*)f'(p_3^*)R^* & B_1 f(p_2^*)f(p_3^*) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & Af'(p_2^*)f(p_3^*)R^* & Af(p_2^*)f'(p_3^*)R^* & Af(p_2^*)f(p_3^*) \end{bmatrix},$$

$$\mathbf{G}_3(\vec{q}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ B_2 f'(p_1^*)R^* & 0 & 0 & B_2 f(p_1^*) \\ 0 & 0 & 0 & 0 \\ Af'(p_1^*)R^* & 0 & 0 & Af(p_1^*) \end{bmatrix},$$

$$\mathbf{G}_4(\vec{q}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ B_3 f'(p_1^*)R^* & 0 & 0 & B_3 f(p_1^*) \\ Af'(p_1^*)R^* & 0 & 0 & Af(p_1^*) \end{bmatrix},$$

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. (4.29) resulting in the linearized system:

$$\dot{y} \approx \mathbf{G}_1(\vec{q})\vec{y}(t) + \mathbf{G}_2(\vec{q})\vec{y}(t-\tau_1) + \mathbf{G}_3(\vec{q})\vec{y}(t-\tau_2) + \mathbf{G}_4(\vec{q})\vec{y}(t-\tau_3). \tag{4.30}$$

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 ($D_1 > 0$, $D_2 > 0$, $D_3 > 0$).

4.1.4 Methods

4.1.4.1 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 [**?**]. In particular, the

linear variational systems Eq. (4.17) and Eq. (4.30) 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

$$\vec{z}_n = \mathbf{U}\vec{z}_{n-1}, \tag{4.31}$$

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 [**?**]. 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 [**?**]. 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 [**?**]. 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 [**?**]. 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 [**?**]. 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 [**?**]. 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 [**?**] 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 [**?**] are applied for discretizing and computing dominant eigenvalues for the metabolic systems to evaluate the system stability at different parameters.

4.1.4.2 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` [**?**]. 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:

$$M_A = \sum_{i=1}^{n} A_i. \tag{4.32}$$

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.

4.1.4.3 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 [**?**]. 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. (4.3) yielding:

$$R_T = R_0 + A\int_{-\tau}^{0} f(p(s))R(s)ds, \tag{4.33}$$

where $R_0 = R(0)$ is the resource value at time $t = 0$. For a response of this system to be valid, Eq. (4.33) 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. (4.33) 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. (4.20) 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.

4.1.4.4 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 [**?**] 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. (4.34) 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 [**?**].

$$\begin{aligned} &\vec{f} = \frac{d\vec{x}}{dt} - T\vec{g}(\vec{x}(t), \vec{x}(t-\tau/T)) = 0,\ t \in [0,1], \\ &\vec{x}(s) - \vec{x}(s+1) = 0,\ s \in [-\tau/T, 0], \\ &p(\vec{x}) = 0, \end{aligned} \tag{4.34}$$

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. (4.12) for the single protein system, and Eq. (4.28) 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 $\dot{\vec{x}}(t)$ [**?**]. 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 [**?**] making it ideal for verifying parameters of the metabolic system that result in oscillating solutions to the system.

4.1.5 Results — Single Protein

The single protein system is analyzed in this section by applying the three methods outlined in Sec. 4.1.4.

4.1.5.1 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 [**?**].

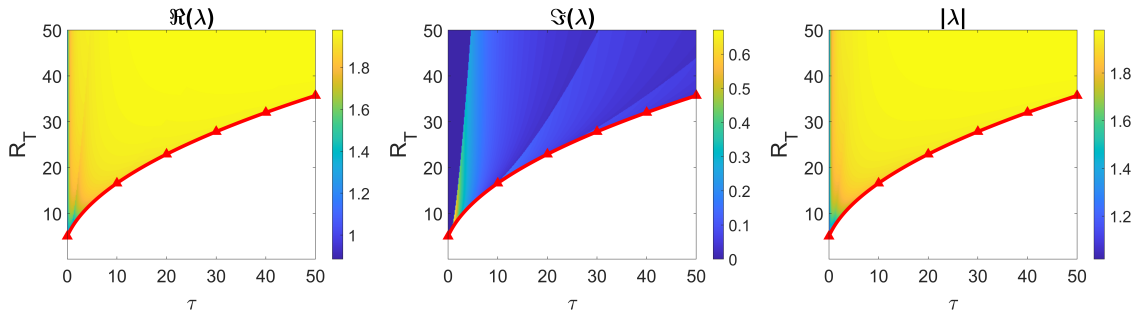


Figure 4.1 Single protein middle equilibrium point stability diagrams. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant eigenvalue, (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.

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 [**?**]. 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. (4.11), 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 4.1.2.3. 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. 4.1 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. 4.2 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 [**?**].

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,

$$R_T = 2.6449\tau + 4.6323, \tag{4.35}$$

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.

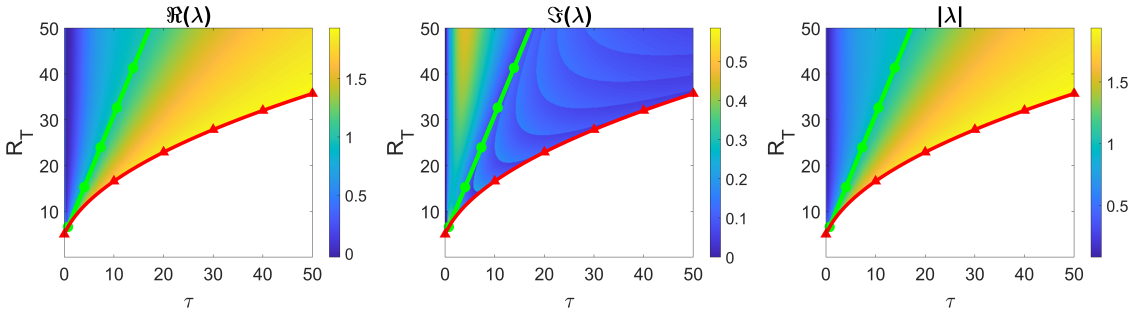


Figure 4.2 Single protein top equilibrium point stability diagrams. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant eigenvalue, (right) 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.

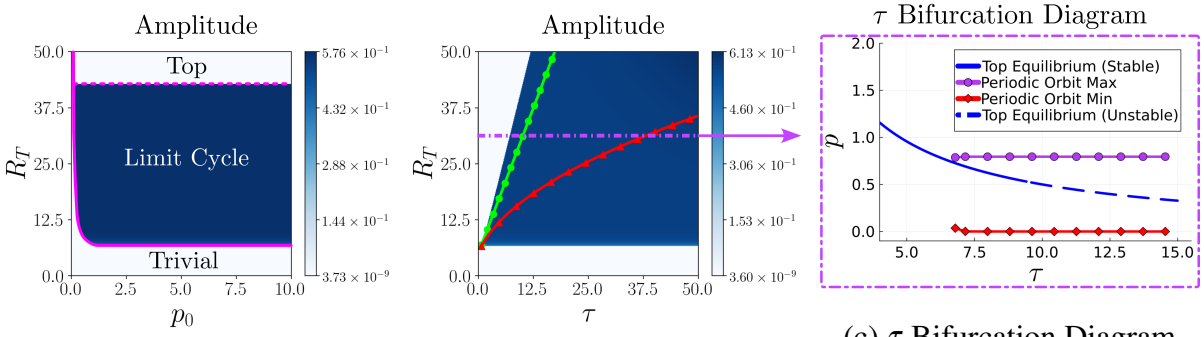
4.1.5.2 Response Features

Response feature diagrams were generated for the single protein system (Eq. (4.5)) 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 shown in Fig. 4.3 where we color pixels in the parameter space according to the response amplitude feature using Eq. 4.32. We used the starving cell history function from Section 4.1.4.3 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. 4.3 (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. 4.3 (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. 4.3 (a) are specific to $\tau = 10$ and will increase as the delay is increased according to the bounds of the periodic region in Fig. 4.3 (b). In other words, at a delay of 10 if we draw a vertical

line in Fig. 4.3 (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. 4.3 (a) for nontrivial p_0. We can also show a horizontal slice of Fig. 4.3 (b) which produces a bifurcation diagram in τ as shown in Fig. 4.3 (c). The stable periodic orbit was generated by setting $R_T = 30$ and using simulations with the initial conditions from Fig. 4.3 (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 4.3 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.

4.1.5.3 Periodic Solutions

Next we utilize the spectral element approach to solve the boundary value problem in Eq. (4.34). 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.(4.34) for the periodic solution, we obtain the response shown in Fig. 4.4. 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

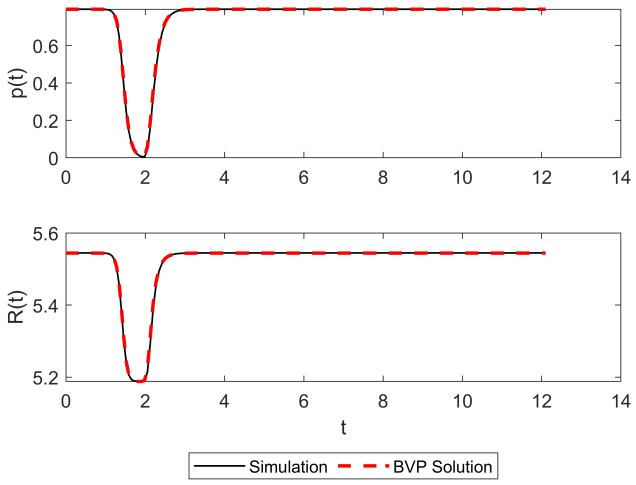


Figure 4.4 $\tau = 12$ and $R_T = 50$ single protein periodic solution results from solving the relevant boundary value problem.

at these parameters from 15,990–16,000 time units and the period was set to 10. Passing this initial guess into the boundary value problem, the obtained periodic solution is shown in Fig. 4.5. Interestingly, we see that the time that the protein production rate spends at 0 is much longer compared to Fig. 4.4. 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.

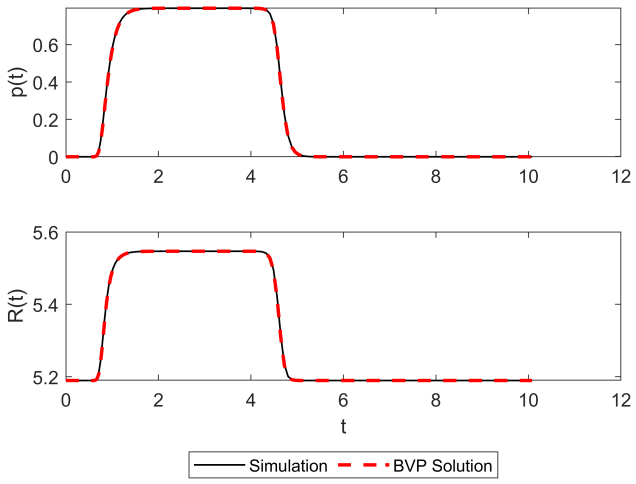


Figure 4.5 $\tau = 10$ and $R_T = 20$ single protein periodic solution results from solving the relevant boundary value problem.

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. 4.6. 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. 4.3 (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.

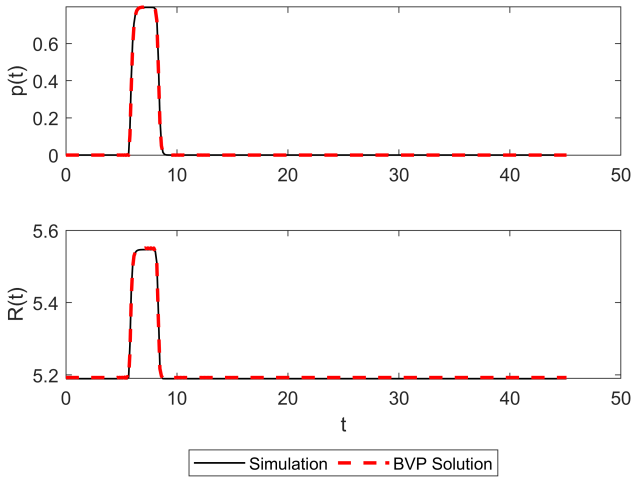


Figure 4.6 $\tau = 45$ and $R_T = 15$ single protein periodic solution results from solving the relevant boundary value problem.

4.1.5.4 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. 4.7 where we see the system approach $(p,R) = (0,R_T)$. We also consider the case where the

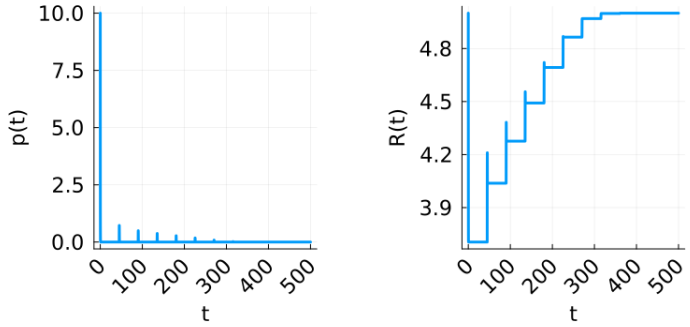


Figure 4.7 Approach to trivial fixed point for low growth conditions in the cell ($\tau = 45$, $R_T = 5$)

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. 4.8. 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 $(p^*,R^*) \approx (1.6412, 8.968)$.

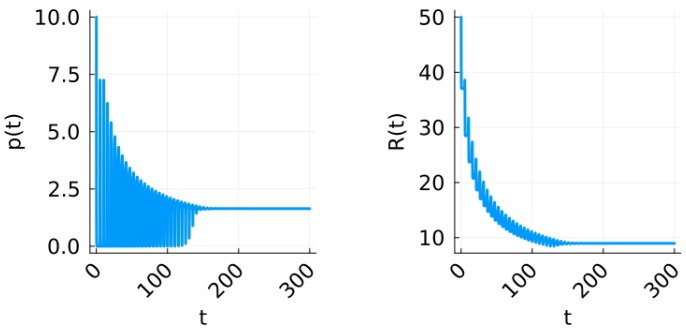


Figure 4.8 Fixed point response for high growth conditions in the cell ($\tau = 5$, $R_T = 50$)

4.1.5.5 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.

4.1.6 Results — Three Proteins

The three protein system is analyzed in this section by applying the three methods outlined in Sec. 4.1.4.

4.1.6.1 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. (4.24), (4.25), and (4.26). 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 [**?**]. 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. (4.23) 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. (4.30) 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 4.1.4.1. 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. 4.9. 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,

$$R_T = 0.0016\tau^3 - 0.1118\tau^2 + 4.8855\tau + 10.3749, \tag{4.36}$$

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. 4.10. 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 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,

$$R_T = 12.0948\tau + 4.7910, \tag{4.37}$$

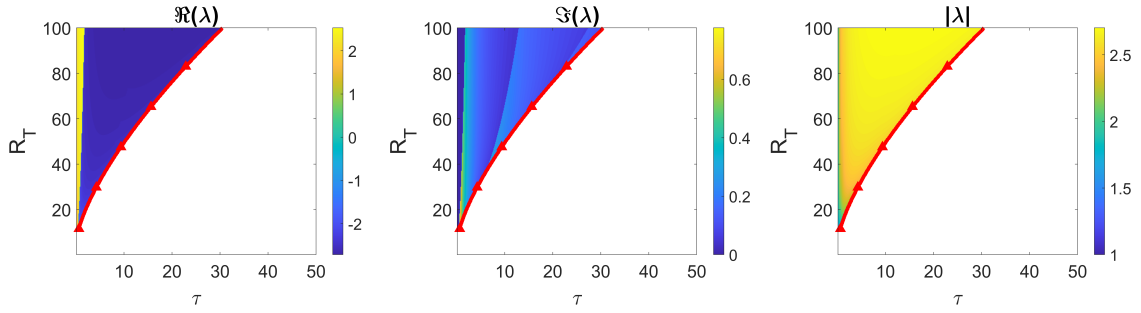


Figure 4.9 Three protein middle equilibrium point stability diagrams at equal delay. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant 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.

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.

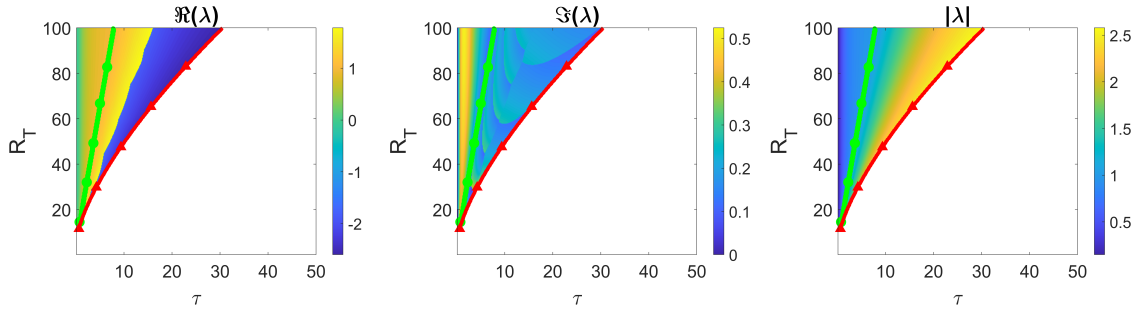


Figure 4.10 Three protein top equilibrium point stability diagrams at equal delay. Specifically, the eigenvalues with maximum magnitude of the monodromy matrix are plotted with respect to the parameters τ and R_T. (left) the real part of the dominant eigenvalue, (middle) imaginary part of the dominant 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.

4.1.6.2 Response Features

Holding the remaining parameters constant at the values from Section 4.1.6.1, 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. 4.1.4.3, and the amplitude feature was plotted in

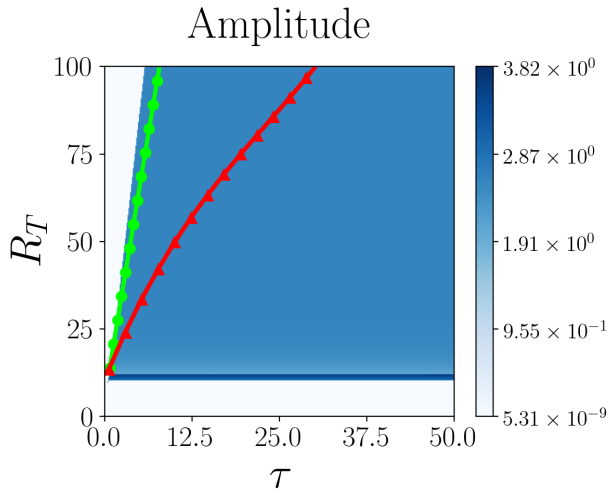


Figure 4.11 Three protein system response amplitude diagram in the $\tau - R_T$ parameter space where τ is the same delay for all three proteins. The low growth history function was used for all simulations in this diagram.

the $\tau - R_T$ space where τ is the same for all three proteins. The results are shown in Fig. 4.11. 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 in these regions are explored further in Section 4.1.6.3. We plot the Hopf bifurcation curve from Section 4.1.6.1 in Fig. 4.11 as the green line with dots. The red curve with triangles corresponds to the saddle node bifurcation curve from the approximations in Section 4.1.6.1.

4.1.6.3 Periodic Solutions

Three points were considered within the region of the parameter space with nontrivial amplitude in Fig. 4.11. 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. 4.12. 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.

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. 4.11 with the same amplitude as the solution

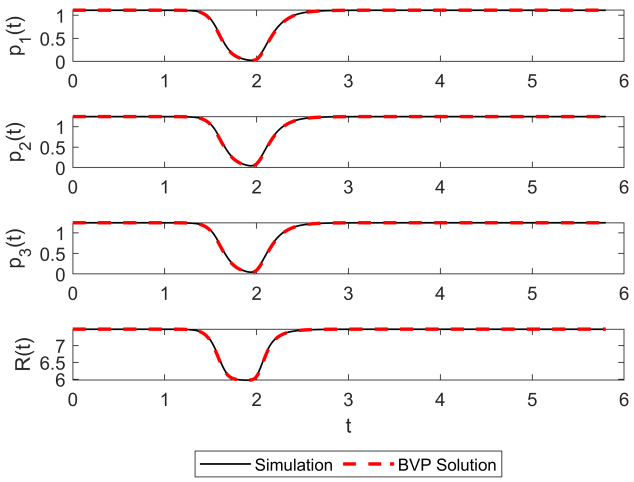


Figure 4.12 Boundary value problem solution for the three protein system with $\tau = 5.7$ and $R_T = 100$.

in Fig. 4.12. Plotting the response for these parameters between 15,000 and 15,025 time units yielded Fig. 4.13. 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

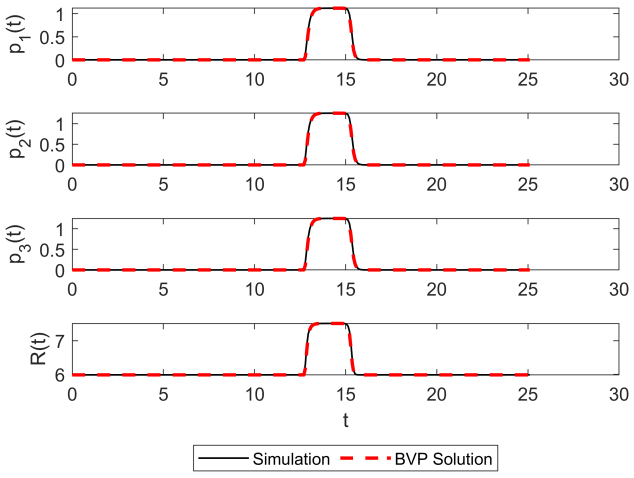


Figure 4.13 Boundary value problem solution for the three protein system with $\tau = 25$ and $R_T = 50$.

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. 4.14. 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.

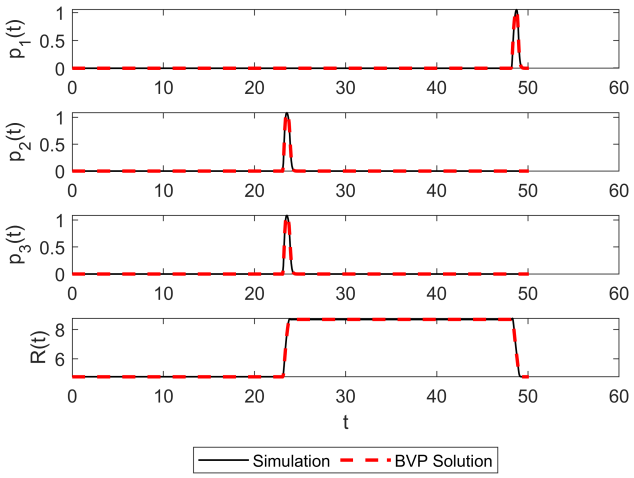


Figure 4.14 Boundary value problem solution for the three protein system with $\tau = 25$ and $R_T = 11.8$.

4.1.6.4 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 [**?**]. 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 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.

4.1.7 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] 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.

[3] M. Carriere, F. Chazal, M. Glisse, Y. Ike, and H. Kannan, "Optimizing persistent homology based functions," 10 2020.

[4] 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.

[5] 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.

[6] A. D. Myers, M. M. Chumley, F. A. Khasawneh, and E. Munch, "Persistent homology of coarse-grained state-space networks," *Physical Review E*, vol. 107, no. 3, p. 034303, 2023.

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

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

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

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

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

[12] 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.

[13] 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.

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

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

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

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

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

[19] 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.

[20] 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.

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

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

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

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

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

[26] 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.

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

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

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

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

[35] 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.

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

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

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

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

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

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

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

[48] 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 computin (un)stable manifolds of vector fields," in *World Scientific Series on Nonlinear Science Series B*, pp. 67–95, World Scientific, 3 2006.

[49] 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.

[50] 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.

[51] 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.

[52] E. Munch, "Teaspoon." `https://github.com/lizliz/teaspoon`, 2018.

[53] 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.

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

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

[56] 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.

[57] 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.

[58] 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.

[59] 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.

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

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

[62] 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.

[63] 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.

[64] M. J. Asher, B. F. Croke, A. J. Jakeman, and L. J. Peeters, "A review of surrogate models and their application to groundwater modeling," *Water Resources Research*, vol. 51, no. 8, pp. 5957–5973, 2015.

[65] F. Darema, A. Aved, E. Blasch, and S. e. Ravela, *Handbook of Dynamic Data Driven Applications Systems*, vol. 2. Springer, 09 2023.

[66] F. Darema, E. Blasch, S. Ravela, and A. A. (eds.), eds., *Dynamic Data Driven Applications Systems - Third International Conference, DDDAS 2020, Boston, MA, USA, October 2-4, 2020, Proceedings*, vol. 12312 of *Lecture Notes in Computer Science*, Springer, 2020.

[67] 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.

[68] M. C. Yesilli, F. A. Khasawneh, and A. Otto, "Topological feature vectors for chatter detection in turning processes," *The International Journal of Advanced Manufacturing Technology*, vol. 119, pp. 5687–5713, jan 2022.

[69] A. Myers and F. A. Khasawneh, "On the automatic parameter selection for permutation entropy," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, p. 033130, mar 2020.

[70] A. D. Myers, M. Yesilli, S. Tymochko, F. Khasawneh, and E. Munch, "Teaspoon: A comprehensive python package for topological signal processing," in *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

[71] A. M. Fraser and H. L. Swinney, "Independent coordinates for strange attractors from mutual information," *Physical review A*, vol. 33, no. 2, p. 1134, 1986.

[72] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Discrete & Computational Geometry*, vol. 37, pp. 103–120, dec 2006.