MESHFREE DATA DRIVEN SCATTERED INTERPOLATION ESTIMATES OF LYAPUNOV EXPONENTS USING OPTIMIZED RADIAL BASIS FUNCTIONS

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Mathematics

with a Concentration in

Dynamical Systems

---

by

Nathanael J. Reynolds

Spring 2022

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Nathanael J. Reynolds:

MESHFREE DATA DRIVEN SCATTERED INTERPOLATION ESTIMATES OF LYAPUNOV EXPONENTS USING OPTIMIZED RADIAL BASIS FUNCTIONS

Christopher W. Curtis, Chair
Department of Mathematics and Statistics

Bowen Shen
Department of Mathematics and Statistics

Arlette Baljon
Department of Physics

Approval Date

DEDICATION

For my mother, Laurie Reynolds, who has always supported me, Perri Gellman, for putting me down the road of mathematics, and the Puyallup Tribe of Indians whom without their financial support, this would not be possible.

We are all apprentices in a craft where no one ever becomes a master.

– Ernest Hemingway

ABSTRACT OF THE THESIS

MESHFREE DATA DRIVEN SCATTERED INTERPOLATION ESTIMATES OF LYAPUNOV EXPONENTS USING OPTIMIZED RADIAL BASIS FUNCTIONS

by
Nathanael J. Reynolds
Master of Science in Applied Mathematics with a Concentration in Dynamical Systems
San Diego State University, 2022

This work develops a globally convergent, data driven algorithm for estimating the Lyapunov exponents of a dynamical system. This work rigorously tests our algorithm with two dynamical systems: The Lorenz equation and the Rossler equation. Additionally the algorithm is used to approximate the Lyapunov exponents of two additional dynamical systems, the Chua equation, and the Li equation *Li et al.* [14]. The time series used as data points for the dynamical system was obtained by numerical integration with the fourth-order Runge-Kutta scheme. Our algorithm then interpolates the time series using radial basis functions. Three basic functions—Gaussian, multiquadrics, and inverse—were tested using our algorithm. The shape parameter of the basic function was optimized using an algorithm developed by *Rippa* [21]. Our interpolation algorithm is used to produce a numerical approximation of the given dynamical system's Jacobian. The Lyapunov exponents of the system are calculated by performing a QR-decomposition on the Jacobian approximation. We conducted several trials with Lorenz and Rossler by varying the simulation parameters t, dt, and nearest neighbors. Our trials provided good approximations of the Lorenz Lyapunov exponents except for two trials where $t = 90$ and one where $dt = 0.1$. Our algorithm also produced good approximations of the Lyapunov exponents for the Chua and Li equations. However, the algorithm struggled providing accurate approximations of the minimal Lyapunov exponent for Rossler but provided accurate approximations for Rossler's maximal Lyapunov exponenent.

TABLE OF CONTENTS

PAGE

LIST OF TABLES

PAGE

LIST OF FIGURES

PAGE

ACKNOWLEDGMENTS

I would like to thank Dr. Curtis for guidance on this topic and his personal kindness, Dr. Shen for his expertise on the Lornenz system and providing resources and feedback for this thesis, Joe Diaz for his positivity and always being helpful no matter what the need was, Shashwat Sharan for his friendship and his help in getting me through tough classes, Andrew Tuma for giving me a sense of camaraderie through this thesis writing process, Jay Lago for always being there to answer questions about research, the applied mathematics graduate cohort of 2022 for fun times and being awesome, and last but not least the Department of Mathematics and Statistics for providing the education that ultimately made this thesis possible.

CHAPTER 1
INTRODUCTION

In 1963, Edward Lorenz made his famous discovery of chaos while studying atmospheric convection. Since Lorenz's discovery, chaos has been a cornerstone in the field of dynamical systems. Both natural and man-made systems can exhibit chaotic behavior so understanding them is of great importance to science. One of the ways to study chaotic systems is by their Lyapunov exponents. Lyapunov exponents are a measure of how quickly the trajectories of two distinct and arbitarily distant points on an attractor seperate. The Lyapunov exponent is a measure of how predictable a given dynamical system is and gives a measure of the attractor's entropy.

The use of meshfree local regression methods have been used independently in statistics dating back to the 19th Century [3]. The use of radial basis functions (RBFs) as a method for meshfree approximations finds its origin with applications for geodesy, geophysics, mapping, and meteorology ([8], pp.13). RBFs have found use in other fields such as finding numerical solutions to PDEs, computer graphics, signal and image processing, sampling theory, statistics, finance, and optimization ([8], pp.1).

This thesis explores a data-driven method for estimating the Lyapunov exponents of a dynamical system. We call this method meshfree scattered data interpolation using radial basis functions (MSDI-RBF). The method works by obtaining time series data from a dynamical system via numerical integration and inputting the time series data into an interpolation matrix. Each matrix element contains a basic function (see §1.4) where data points in time series and a shape parameter ϵ (see §2.2), are taken as arguments of the function. The interpolation matrix elements use ϵ to find the best fit to the data. This thesis explores a method for finding an optimal ϵ—developed by *Rippa* [21]—as we move along the attractor. This allows the interpolation matrix to adapt to changing conditions on the attractor. The interpolation matrix is used to calculate an estimate of the dynamical system's Jacobian matrix which is then used in a QR-decomposition method to calculate the system's Lyapunov exponents.

This thesis explores dynamical systems with known equations and uses the fourth-order Runge-Kutta method to numerically integrate in order to obtain time series

data. No experimental data from a physical dynamical system has been used in this work. However, as this method is data-driven, it can be used on experimental data where the underlying equations of the dynamical system are unknown. The method offers an approach with non-local (global) convergence to the data set. Four systems are studied in this work: the Lorenz equation, the Rossler equation, the Chua equation, the Li equation.

1.1 Dynamical Systems

Dynamical systems are the mathematics of the real world. They are used to study problems as diverse as: weather systems, traffic flow, star formation, stock market pricing, social media behavior, disease spread, population growth, chemical reactions, etc. Further reading on these topics can be found in [11][24]. Dynamical systems can be represented as discrete or continuous systems, called *maps* and *flows* respectively. This thesis only works with flows. The curious reader can read further about maps in [1]. A flow's trajectory is the path that the flow takes with the progression of time. The space in which trajectories are plotted is called the *phase space*. Since any point can serve as an initial condition for the dynamical system, the phase space is completely filled with trajectories.

A strange attractor attracts trajectories in the phase space onto them. However, trajectories are not attracted to a single point nor do trajectories on the attractor fall into periodicity. Trajectories on a strange attractor are space filling; they will visit all points within the attractor as $t \to \infty$ but they will not visit the same point twice. Strange attractors are inherently nonlinear phenomena and a dynamical system exhibiting strange attractor behavior is said to be chaotic as trajectories on the attractor meet the three requirements of chaos discussed in §1.2.

The dynamical systems studied in this work are presented in the sections below. These systems were chosen because all of them are able to exhibit chaotic behavior and some are quite famous; as such their properties are well established in the literature. Historical details about the attractors will be provided where relevant. Lorenz and Rossler are famous attractors in the field of dynamical systems and a natural starting point for this study. The Chua circuit is an application from physics. The signum switch is an attractor of personal interest to the author and was explored by *Li, et al.*[14].

1.1.1 The Lorenz Equations

In 1963, the meteorologist Edward Lorenz was running a twelve equation weather simulation when he made a peculiar discovery. Resuming his calculations from the previous day, he wanted to continue calculating where the simulation had last left off, so he input a value from the previous day into the model. In order to save space, he rounded the six decimal places down to three. He then observed that his calculations diverged over time turning into something completely unrecognizability from the previous day. Lorenz had discovered that this weather model was chaotic. In order to better understand this phenomenon, Lorenz sought a simpler system that could replicate this same behavior. Studying a system developed by Barry Saltzman in 1962 [15][23], he came up with a simplified system of three ordinary differential equations given by:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(r - z) - y \\ \dot{z} &= xy - bz \end{aligned} \tag{1.1}$$

While the equations were designed to model convection within a fluid, they also model an electric dynamo [10] and the chaotic waterwheel [10]([24], pp.310-312). When plotted in phase space, these equations produce the famous butterfly wing phase plot (Figure 1.1a), which is apropos for the colloquial name of chaos, The Butterfly Effect.

1.1.2 The Rossler Attractor

In 1976, inspired by Lorenz, Rossler created a simple model for studying chaos [22], the Rossler attractor (Figure 1.1b). The system is defined as:

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{aligned} \tag{1.2}$$

The system has only one equilibrium point and two of the equations involved are linear, making the system's eigenvalues easier to find.

1.1.3 The Chua Circuit

The Chua circuit is the simplest circuit that is capable of exhibiting chaotic behavior [2]. The circuit was developed by Leon Chua in 1983. The purpose for developing the circuit was two-fold. First, researchers wanted to realistically model the Lorenz equations in order to demonstrate chaos as a robust physical phenomenon and not the result

of numerical error [2]. The second motivation was to show that the Lorenz equations were chaotic in a rigorous mathematical sense [2]. This work uses the Chua equations as presented in *Meador* [18]:

$$\begin{aligned} \dot{x} &= \alpha(y - x - f(x)) \\ \dot{y} &= x - y + z \\ \dot{z} &= -\beta y \end{aligned} \tag{1.3}$$

$$f(x) = \begin{cases} bx - a + b & \text{for} \quad x \geq 1 \\ ax & \text{for} \quad -1 < x < 1 \\ bx + a - b & \text{for} \quad x \leq 1 \end{cases} \tag{1.4}$$

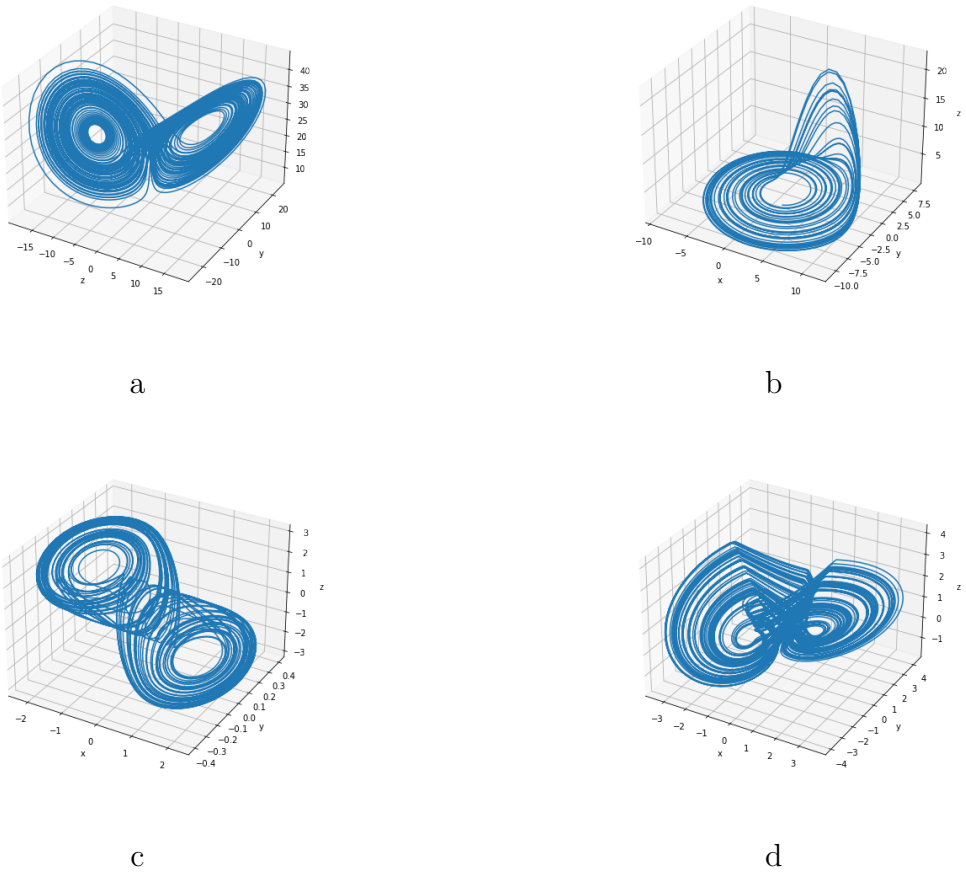
The Chua circuit was successful in its goals. In 1984, the following year, Matsumoto was able to numerically confirm the existence of chaotic attractors on the circuit [2][16]. In 1985, Zhong and Ayrom experimentally observed chaotic attractors [2][5] and in 1986 *Chua et al.* rigorously proved chaos for the equations [2].

1.1.4 The Signum "Li" Switch

The signum switch is a circuit that was contructed and studied by *Li et al.*[14] Mathematically, the dynamical system is concise:

$$\begin{aligned} \dot{x} &= y - x \\ \dot{y} &= -z \ \mathrm{sgn}(x) \\ \dot{z} &= x \ \mathrm{sgn}(y) - a \end{aligned} \tag{1.5}$$

The equations are derived from a dimensionless form of the Lorenz equations. Like Chua, the attractor has a piecewise defined element. Piecewise attractors are common for circuits due to their simplicity to construct [14]. *Li et al.* presents a four-dimensional version of this attractor in their work. This work only looks at the three-dimensional version of the attractor. This attractor has seen some interest in the music world as the company Nonlinearcircuits created a module for use in modular synthesizers based on the work in *Li et al.*



a b

c d

Figure 1.1. a. Lorenz Attractor for $\sigma = 10,\ r = 28,\ b = \frac{8}{3}$, b. Rossler Attractor for $a = 0.2,\ b = 0.2,\ c = 5.7$, c. Chua Attractor for $\alpha = 9,\ \beta = \frac{100}{7},\ a = -\frac{8}{7},\ b = -\frac{5}{7}$, d. Li Attractor for $a = 1$

1.2 Chaos

Chaos is the synthesis of the dialectic between, in a poetic sense, fate and free-will, between predetermination and an unknowable future. If one knows the initial state of a *dynamical system*—a system that changes over time—with infinite precision, one is able to predict the time evolution of that system exactly from now until the heat death of the universe. However, in a chaotic system any infinitesimal perturbation in the measurement of this initial state will lead to drastically different futures for the perturbed system.

While no universally accepted definition of chaos exists ([24], pp.331), there are agreed upon characteristics of chaotic systems:

1. The system must exhibit **aperiodic long-term behavior**. Trajectories never settle down to fixed points, periodic orbits, or quasiperiodic orbits as $t \to \infty$.

2. The system is **deterministic**. Irregular behavior arises from the system's nonlinearity rather than any noisy driving forces. Generally speaking, there are no random or noisy input parameters.
3. The system exhibits **sensitive dependence on initial conditions**. Nearby trajectories seperate from each other exponentially fast.

1.3 Lyapunov Exponents

The motion on strange attractors exhibit sensitive dependence on initial conditions (see §1.2). Lyapunov exponents are a measure of the exponential rates of divergence of neighboring trajectories on an attractor [21]. To explore the concept more thoroughly consider a dynamical system:

$$\frac{d}{dt}\mathbf{y} = \mathbf{f}(\mathbf{y}),\ \mathbf{y}(0) = \mathbf{x} \in \mathbb{R}^n \tag{1.6}$$

The affiliated flow map via the function is given by $\phi_t(\mathbf{x})$ so that $\mathbf{y}(t;\mathbf{x}) = \phi_t(\mathbf{x})$. $\mathbf{f}(\mathbf{y})$ is called the velocity field (or vector field) of the flow. In this work $\mathbf{f}(\mathbf{y})$ will be referred to as the *unknown function*. The unknown function corresponds to a data representation of the underlying differential equation expressions of the dynamical system. We will be interested in calculating the unknown function in order to generate numerical Jacobian approximations. To find the solution of $\mathbf{y}(t;\mathbf{x})$ we first compute the Jacobian $J(t)$ where

$$J(t) = D_y\mathbf{f}(\mathbf{y}(t;\mathbf{x})) \tag{1.7}$$

An affilated time-dependent linear system can then be defined as

$$\frac{d\mathbf{U}}{dt} = J(t)\mathbf{U},\ (0) = \mathbf{I} \tag{1.8}$$

Then the Lyapunov exponents are the eigenvalues of the matrix Λ defined as

$$\Lambda = \lim_{t\to\infty} \frac{1}{2t}\log\mathbf{U}(t)\mathbf{U}^T(t) \tag{1.9}$$

Lyapunov exponents give us a measure of how predictable the dynamical system is and a measure of entropy on the attractor. The presence of a positive Lyapunov exponent indicates sensitive dependence on initial conditions and is the telltale sign of chaos. However, dynamical systems can also have negative Lyapunov exponents which indicate no or contracting growth.

Remark 1.1. *Exponential divergence of trajectories must stop when seperation distances are comparable to the diameter of the attractor ([24], pp.330). This phenomenon is called boundedness and is an important concept to observe for chaotic systems. Dynamical systems such as $\dot{x} = ax$ also exhibit sensitive dependance on initial conditions (and have positive Lyapunov exponents). However, the aforementioned dynamical system is unbounded, nearby trajectories seperate from each other exponentially fast and become arbitarily distant from each other as $t \to \infty$. This latter property is not shared by chaotic attractors.*

Lyapunov exponents are often difficult to compute analytically and thus are commonly solved with numerical methods. Several methods have been developed for numerically approximating Lyapunov exponents and are explored in the literature. Techniques for the calculation of exponents can either be done from analytic Jacobian calculations or be data driven. *Eckmann et al.* develop a method for determining Lyapunov exponents from time series data with the use of Taken's Embedding Theorem [7]. *Wolf* introduces a method that utilizes Gram-Schmidt reorthonormalization [25]. *Deici et al.* uses QR-decomposition of the Jacobian of the dynamical system to estimate the Lyapunov exponents [6], *Arbarbanel et al.* utilizes a similar OR-decomposition method but introduces a data driven method that uses Taylor series to approximate the Jacobian [12]. This work also utilizes QR-decomposition to calculate Lyapunov exponents, details can be found in §2.1

1.4 Meshfree Scattered Data Interpolation using Radial Basis Functions

Mesh generation remains one of the most time consuming aspects of mesh-based numerical simulations such as: finite differences, finite elements, or finite volumes ([8], pp.1). These costs can be eliminated with meshfree methods as they rely on sets of independent points. Traditional numerical methods are often only suitable for low-dimensional systems ([8], pp.1). However, many real-world problems in science and engineering are high-dimensional problems that can often involve hundreds or thousands of variables. As such, applications for meshfree methods can be found in many different fields of study ranging from differential equations to machine learning.

The *Gaussian* is a good first example of a *basic function* to begin our discussion of radial basis functions (RBFs):

$$\phi(r;\epsilon) = e^{-(\alpha r)^2} \tag{1.10}$$

where α represents the *shape parameter* (see §2.2). RBFs are radially symmetric about their center and the shape parameter determines the flatness of the function. In order to build up a rigorous definition let

$$\gamma_k(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_k\|_2 \tag{1.11}$$

Then by $\phi(\gamma_k(\mathbf{x}))$ one can obtain for any fixed center $\mathbf{x}_k \in \mathbb{R}^s$ ([8], pp.17) a multivariate function

$$\Phi_k(\mathbf{x}; \epsilon) = \phi(\gamma_k(\mathbf{x})) = e^{-\alpha^2 \|\mathbf{x}-\mathbf{x}_k\|_2^2}, \ \mathbf{x} \in \mathbb{R}^s \tag{1.12}$$

where s is the dimension of the data space. This leads to a rigorous definition of radial function:

Definition 1.2. *A function $\Phi : \mathbb{R}^s \to \mathbb{R}$ is called radial provided there exists a univariate function $\phi : [0, \infty) \to \mathbb{R}$ such that*

$$\Phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$$

and $\|\cdot\|$ is some norm on $\mathbb{R}^s$ – usually the Euclidean norm ([8], pp.17).

When $\phi(r)$ in definition 1.2 is a basic function, we call it a radial basis function (RBF). The Φ_k presented is a distance matrix. Using a radial basis function expansion one is able to solve scattered data interpolation problems on $\mathbb{R}^s$ by assuming

$$\mathcal{P}_f(\mathbf{x}) = \sum_{k=1}^{N} c_k \phi(\gamma_k(\mathbf{x})) \tag{1.13}$$

coefficients c_k are found by solving the linear system

$$\begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|_2) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_N\|_2) \\ \vdots & \ddots & \vdots \\ \phi(\|\mathbf{x}_N - \mathbf{x}_1\|_2) & \cdots & \phi(\|\mathbf{x}_N - \mathbf{x}_N\|_2) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} g(\mathbf{x}_1) \\ \vdots \\ g(\mathbf{x}_N) \end{bmatrix} \tag{1.14}$$

Where $g(\mathbf{x})$ represents a general test function. In this thesis we used $g(\mathbf{x}) = \text{sinc}(\omega x_1)\text{sinc}(\omega x_2)\ldots\text{sinc}(\omega x_N)$ as the test function. The work presented tests the scheme with several different RBFs. The basic functions used in the interpolator are presented in Table 1.1.

Name	ϕ(r)
Gaussian	$e^{-\frac{r^2}{\epsilon^2}},\ \epsilon > 0$
Multiquadric	$(r^2+\epsilon^2)^{1/2},\ \epsilon \geq 0$
Inverse	$(r^2+\epsilon^2)^{-1/2},\ \epsilon \geq 0$

Table 1.1. Basic functions

CHAPTER 2

Methods

The following section details how the MSDI-RBF algorithm works and the conditions under which the algorithm was tested. There are several components to to MSDI-RBF algorithm and the following subsections will discuss each. The algorithm relies on QR-decomposition of the Jacobian to calculate Lyapunov exponents. As MSDI-RBF is data driven, we cannot rely on analytic methods to find the Jacobian. Therefore, we construct an interpolation matrix containing basic functions which take in as arguments time series data and a shape parameter ϵ. The shape parameter must be optimized to find a best fit to the data in order to obtain the most accurate results. Linear algebraic operations are then performed to calculate the Jacobian.

2.1 Lyapunov Exponent Estimation

2.1.1 QR-Decomposition

In this work an algorithm developed by *Deici et al.* [6] is used in order to check the results of the MSDI-RBF. Deici's algorithm uses QR-decomposition of a dynamical system's Jacobian matrix in order to calculate the Lyapunov exponents. The method is not data driven and relies on calculations utilizing the exact Jacobian of the system computed from the dynamical system's differential equations. The work of this thesis also uses the QR-decomposition of the Jacobian. However, we use time series data fed into the MSDI-RBF algorithm in order to approximate the Jacobian matrix rather than doing calculations with the analytic Jacobian. A brief overview of the QR-algorithm developed by Deici is presented in this section. However, see *Deici et al.* [6] for a more comprehensive discussion.

As an instructive example, consider the Lorenz system from equation (1.1). In this work the following values were chosen for the parameters of the Lorenz system: $\sigma = 10,\ r = 28,\ b = \frac{8}{3}$. For a generic initial condition a general solution can be described by the flow map $\mathbf{y}(t) = \phi(t; \mathbf{x})$ where $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ represents the initial condition. When numerically integrating using the 4th-order Runge-Kutta scheme, one obtains the

famous Lorenz butterfly (Figure 1.1a).

For a given solution of (1.1) one can linearize the system (1.8) by computing the affiliated Jacobian, where

$$J(t) = \begin{bmatrix} -\sigma & \sigma & 0 \\ r - z(t) & -1 & -x(t) \\ y(t) & x(t) & -b \end{bmatrix} \tag{2.1}$$

A approximation to the solution of (1.1) is defined at discrete times $\{t_j\}_{j=0}^{N_T}$, $t = 0$ and $t_j = j\delta t$. To generate an approximation to the Lyapunov exponents we solve

$$\frac{d}{dt}\mathbf{U}_j = J(t)\mathbf{U}_j, \;\; \mathbf{U}_j(t_j) = \mathbf{Q}_j \tag{2.2}$$

where $\mathbf{Q}_0 = \mathbf{I}$ and we perform a QR-decomposition such that

$$\mathbf{U}_j(t_{j+1}) = \mathbf{Q}_{j+1}\mathbf{R}_{j+1} \tag{2.3}$$

The m^{th} Lyapunov exponent, λ_m is then given by

$$\lambda_m = \lim_{j\to\infty} \frac{1}{t_j} \sum_{k=1}^{j} \log |(\mathbf{R}_k)_{mm}| \tag{2.4}$$

2.1.2 Computing Lyapunov exponents from data

Deici et al.'s [6] method is not a data driven method. It relies on the Jacobian obtained by linearizing the equations of a dynamical system equations. The knowledge of the underlying equations of a given dynamical system is a luxury not often had with real world dynamical systems. Instead, what is the researcher has are data points collected from an experimental system. Therefore, proposed in this work is a data driven method for calculating the Lyapunov exponents of a dynamical system.

In *Abarbanel et al.* [12] the aforementioned problem is dealt with using least squares fitting to Taylor series expansions. This method is algorithmically unwieldy. Additionally, the accuracy of Taylor series expansions rely on proximity arguments which cannot be guaranteed to hold. Our method, MSDI-RBF, generates non-local (global) approximations to the map at each time step using nearest neighbors to a given point $\mathbf{y}(t_j) \equiv \mathbf{y_j}$. However, from [12] our algorithm inherits

1. Use of a KD-tree algorithm. We find a list of the nearest neighbors N_{nb}, to the point we wish to approximate $\mathbf{y}(t_j)$. We label the l^{th} nearest neighbor of $\mathbf{y}_j$ as $\mathbf{y}^n_{l,j}$. While geometrically close to $\mathbf{y}_j$, $\mathbf{y}^n_{l,j}$ could correspond to a different time t_k. This set of points $\mathbf{y}_j \cup \{\mathbf{y}^n_{l,j}\}_{l=1}^{N_{nb}}$ is denoted as $\mathcal{I}_j$
2. We follow each nearest neighbor one time step, δt, forward in time generating the points $\mathbf{y}_{j+1}$. Additionally, we generate $\mathbf{y}^{n,+}_{l,j} = \phi_{\delta t}(\mathbf{y}^n_{l,j})$ and $\mathbf{y}^{n,-}_{l,j} = \phi_{-\delta t}(\mathbf{y}^n_{l,j})$
3. Using second-order centered-finite differencing we obtain an approximation for the unknown function $\mathbf{f}(\mathbf{y})$

$$\mathbf{f}_j = \frac{\mathbf{y}_{j+1} - \mathbf{y}_{j-1}}{2\delta t}, \ \mathbf{f}^n_{l,j} = \frac{\mathbf{y}^{n,+}_{l,j} - \mathbf{y}^{n,-}_{l,j}}{2\delta t}$$

2.1.3 Calculating the Jacobian

From this point our method diverges from [12]. We use RBFs to build interpolatory aprroximations for the points (y_j, f_j) and $\{(y^n_{l,j}, f^n_{l,j})\}$. We can then differentiate to find approximations of $J(t)$. This section explores the algorithm used to calculate the approximated Jacobian $\tilde{J}(t)$. As stated prior, MSDI-RBF converges globally and can be used to generate analytic approximations to the map at each time step using the nearest-neighbors to a given point in $\mathbf{y}(t_j)$. The method produces a sequence of approximate Jacobians, $\tilde{J}_j = \tilde{J}(t_j)$. From time-step to time-step, we have the analytic formula for the matricies $\mathbf{U}_j$

$$\mathbf{U}_j(t_{j+1}) = \mathbf{Q}_j + \int_{t_j}^{t_{j+1}} \tilde{J}(t)\mathbf{U}_j(t)dt \tag{2.5}$$

The Trapezoid approximation scheme is used to get a stable approximation of (2.5)

$$\left(\mathbf{I} - \frac{\delta t}{2}\tilde{J}_{j+1}\right)\mathbf{U}_j\left(t_{j+1}\right) = \left(\mathbf{I} - \frac{\delta t}{2}\tilde{J}_j\right)\mathbf{Q}_j \tag{2.6}$$

In order to avoid transients, the first $\frac{50}{\delta t}$ points from the data set $\{\mathbf{y}_j\}_{j=0}^N$ are removed to produce the reduced data set $\mathbf{y}_{\text{red}} = \{\mathbf{y}_j\}_{j=50/\delta t}^N$. A list of the neartest 50 neighbors $\mathbf{y}^n_{l,j}$ to every point on the trajectory $\mathbf{y}_{\text{red}}$ is then generated. The data from $\mathbf{y}_{\text{red}}$ is placed into the distance matrix Φ represented by (1.14) and an RBF is chosen to represent $\phi(r, \epsilon)$. The interpolation problem is then solved for $\mathbf{c} = [c_1, ..., c_N]^T$.

Using our approximations $\mathbf{f}_j$ and $\{\mathbf{f}^n_{l,j}\}_{l=1}^{N_{nb}}$ over interpolation points $\mathbf{y}_{\text{red}}$ and $\{\mathbf{y}^n_{l,j}\}_{l=1}^{N_{nb}}$ we find the RBF approximation $\mathbf{f}_m$ via

$$\mathbf{f}_m(\mathbf{y}) = \sum_{l=0}^{N_{nb}} c_{ml}\phi(\|\mathbf{y} - \mathbf{y}^n_{l,j}\|_2) \tag{2.7}$$

where $\mathbf{y}^n_{0,j} = \mathbf{y}_j$ and $m = \{1, ..., n\}$. We find the approximation to the Jacobian $D_{\mathbf{y}}\mathbf{f}(\mathbf{y})|_{\mathbf{y}=\mathbf{y}_j}$ by finding gradients

$$\nabla_{\mathbf{y}}\mathbf{f}(\mathbf{y})|_{\mathbf{y}=\mathbf{y}_j} = \sum_{l=1}^{N_{nb}} c_{m,l}\phi'(\|\mathbf{y} - \mathbf{y}^n_{l,j}\|_2)\frac{\mathbf{y} - \mathbf{y}^n_{l,j}}{\|\mathbf{y} - \mathbf{y}^n_{l,j}\|_2} \tag{2.8}$$

using (2.8) we are now able to perform QR-decomposition with (2.3) and subsequently calculate the estimates of the affiliated Lyapunov exponents using (2.4).

2.2 Shape Parameter

The accuracy of many RBF schemes is dependent on a shape parameter ϵ (see Table 1.1). The ϵ value controls the steepness of the RBF. Generally when making approximations on data, the RBF should be made as flat as possible. However, nearly flat RBFs often cause the interpolation matrix to become near-singular. There is often an optimal ϵ for a corresponsing RBF on a given data set and this optimal ϵ is often found where in areas where the condition number is large.

There are several different methods for chosing an optimal shape parameter for an RBF. In this work an algorithm developed by *Rippa* ([8], pp.146)[21]. Other methods presented are by Faushauer ([8], pp.142-153) but will not be discussed in any detail in this work. Before discussing the optimization of ϵ however, we first look at the conditioning of matricies.

2.2.1 Condition Number

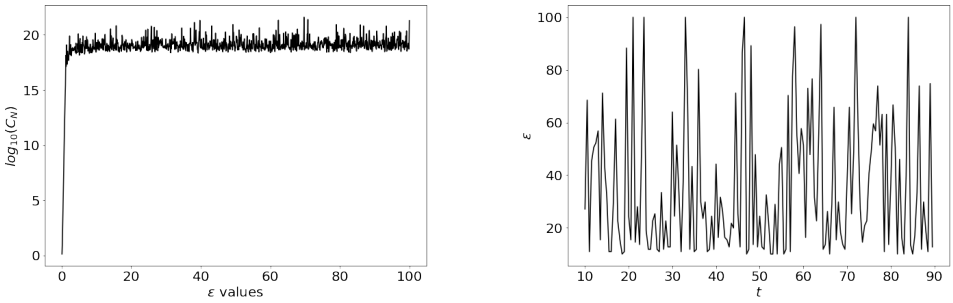
A matrix $\mathbf{A}$ is said to be singular if the det($\mathbf{A}$)=0. Let $\mathbf{B}$ be an invertible matrix where det($\mathbf{B}$)= $0 + \eta$ where $0 < \eta << 1$. $\mathbf{B}$ is then said to be *ill-conditioned*. These matricies are said to be *near-singular*, meaning that small perturbations in the matrix elements can cause the matrix to become singular ([13], pp.114). Within numerical linear algebra, Ill-conditioned matricies can produce inaccurate results due to roundoff errors. However, it is not necessarily the case that results obtained from ill-conditioned matricies are inaccurate. However, the results cannot be trusted on their face without

independent verification.

The *condition number* of a matrix is a measure of how ill-conditioned a matrix is and is given by

$$\kappa(\mathbf{B}) = \|\mathbf{B}\|\|\mathbf{B}^{-1}\| \tag{2.9}$$

Singular matricies have an infinite condition number thus the larger the condition number of a matrix, the more ill-conditioned the matrix is. In trying to optimize ϵ to obtain accurate results for our approximations, we face a contridiction: for large data sets, the optimal ϵ is often found for parameters where the interpolation matrix Φ is severely ill-conditioned (Figure 2.1). Thus, there is no guarantee that the scheme is stable for a given data set. This feature is what is called the *trade-off principle* ([8], pp.135-140).



a b

Figure 2.1. a. Condition number vs. ϵ plot for Lorenz equation b. Optimal shape parameter measured against time for Lorenz equation

Faushauer identifies three trade-off principles

1. **Accuracy vs. Stability**. The standard approach to solving RBF interpolation problems has a conflict between theoretically achievable accuracy and numerical stability. If we use ϵ to (exponentially) increase accuracy then the condition number also grows exponentially.
2. **Accuracy and Stability vs. Problem Size**. The Contour-Pade algorithm solves the first trade-off principle and is able to find an optimal ϵ without sacrificing accuracy or stability. However, it comes at the expense of ability to deal with large data sets. The details of this algorithm will not be discussed any further. The interested reader may refer to ([8], pp.140)
3. **Accuracy vs. Efficiency**. This trade-off principle is associated with compactly supported functions which will not be discussed further. Interested readers may refer to ([8], pp.140)

In the following subsections, we discuss the method used in this thesis for finding an optimal ϵ to interpolate the attractor data using MSDI-RBF.

2.2.2 Trial and Error

The simplest method to choose a good shape parameter (ϵ) is to run many experiments with different shape parameters and select the best one. This strategy can be effective if the underlying function generating the data is known since an error function can be generated ([8], pp.142). Without knowing the underlying function, it becomes difficult to determine what the best shape parameter value is. Additionally, if the data generating function is known, then the exercise of interpolating is pointless outside of academic examples.

In this work—and in study of dynamical systems generally—the systems are capable of exhibiting chaotic behavior. The function of a chaotic attractor is almost never known. Additionally, underlying functions of real-world systems are also rarely known leaving this approach unsuitable for the our work.

2.2.3 LOOCV "Drop-One" Algorithm

Rippa [21] develops an algorithm to find an optimal ϵ value. This algorithm is used in this work. This algorithm works by taking a single data-point out of the data set, or dropping one, and interpolating to generate a point for the dropped data point. Then we perform a least-squares error fit on the dropped data set, comparing the error of the interpolated point in the 'drop one' data set and the known point in the original data set ([8], pp.146). This algorithm is presented in this section. Rippa'a algorithm will be referred to as leave-one-out-cross-validation (LOOCV)

To examine how this scheme works, consider the radial basis function interpolant $\mathcal{P}_f^{[k]}$ to the data $\{f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_N\}$. From (1.13) one arrives at interpolation problem

$$\mathcal{P}_f^{[k]}(x) = \sum_{j=1}^{N} c_j^{[k]} \phi(\gamma_k(x)), \ j \neq k \tag{2.10}$$

where $\mathcal{P}_f^{[k]}(x_i) = f_i$ and $i = \{1, \ldots, k-1, k+1, \ldots, N\}$. E_k represents the error between the original data set and the interpolated dropped point data set. E_k is given by

$$E_k = f_k - \mathcal{P}_f^{[k]}(x_k) \tag{2.11}$$

This implementation of the algorithm using (2.11) is expensive ([8], pp.146). However, E_k can be simplified into

$$E_k = \frac{c_k}{\Phi_{kk}^{-1}} \tag{2.12}$$

with c_k representing the kth coefficient of $\mathcal{P}_f$. $\Phi_{kk^{-1}}$ represents the kth diagonal element of the inverse interpolation matrix. The derivation for this formula is given in [21]. The quality of the fit is obtained from the norm of the error vectors $\mathbf{E} = [E_1, \ldots, E_N]^T$ obtained by removing one point from the original data and comparing the fit with the known removed value at the removed point ([8], pp.146). By looping over the shape parameter ϵ, we can compare the error norms for different values of ϵ and choose the ϵ associated with the smallest error norm.

The LOOCV algorithm has the advantage of not needing to know the function that generated the data, it works for large data sets, and its error bounds do not depend on the basic function. For this reason, the LOOCV method is a good algorithm for selecting an optimal ϵ and thus is used in this work.

2.3 Fill Distance

The attractors in this thesis have different geometries. To get a sense of how MSDI-RBF distributes data on the attractor, the fill distance is calculated. The fill distance can be understood as the pairwise distance between a set of evaluation points and the data set $\mathbf{y}$ and is given by

$$h = h_{\chi,\Omega} = \sup_{\mathbf{y}\in\Omega} \min_{\mathbf{y}_j\in\chi} \|\mathbf{y} - \mathbf{y}_j\| \tag{2.13}$$

where χ is the data set and Ω is the domain. Faushauer creates a uniform grid in $\mathbb{R}^2$ to create the grid to measure the fill distance ([8], pp.22). The data set worked with in ([8], pp.22) fills $\mathbb{R}^2$ as a scattered data set. This work is interested in spacing on the attractor and not all space in $\mathbb{R}^3$ so the same attractor calculated with a finer time step or longer simualtion time length is used in order to calculate the fill distances on the attractor. This thesis calculates fill distances as the trajectories move across the attractor. The fill distances calculated and plotted as a function of time.

CHAPTER 3

Results

Presented in this section are the results of different experiments done with the MSDI-RBF. In this work, all of the dynamical systems were solved using the 4th-order Runge-Kutta method. The Lorenz and Rossler equations are used to stress test MSDI-RBF. We explore the effect that the time length of the simulation, t, has on the accuracy of MSDI-RBF approximations, the effect of the time step, dt, for the Runge-Kutta scheme, and the effect of the number of nearest neighbors N_{nb}. Additionally, final Lyapunov estimations from the MSDI-RBF scheme will be presented for the Chua and Li attractors. However, Chua and Li will not be used to explore the aforementioned parameter experiments. For all of the attractors we study the effect that the choice of RBF has on the accuracy of our approximations.

The initial condition used for all systems was $\mathbf{x}_0 \equiv (x, y, z) = (0, 1, 0)$. We tested Lorenz using its canonical parameter values $\sigma = 28,\ r = 40,\ b = \frac{8}{3}$. We tested Rossler using parameter values $a = 0.2, b = 0.2, c = 5.7$. Chua tested using values $\alpha = 9, \beta = \frac{100}{7}, a = -\frac{8}{7}, b = -\frac{5}{7}$ and Li used the parameter $a = 1$. In general, the simulation run time $t = 1000$, time step $dt = 0.01$, nearest neighbors $N_{nb} = 50$, and a Gaussian basic function were used as default simulation parameters which all other experiments were measured against. It should be assumed that the aforementioned simulation parameters are used unless otherwise specified. Optimal ϵ values were tested in the range (0,100]. The trace of the Jacobian was calculated for all systems where $tr(J(t)) = \lambda_1 + \lambda_2 + \lambda_3$.

The algorithm from [6] was used to establish a baseline from which to measure the accuracy and precision of the MSDI-RBF algorithm and results are presented in Table 3.1.

3.1 Lorenz Equation

We tracked the Jacobian error measured across the entire time series $\mathbf{y}$ (Figures 3.1, 3.3, and A.1) and the affiliated optimal shape parameter as we move around the

Attractor	t	dt	λ_1	λ_2	λ_3	$tr(J(t))$
Lorenz	90	0.01	0.715488	0.0346102	-14.4151	-13.6650
Lorenz	500	0.01	0.869117	0.0078071	-14.5432	-13.6663
Lorenz	1000	0.01	0.889032	0.0032922	-14.5587	-13.6664
Lorenz	1000	0.1	0.465994	0.172585	-11.4328	-10.7942
Lorenz	1000	0.05	0.886379	0.014464	-14.4606	-13.5598
Lorenz	1000	0.005	0.881813	0.00477522	-14.5532	-13.6666
Lorenz	1000	0.001	0.891902	0.00378951	-14.5623	-13.6666
Rossler	180	0.01	0.0665055	0.0193292	-5.47756	-5.3917
Rossler	1000	0.01	0.071127	0.0034966	-5.41067	-5.3360
Rossler	2000	0.01	0.0716397	0.00127227	-5.39827	-5.3254
Rossler	1000	0.1	0.0702216	0.0034646	-5.24156	-5.3370
Rossler	1000	0.05	0.0718241	0.00270556	-5.39934	-5.3248
Rossler	1000	0.005	0.0699829	0.00296756	-5.40376	-5.3308
Rossler	1000	0.001	0.0723338	0.00249077	-5.40338	-5.3288
Chua	1000	0.01	0.32169	0.00143785	-2.56737	-2.2442
Li	1000	0.01	0.146263	0.00205924	-1.14831	-1

Table 3.1. Deici algorithm results

attractor (Figure 3.2). Jacobian errors were calculating using the equation

$$\frac{\|\tilde{J}(t) - J(t)\|}{\|J(t)\|} \tag{3.1}$$

Where $\tilde{J}(t)$ represents the approximated Jacobian, $J(t)$ respresents the exact Jacobian calculation at time t, and $\|\cdot\|_2$ is the Frobenius norm. The Lorenz equation was tested with simulation lengths $t = 90$ and $t = 1000$. Time-steps $dt = 0.01$ and $dt = 0.001$ were used in calculation for both simulations lengths. Additionally, we calculated Jacobian error estimates for, $t = 500$ for $dt = 0.01$, and $dt = 0.1,\ 0.05,\ 0.005,\ 0.001$ for $t = 1000$ (Figure A.1). Lyapunov estimations were made for the aforementioned parameters as well as number of nearest neighbors and multiquadric and inverse basic functions at $t = 1000$ and $dt = 0.01$ (Figure 3.3).

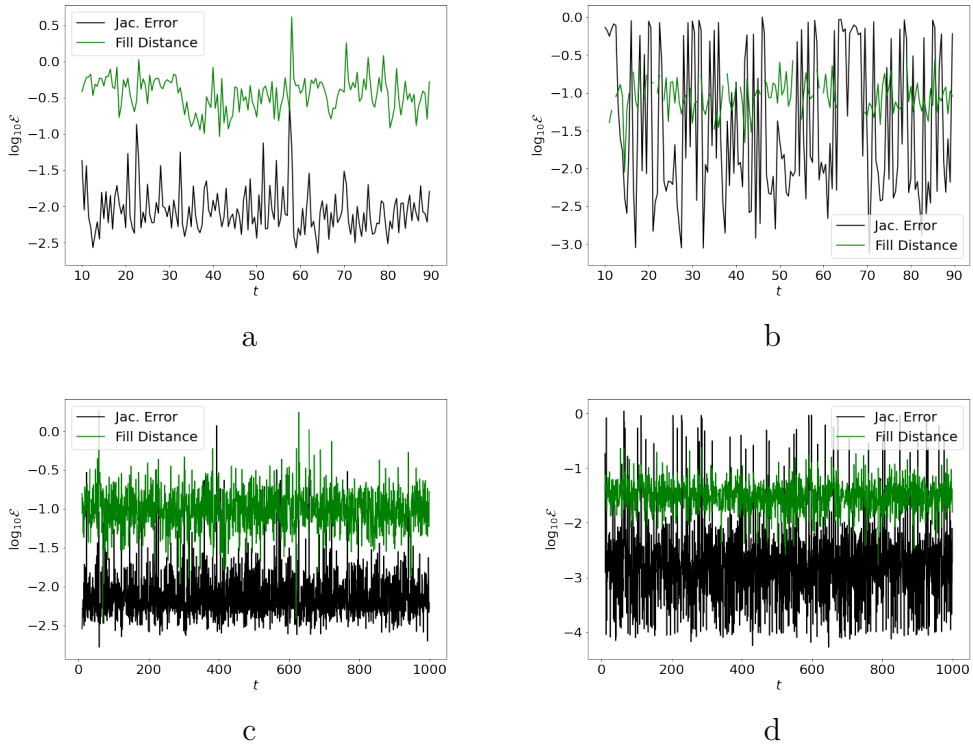


Figure 3.1. Lorenz Jacobian Error and Fill Distance, a. $t = 90$, $dt = 0.01$ **b.** $t = 90$, $dt = 0.001$ **c.** $t = 1000$, $dt = 0.01$ **d.** $t = 1000$, $dt = 0.001$

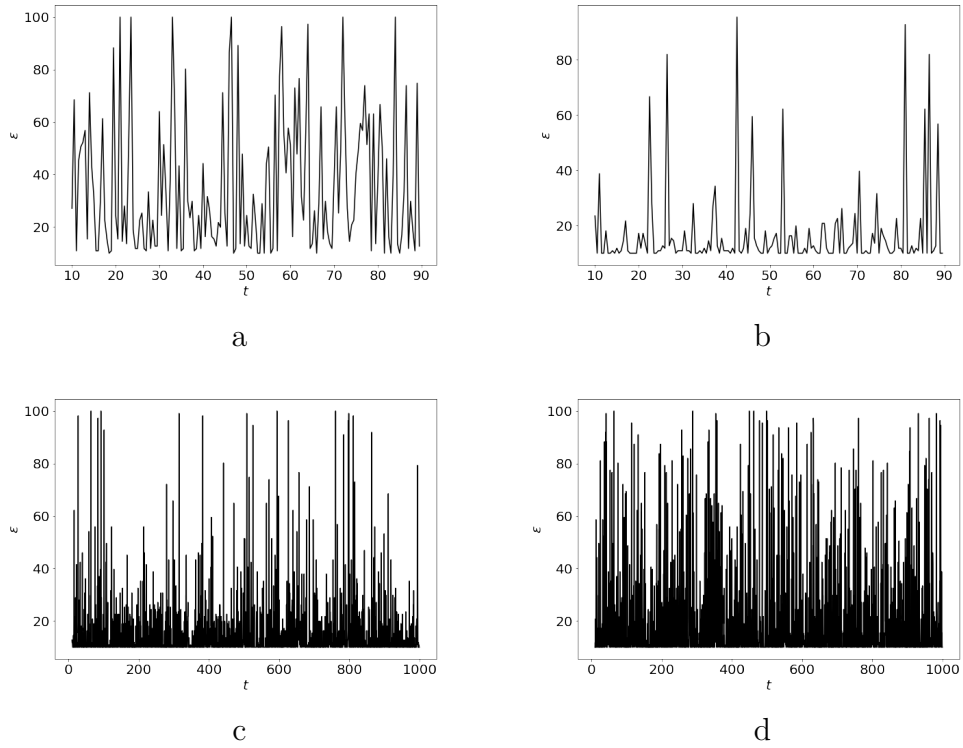


Figure 3.2. Lorenz shape parameter vs. time, a. $t = 90$, $dt = 0.01$ **b.** $t = 90$, $dt = 0.001$ **c.** $t = 1000$, $dt = 0.01$ **d.** $t = 1000$, $dt = 0.001$

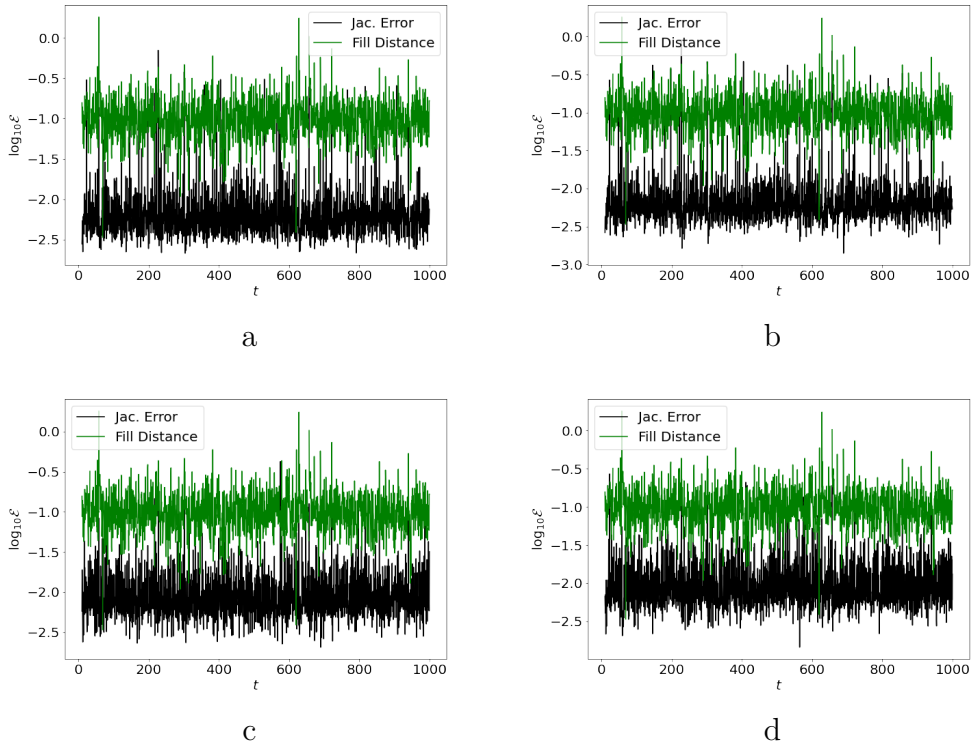


Figure 3.3. Lorenz Jacobian Error and Fill Distance, a. multiquadric b. inverse c. $N_{nb} = 100$ **d.** $N_{nb} = 150$

$\phi(x,\epsilon)$	t	dt	N_{nb}	λ_1	λ_2	λ_3	$tr(\tilde{J}(t))$
gaussian	90	0.01	50	0.647603	0.108388	-12.8713	-12.1153
gaussian	90	0.001	50	0.776243	-0.0293548	-8.41689	-7.6700
gaussian	500	0.01	50	0.755684	0.124363	-14.3492	-13.4692
gaussian	1000	0.01	50	0.76659	0.129778	-14.5085	-13.6121
multiquadric	1000	0.01	50	0.770113	0.125643	-14.5009	-13.6051
inverse	1000	0.01	50	0.771143	0.124422	-14.5006	-13.6050
gaussian	1000	0.01	100	0.763427	0.130108	-14.5238	-13.6303
gaussian	1000	0.01	150	0.764278	0.131224	-14.5234	-13.6279
gaussian	1000	0.1	50	2.87036	-1.49536	-20.4797	-19.1047
gaussian	1000	0.05	50	1.44894	-0.465272	-17.208	-16.2243
gaussian	1000	0.005	50	0.847037	0.0399775	-14.3954	-13.5084
gaussian	1000	0.001	50	0.895842	0.000249988	-14.1642	-13.2681

Table 3.2. MSDI-RBF Lyapunov Estimates for Lorenz equation

3.2 Rossler Equation

We tested Rossler equation under the same parameter values as the Lorenz equation with one exception, the simulation run time of the Rossler tests were doubled from what was tested in Lorenz. However, the same procedure is followed. The Jacobian errors and fill distances measured against time are presented in Figure 3.4, 3.6, and A.2 and optimal shape parameters in Figure 3.5. The Lyapunov exponent estimations are presented in Table 3.3.

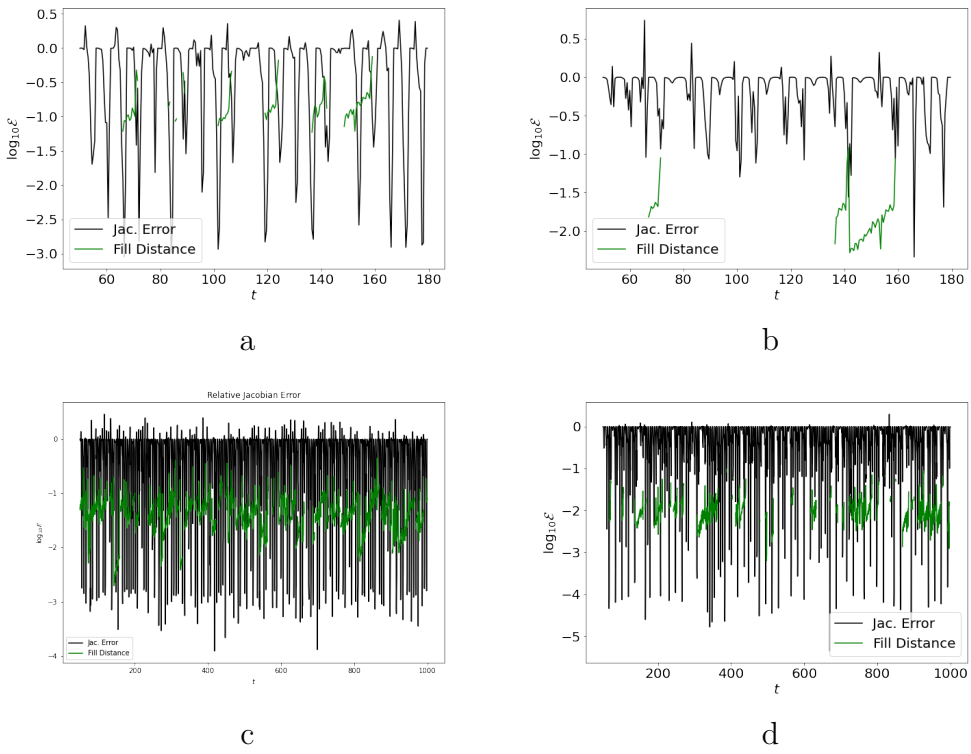


Figure 3.4. Rossler Jacobian Error and Fill Distance, a. $t = 180$, $dt = 0.01$ **b.** $t = 180$, $dt = 0.001$ **c.** $t = 1000$, $dt = 0.01$ **d.** $t = 1000$, $dt = 0.001$

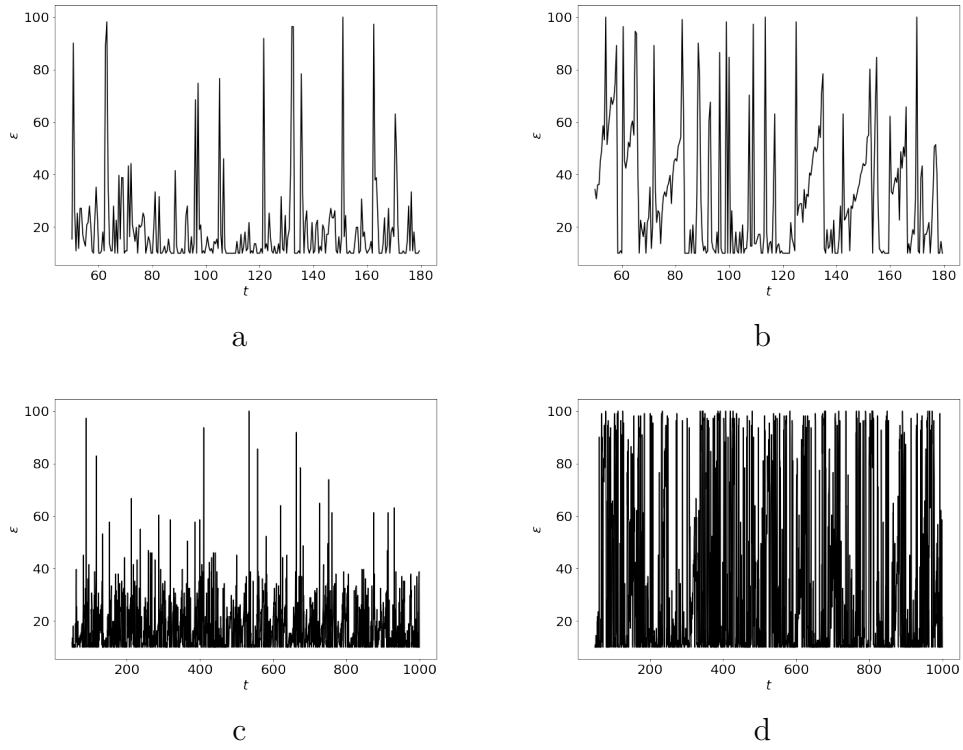


Figure 3.5. Rossler shape parameter vs time, a. $t = 180$, $dt = 0.01$ **b.** $t = 180$, $dt = 0.001$ **c.** $t = 1000$, $dt = 0.01$ **d.** $t = 1000$, $dt = 0.001$

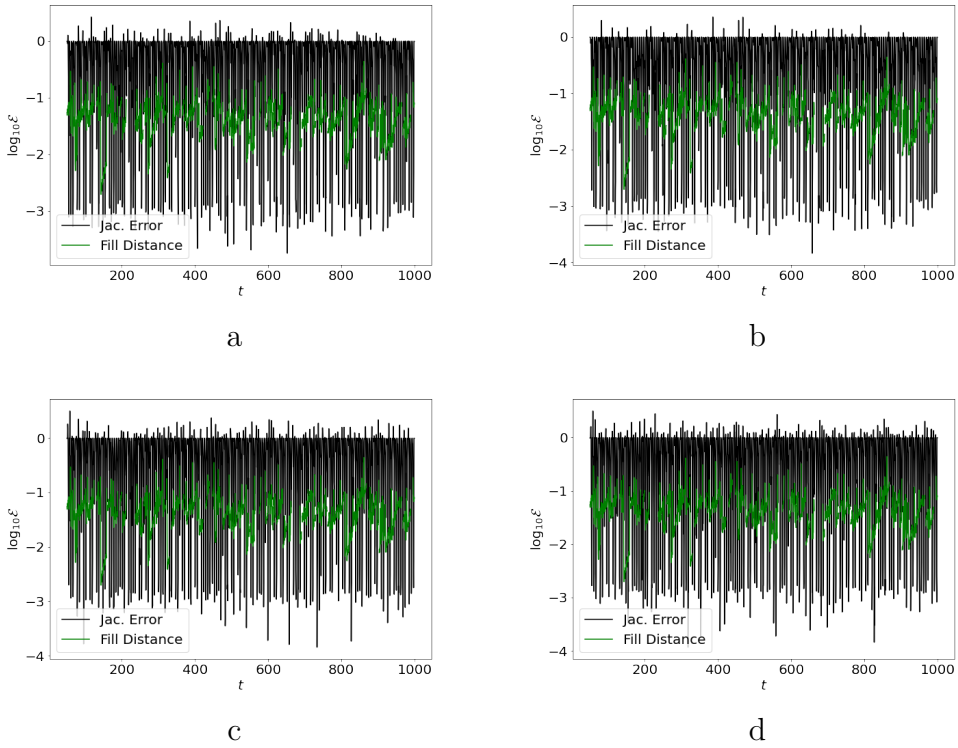


Figure 3.6. Rossler Jacobian Error and Fill Distance, a. $t = 2000$, $dt = 0.01$ **b.** $t = 1000$, $dt = 0.1$ **c.** $t = 1000$, $dt = 0.05$ **d.** $t = 1000$, $dt = 0.005$

$\phi(x, \epsilon)$	t	dt	N_{nb}	λ_1	λ_2	λ_3	$tr(\tilde{J}(t))$
gaussian	180	0.01	50	0.0323797	0.00138742	-0.91939	-0.8856
gaussian	180	0.001	50	0.0290489	0.0011676	-0.228423	-0.1982
gaussian	1000	0.01	50	0.0670752	0.000405778	-1.69032	-1.6228
gaussian	2000	0.01	50	0.069549	-0.000608606	-1.53759	-1.4686
multiquadric	1000	0.01	50	0.0670853	0.000419429	-1.6349	-1.5674
inverse	1000	0.01	50	0.0670964	0.000517378	-1.71732	-1.6498
gaussian	1000	0.01	100	0.0670358	0.000614184	-1.79644	-1.7288
gaussian	1000	0.01	150	0.0669915	0.000366926	-1.66476	-1.5974
gaussian	1000	0.1	50	0.0559489	-0.0302194	-2.59748	-2.5718
gaussian	1000	0.05	50	0.0695828	-0.00497013	-1.95738	-1.8928
gaussian	1000	0.005	50	0.0660092	-8.18815e-05	-1.44486	-1.3789
gaussian	1000	0.001	50	0.0687768	-0.000587808	-1.06623	-0.9980

Table 3.3. MSDI-RBF Lyapunov Estimates for Rossler equation

3.3 Chua and Li Attractors

The Chua equation and Li equation were tested using the three basic functions presented in this thesis. In both systems, $t = 1000$, $dt = 0.01$, and $N_{nb} = 50$. The Lyapunov exponent estimations for Chua and Li can be found in Tables 3.4 and 3.5 respectively.

$\phi(x,\epsilon)$	t	dt	N_{nb}	λ_1	λ_2	λ_3	$tr(\tilde{J}(t))$
gaussian	1000	0.01	50	0.380541	-0.0169871	-2.59346	-2.2299
multiquadric	1000	0.01	50	0.384737	-0.0261238	-2.61506	-2.2564
inverse	1000	0.01	50	0.412058	0.000260987	-2.58251	-2.1702

Table 3.4. MSDI-RBF Lyapunov Estimates for Chua equation

$\phi(x,\epsilon)$	t	dt	N_{nb}	λ_1	λ_2	λ_3	$tr(\tilde{J}(t))$
gaussian	1000	0.01	50	0.565762	-0.0335047	-1.45285	-0.9206
multiquadric	1000	0.01	50	0.536935	-0.0137958	-1.45266	-0.9295
inverse	1000	0.01	50	0.515629	0.0192148	-1.52953	-0.9947

Table 3.5. MSDI-RBF Lyapunov Estimates for Li equation

CHAPTER 4

Discussion

In Table 3.1 we see significant variation in the results to the test algorithm. This is seen strongly in λ_2 for Lorenz where at $dt = 0.1$ $\lambda_2 \approx 0.17$ but at $dt = 0.001$ $\lambda_2 \approx 0.004$. However, we do see convergent behavior as we decrease the time step of this algorithm when comparing the results the results to the literature (Table 4.1). Therefore we can conclude that the test algorithm is a good metric to compare MSDI-RBF against.

The results from Lorenz are promising and suggest that we can trust the Lyapunov exponent approximations produced by the MSDI-RBF scheme for the Lorenz equations. Comparing Figure 3.1 to Figure 3.2 we see that the shape parameter ϵ is more responsive to moving along the trajectory than the fill distance in respect to the Jacobian error. We see that the optimal shape parameter is erratic with respect to time. The shape parameter allows the RBF to adapt as the data set evolves in time and the behavior in Figure 3.2 explains the ability of the RBF to adapt to changing conditions keeping the overall relative error of the Jacobian approximations small. This result tells us that RBFs give us a straightforward and accurate means of approximating Jacobians along a flow.

The choice of a time step for the Runge-Kutta scheme has a strong impact on the accuracy of the MSDI-RBF scheme for the Lorenz equation. In Figure 3.1a we can see a minimum error around the order of -2.5. However, in Figure 3.1b we see errors on the order of -3. However, we also see periodic order one errors. The associated shape parameter in Figure 3.2b also appears less dynamic, with most values < 40 with rare and drastic spikes interspersed. We see the result of these errors and lack of adaptability in Table 3.2 for the trial $t = 90$, $= 0.001$ as the minimal Lyapunov exponent is inaccurate. This is a manifestation of the accuracy vs. stability trade off principle. The length of the simulation run time also seems to have an impact on on the accuracy of the scheme but its contribution is dwarfed by the time step. We can see that this is the case in Figure 3.1c as the order of error barely exceeds -2.5, giving us less accuracy than the significantly shorter simulation run time presented in Figure 3.1b. Additionally, In Figure 3.1d the error decreases by a magnitude from Figure 3.1b reaching an order of -4

demonstrating that simulation run time has a stronger impact the finer the data is.

While the maximal Lyapunov exponent of $t = 90,\ dt = 0.001$ is on par with $t = 1000,\ dt = 0.01$ (Table 3.2), the smallest Lyapunov exponent of the latter is more accurate according to Table 3.1 and results that are found in the literature (Table 4.1). This suggests that the length of the simulation run time may have a greater role to play in the accuracy of the smallest Lyapunov exponent. As stated previously, we can see the order of error jump periodically to order one in Figure 3.1b. These moments of high error contribute to the higher degree of inaccuracy for the smallest Lyapunov exponent. These periodic jumps seem to lessen as we increase the time length (Figure 3.1d). It is likely that both of the aformentioned parameters have a role to play in the accuracy of the minimal Lyapunov exponent as we see convergence in the minimal exponent for $t = 1000,\ dt = 0.001$.

Turning our attention to Rossler, the Jacobian errors behave very differently from what we saw for the Lorenz Jacobian errors. Rossler's Jacobian errors have periodic order one reoccurances with strong, but rare, radical departures. The Jacobian errors for Rossler do not seem well correlated to the fill distance, suggesting that the problem lies in the centered difference approximation underlying the interpolation scheme. As can be seen in Figure 3.5, the shape parameter appears less adaptive than what we see with Lorenz. Rossler has strong temporal gradients due to a nearly singular perturbative phenomenon in its dynamics. This manifests as stiffness in our algorithm.

Figure 3.4 lends some evidence to the aforementioned hypothesis. Looking at Figure 3.4a, we can see that this experiment yielded minimum Jacobian errors on the order of -3 while Figure 3.4b yielded minimal errors on the order of -2. If we look at the corresponding data in Table 3.3, $t = 180,\ dt = 0.01$ and $t = 180,\ dt = 0.001$ respectively, the latter's maximal Lyapunov exponent is less accurate than the former's according to Table 3.1. In Figure 3.4c and d we see that the minimal Jacobian errors reach the order of -4 (with one spike reaching -5 in Figure 3.4d). Referring back to Table 3.3 we can see convergent behavior of the maximal Lyapunov exponent with respect to Table 3.1 and the literature [18]. Additionally, we can see similar behavior in the minimal Lyapunov exponent for Rossler generally and $t = 180,\ dt = 0.001$ in Table 3.2.

We conducted trials both Rossler and Lorenz measuring the effect of increasing the number of nearest nearest neighbors on the Jacobian error and Lyapunov estimates

(Figures 3.3 and 3.6). However, the effect the number of nearest neighbors has in this range seems negligible on the Jacobian error. Multiquadric and inverse basic functions also appear to have a negligible effect on our estimates (Figures 3.3 and 3.6). The Lyapunov exponent estimates for both systems are presented in Table 3.2 and 3.3.

The trials for the Chua and Li equations were not as extensive in scope as Lorenz and Rossler. The Chua and Li attractors share similar geometry with the Lorenz attractor and as such we can hypothesize that they will have similar performance with the MSDI-RBF algorithm. This hypothesis is confirmed by the Lyapunov exponent estimates of Chua and presented in Table 3.4 and of Li in Table 3.5. These results demonstrate the effectiveness of the MSDI-RBF algorithm for attractors beyond Lorenz. For both systems we have similar results to the Lyapunov exponents calculated for these systems in Table 3.1 as well as what is presented in the literature (Table 4.1).

system	λ_1	λ_2	λ_3
Lorenz [18]	9.056	0	-14.5723
Rossler [18]	0.0714	0	-5.3943
Chua [18]	0.3271	0	-2.5197
Li [14]	0.131	0	1.131

Table 4.1. Lyapunov exponents for systems from literature

CHAPTER 5

Conclusion

This thesis has shown that the MSDI-RBF method is a promising method for the estimation of Lyapunov exponents. The method offers a globally convergent data driven method for calculating Lyapunov exponents of dynamical systems, but there are situations where the scheme struggled. The Lorenz estimates for finer time steps $t = 1000,\ dt = 0.005,\ = 0.001$ gave accurate estimates of the Lyapunov exponent. The estimates for Chua are very good. Li's Lyapunov estimates are less accurate. However, based on what we see in Table 3.2 for the trials where $t = 1000,\ dt = 0.01,\ N_{nb} = 50$, we can infer that Li would benefit from an improved time step. It is encouraging that the trials for these attractors yielded results similar to what can be seen in the literature (see Table 4.1).

Singularly perturbed flows such as Rossler suffer from stiffness in the algorithm causing inaccurate results. Additionally, there is some evidence that fine data collected over too short of an amount of time may yield inaccurate results particular in respect to the minimal Lyapunov exponent as we see in Table 3.2 for trial $t = 90,\ dt = 0.001$. However, the maximal Lyapunov exponent seems easier for MSDI-RBF to discern even for difficult attractors such as Rossler. The MSDI-RBF algorithm can then be used to firmly establish the presence of chaos in a dynamical system.

Centered finite differencing is based on Taylor series expansions and as such are locally convergent and may introduce errors. A topic of future work may be to develop an RBF scheme, or use other numerical differentiation methods, to generate the initial approximation of points in the unknown function $\mathbf{f}(\mathbf{y})$ instead of relying on finite differencing methods.

Preconditioning may help to produce more accurate of singularly perturbed systems. Recently, machine learning has been utilized to study dynamical systems. There is a family of techniques that utilize the Koopman operator called Dynamic Mode Decomposition (DMD). The Koopman operator allows one to take complicated nonlinear dynamics and put them into a space where they appear linear. Recent work by *Curtis et al.* [4] and *Gilpin* [9] use autoencoders to achieve this. Use of such techniques may

allow one to take ill-behaved systems and put them in a space where the MSDI-RBF algorithm is easier to work with.

Once accurate results for the Rossler equation can be obtained with an improved MSDI-RBF procedure. The improved scheme should be tested with the Rabinovich-Fabrikant equation across different equation parameters. The Rabinovich-Fabrikant equation is a dynamical system also capable of chaotic behavior. Changing the equation parameters of this system can radically alter the geometry of the attractor. an attractor of similar singularly perturbed geometry that Rossler exhibits [18]. Additionally, the same equation parameter This equation is capable of producing different attractors by using different time steps in the numerical integrator [5]. The difficulty of this equation would provide a good stress test to an improved MSDI-RBF in future work.

BIBLIOGRAPHY

[1] K. T. Alligood, T. D. Sauer, and J. A. Yorke, *Chaos: An Introduction to Dynamical Systems*, Springer, New York, 1996.

[2] L. O. Chua, *Chua circuit*, Scholarpedia, 2 (2007), p. 1488. revision #140932.

[3] W. S. Cleveland and C. Loader, *Smoothing by local regression: Principles and methods*, in Statistical Theory and Computational Aspects of Smoothing, W. Härdle and M. G. Schimek, eds., Heidelberg, 1996, Physica-Verlag HD, pp. 10–49.

[4] C. W. Curtis, D. J. Alford-Lago, and O. Issan, *Deep learning enhanced dynamic mode decomposition*, CoRR, abs/2108.04433 (2021).

[5] M.-F. DANCA and G. CHEN, *Bifurcation and chaos in a complex model of dissipative medium*, International Journal of Bifurcation and Chaos, 14 (2004), pp. 3409–3447.

[6] L. Dieci, R. Russell, and E. Vleck, *On the compuation of lyapunov exponents for continuous dynamical systems*, Siam Journal on Numerical Analysis - SIAM J NUMER ANAL, 34 (1997).

[7] J. P. Eckmann, S. O. Kamphorst, D. Ruelle, and S. Ciliberto, *Liapunov exponents from time series*, Phys. Rev. A, 34 (1986), pp. 4971–4979.

[8] G. E. Faushauer, *Meshfree Approximation Methods with MATLAB*, vol. 6 of Interdisciplinary Mathematical Sciences, World Scientific Publishing Co., Singapore, 2007.

[9] W. Gilpin, *Deep reconstruction of strange attractors from time series*, (2020).

[10] J. Gleick, *Chaos: Making a New Science*, Open Road Media, New York, 2011.

[11] R. Haberman, *Applied Partial Differential Equations*, Pearson, Upper Saddle River, New Jersey, 5 ed., 2013.

[12] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, *Determining embedding dimension for phase-space reconstruction using a geometrical construction*, Phys. Rev. A, 45 (1992), pp. 3403–3411.

[13] D. C. Lay, *Linear Algebra*, Addison Wesley, Boston, 4 ed., year.

[14] C. Li, J. C. Sprott, W. Thio, and H. Zhu, *A unique signum switch for chaos and hyperchaos*, 08 2015.

[15] E. N. Lorenz, *Deterministic nonperiodic flow*, Journal of Atmospheric Sciences, 20 (1963), pp. 130 – 141.

[16] T. Matsumoto, *A chaotic attractor from chua's circuit*, IEEE Transactions on Circuits and Systems, 31 (1984), pp. 1055–1058.

[17] C. Meador and Marshall, *A comparison of two 4 th-order numerical ordinary differential equation methods applied to the rabinovich-fabrikant equations*, 2009.

[18] C.-E. E. Meador, *Numerical calculation of lyapunov exponents for three-dimensional systems of ordinary differential equations*.

[19] M. I. Rabinovich and A. L. Fabrikant, *Stochastic wave self-modulation in nonequilibrium media*, Zhurnal Eksperimentalnoi i Teoreticheskoi Fiziki, 77 (1979), pp. 617–629.

[20] L. M. Resler, *Edward n lorenz's 1963 paper, "deterministic nonperiodic flow", in journal of the atmospheric sciences, vol 20, pages 130–141: Its history and relevance to physical geography*, Progress in Physical Geography: Earth and Environment, 40 (2016), pp. 175–180.

[21] S. Rippa, *An algorithm for selecting a good parameter c in radial basis function interpolation*, Advances in Computational Mathematics, 11 (1999), pp. 193–210.

[22] O. Rössler, *An equation for continuous chaos*, Physics Letters A, 57 (1976), pp. 397–398.

[23] B. Saltzman, *Finite amplitude free convection as an initial value problem—i*, Journal of Atmospheric Sciences, 19 (1962), pp. 329 – 341.

[24] S. H. Strogatz, *Nonlinear Dynamics and Chaos*, World Scientific Publishing Co., Singapore, 2 ed., 2007.

[25] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, *Determining lyapunov exponents from a time series*, Physica D: Nonlinear Phenomena, 16 (1985), pp. 285–317.

[26] G.-Q. Zhong and F. Ayrom, *Experimental confirmation of chaos from chua's circuit*, International Journal of Circuit Theory and Applications, 13 (1985), pp. 93–98.

APPENDIX

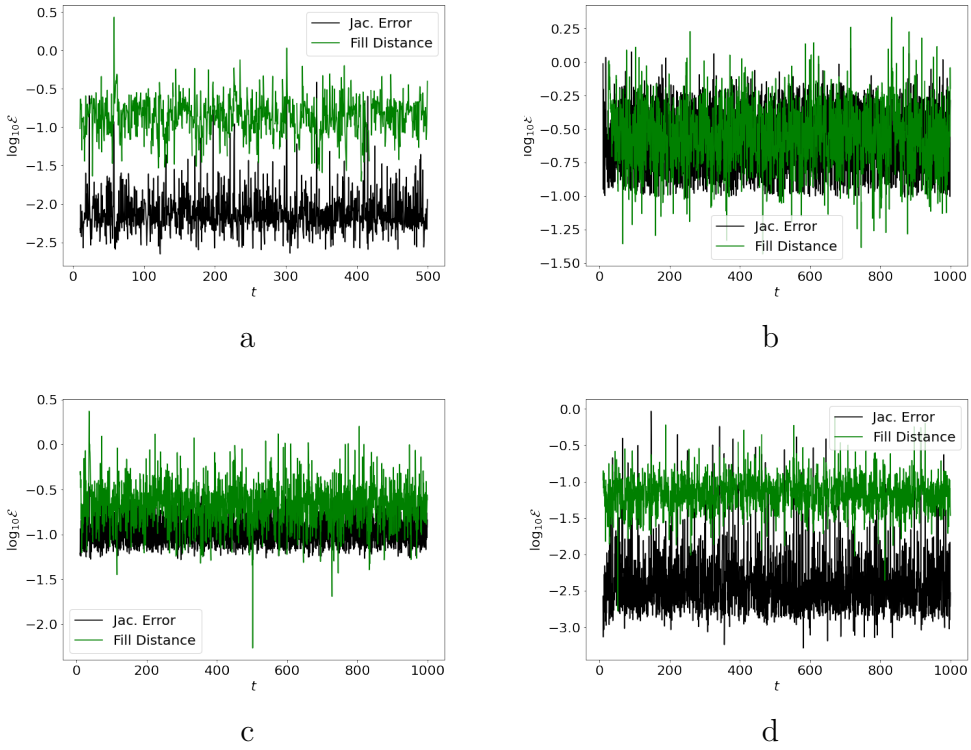


Figure A.1. Lorenz Jacobian Error and Fill Distance, a. $t = 500$, $dt = 0.01$ **b.** $t = 1000$, $dt = 0.1$ **c.** $t = 1000$, $dt = 0.05$ **d.** $t = 1000$, $dt = 0.005$

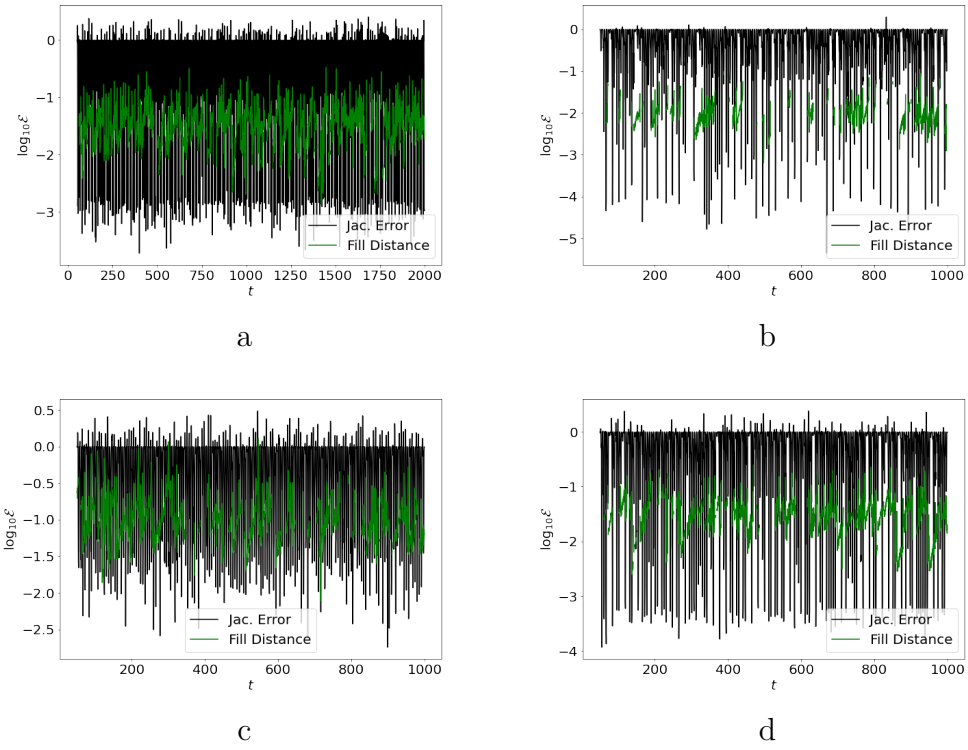


Figure A.2. Rossler Jacobian Error and Fill Distance, a. $t = 2000$, $dt = 0.01$ **b.** $t = 1000$, $dt = 0.1$ **c.** $t = 1000$, $dt = 0.05$ **d.** $t = 1000$, $dt = 0.005$