San Diego State University
Department of Mathematics

Meshfree data driven scattered interpolation estimates of lyapunov exponents using optimized radial basis functions

Nathanael J. Reynolds

Advisor
Christopher W. Curtis

April 4, 2022

Abstract

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 determine the Lyapunov exponents of two additional dynamical systems, the Chua equation, and The Li equation *Li et al.* [12], and Rabinovich-Fabrikant. Rabinovich-Fabrikant is tested with two different parameter values that produce attractors that are geometrically distinct. The time series used as data points for the scheme were obtained by integrating all systems using 4-th-order Runge-Kutta. The meshfree scattered data interpolation with radial basis functions (MSDI-RBF) scheme appears to be sensitive to data

For my mother, Laurie Reynolds, who has always supported me and Perri Gellman, for putting me down the road of mathematics.

Acknowlegements

- Christopher W. Curtis for his guidance on this topic and his personal kindness
- Joseph Diaz for his positivity and helping me through some difficult classes and always being available to help
- Jay Lago for always being there to answer questions regarding research
- The applied mathematics Fall 2020 graduate cohort for the support and genuine good times

Contents

1 Introduction **5**
- 1.1 Chaos . . . 5
- 1.2 Dynamical systems . . . 6
  - 1.2.1 The Lorenz Equations . . . 7
  - 1.2.2 The Rossler Attractor . . . 7
  - 1.2.3 The Chua Circuit . . . 8
  - 1.2.4 The Signum "Li" Switch . . . 9
- 1.3 Lyapunov Exponents . . . 10
  - 1.3.1 Numerical Methods . . . 11
- 1.4 Meshfree Scattered Data Interpolation . . . 12
  - 1.4.1 Radial Basis Functions . . . 12

2 Methods **13**
- 2.1 Lyapunov Exponent Estimation . . . 14
  - 2.1.1 QR-Decomposition . . . 14
  - 2.1.2 Calculating the Jacobian . . . 15
- 2.2 Shape Parameter . . . 16
  - 2.2.1 Condition Number . . . 17
  - 2.2.2 Trial and Error . . . 17
  - 2.2.3 LOOCV "Drop-One" Algorithm . . . 18
- 2.3 Fill Distance . . . 19

3 Results **19**

4 Discussion **19**

5 Conclusions **19**

6 Appendix **23**
- 6.1 Attractor Plots . . . 23
- 6.2 Condition Number Plots . . . 29
  - 6.2.1 Lorenz . . . 29
  - 6.2.2 Rossler . . . 31
  - 6.2.3 Chua . . . 34
  - 6.2.4 Signum . . . 37
  - 6.2.5 RF1 . . . 40
  - 6.2.6 RF2 . . . 43

6.3 ϵ vs error plots . . . 46
6.3.1 Lorenz . . . 46
6.3.2 Rossler . . . 48
6.3.3 Chua . . . 51
6.3.4 Signum . . . 54
6.3.5 RF1 . . . 57
6.3.6 RF2 . . . 60
6.4 Jacobian . . . 63
6.4.1 Analytic Jacobians in General Form . . . 63
6.4.2 Exact Numerical Jacobians . . . 64
6.4.3 Approximated Jacobians . . . 65
6.5 Fill Distances . . . 71
6.6 Lyapunov Estimates . . . 71

1 Introduction

Since 1963, when Edward Lorenz made his famous discovery while studying atmospheric convection, chaos has been a cornerstone in the field of dynamical systems. Chaotic systems appear in many systems both natural and man-made, understanding them is of great importance to science. One of the ways to study chaotic systems is by their Lyapunov exponents. The Lyapunov exponent is a measure of how quickly the trajectories of two distinct and arbitarily distant points On the attractor seperate.

This thesis explores a data method for estimating the Lyapunov exponent of a dynamical system. We call this method meshfree scattered data interpolation using radial basis functions (MSDI-RBF). The method offers an approach that is globally convergent to the data. Six systems are studied in this work: The Lorenz Attractor, The Rossler Attractor, The Chua Circuit, The Signum "Li" Switch, and the Rabinovich-Fabrikant equations with two sets of parameters.

1.1 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.

No universally accepted definition of chaos exists [19]. However, there are agreed upon characteristics of chaotic systems:

1. The system must exhibit **aperiodic long-term behavior**
2. The system is **deterministic**
3. The system exhibits **sensitive dependence on initial conditions**

This thesis focuses on a data driven method for calculating the Lyapunov exponents of a dynamical system. The method uses MSDI-RBF in order to approximate the Lyapunov exponents of a dynamical system. Before looking at Lyapunov exponents however, we explore dynamical systems first.

1.2 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 [9][19]. 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 [2].

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 [Figure **??**].

Strange attractors are the topic of this thesis. Strange attractors attract all trajectories in the phase space onto them but they do not attract to a single point nor do trajectories on the attractor fall into periodicity. Trajectories on a strange attractor are said to be "space filling," they will visit all points within the attractor as $t \to \infty$ but they will not visit the same point twice. An example of this space filling behavior can be seen in Figure 6. Strange attractors are inherently nonlinear phenomenon and a dynamical system exhibiting strange attractor behavior is said to be chaotic (see §1.1). The trajectories of dynamical systems are subject to change by changing parameters within the system e.g. the trajectories can be drawn to a limit-cycle for one set of parameters and a strange attractor for another. This phenomenon is called bifurcation and will not be further explored in this work. The interested reader may refer to [19].

The dynamical systems studied in this work are presented in the subsections 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.*[12].

1.2.1 The Lorenz Equations

In 1963 the meteorologist Edward Lorenz was running a twelve equation weather simulation when he made a peculiar discovery. He wanted to continue calculating where the simulation had last left off and 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, diverged into complete 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. What he came up with is a system of three ordinary differential equations given by:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(r - z) - y \\ \dot{z} &= xy - bz \end{aligned} \tag{1}$$

These equations are known today as the Lorenz equations. While the equations were designed to model convection within a fluid, they also model an electric dynamo [8] and the chaotic waterwheel[8][19]. When plotted in phase space, these equations produce the famous butterfly wing phase plot [Figure 1], which is apropos for the colloquial name of chaos, The Butterfly Effect.

1.2.2 The Rossler Attractor

In 1976 Rossler created a model of a model[18], the Rossler attractor [Figure 2] was created as a simpler approximation of the Lorenz system. The system is defined as:

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{aligned} \tag{2}$$

The attractor only has one equilibrium point and around it is a spiral. Part

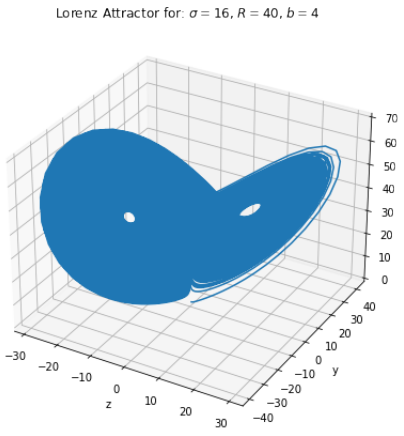

Figure 1: Phase space of the Lorenz Attractor for initial point (0, 1, 0)

of the beauty of this system is that two of the equations involved are linear. This aspect of the system makes eigenvalues of the system easier to find.

1.2.3 The Chua Circuit

The Chua circuit is the simplest circuit that is capable of exhibiting chaotic behavior[11]. 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 eqautions in order to demonstrate chaos as a robust physical phenomenon and not the result of numerical error[11]. The second motivation was to show that the Lorenz equations were chaotic in a rigorous mathematical sense[11]. This work uses the Chua equations as presented in *Meador*[15]:

$$\begin{aligned} \dot{x} &= \alpha(y - x - f(x)) \\ \dot{y} &= x - y + z \\ \dot{z} &= -\beta y \end{aligned} \tag{3}$$

$$f(x) = \begin{cases} bx - a + b & \text{for} \quad x \geq 1 \\ ax & \text{for} \quad -1 < x < 1 \\ bx + a - b & \text{for} \quad x \leq 1 \end{cases} \tag{4}$$

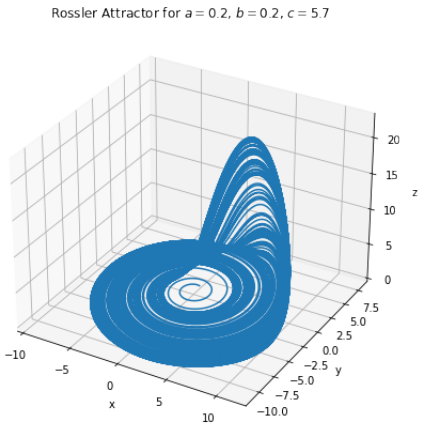

Figure 2: Rossler Attractor for initial point (0, 1, 0)

Like Rossler, two of the Chua equations are linear. 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[11][13]. In 1985, Zhong and Ayrom experimentally observed chaotic attractors[11][cite] and in 1986 *Chua et al.* rigorously proved chaos for the equations [11].

1.2.4 The Signum "Li" Switch

The signum switch is a circuit that was contructed and studied by *Li et al.*[12] Mathematically, the dynamical system is concise:

$$\begin{aligned} \dot{x} &= y - x \\ \dot{y} &= -z\text{sgn}(x) \\ \dot{z} &= x\text{sgn}(y) - a \end{aligned} \tag{5}$$

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 [12]. *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 Nonlin-

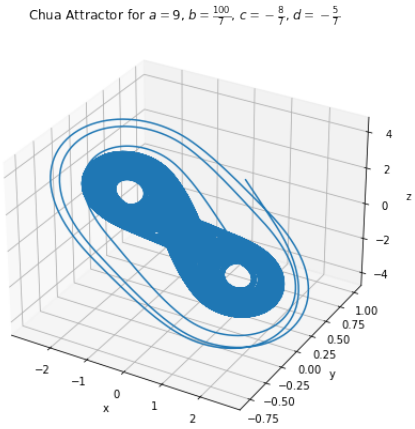

Figure 3: Chua Attractor for initial point (0, 1, 0)

earcircuits created a module for use in modular synthesizers based on the work in *Li et al.*

1.3 Lyapunov Exponents

The motion on strange attracts exhibits sensitive dependence on initial conditions (see §1.1). This means that any two trajectories that are not identical to each other exponentially diverge from one another. Lyapunov exponents are a measure of the exponential rates of divergence of neighboring trajectories of a dynamical system [17]. Lyapunov exponents give us a measure of how predictable the dynamical system is. The presence of a positive Lyapunov exponent is the telltale indicator of chaos. The Lyapunov exponent of a real valued function $f(t)$ for $t \geq 0$ is given by the equation:

$$\lambda = \lim_{t \to \infty} \frac{1}{t} log(|f(t)|) \tag{6}$$

whereby λ characterizes the long-term exponential behavior of a given n-dimensional linear system [4]:

$$\dot{\mathbf{y}}(t) = A(t)\mathbf{y}(t) \tag{7}$$

.

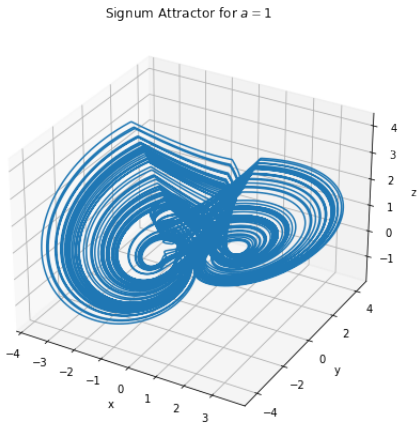

Figure 4: Li Attractor for initial point (0, 1, 0)

$A(t)$ represents a bounded coefficient matrix and as such λ is well-defined [4]. The solution to (7) take the form $\mathbf{y}_i = Y(t)\mathbf{p}_i$ for $i \in \mathbb{N}$. $Y(t)$ represents the fundamental solution matrix and $\mathbf{p}_i$ is an orthonormal basis in $\mathbb{R}^n$ [4]. Thus i-th Lyapunov exponent can be represented as:

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} log(\|Y(t)\mathbf{p}_i\|) \tag{8}$$

1.3.1 Numerical Methods

Lyapunov exponents are often difficult to compute analytically and thus or commonly solved with numerical ‘ methods. Several methods have been developed for numerical estimation of Lyapunov exponents. 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 [5]. *Wolf* introduces a method that utilizes Gram-Schmidt reorthonormalization [20]. *Deici et al.* uses a QR-decomposition of the **Jacobian** of the dynamical system to estimate the Lyapunov exponents [4], *Arbarbanel et al.* utilize a similar OR-method but introduces a data driven method that uses Taylor series to approximate the Jacobian [1].

1.4 Meshfree Scattered Data Interpolation

Mesh generation remains one of the most time consuming aspects of mesh-based numerical simulations such as: finite differences, finite elements, or finite volumes [6, 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 [6]. 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.

1.4.1 Radial Basis Functions

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

$$\phi(r;\epsilon) = e^{-(\epsilon r)^2} \tag{9}$$

where ϵ represents the *shape parameter*. 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

$$\gamma_k(x) = \|x - x_k\|_2 \tag{10}$$

Then by $\phi(gamma(x))$ one can obtain a fixed center $x_k \in \mathbb{R}^s$ [6]

$$\Phi_k(x;\epsilon) = \phi(\gamma_k(x)) = e^{-\epsilon\|x-x_k\|_2^2},\ x \in \mathbb{R}^s \tag{11}$$

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

Definition 1.1. 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

$$\Phi(x) = \phi(r), \quad \text{where} \quad r = \|\mathbf{x}\|$$

and $\|\cdot\|$ is some norm on $\mathbb{R}^s$ – usually the Euclidean norm [6, pp.17].

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: Basic functions

When $\phi(r)$ in definition 1.1 is a basic function, we call it a radial basis function (RBF). The Φ_k presented is a simple Euclidean distance matrix. However, using a radial basis function expansion one is able to solve scattered data interpolation problems on $\mathbb{R}^s$ by assuming

$$\mathcal{P}_f(x) = \sum_{k=1}^{N} c_k \phi(\gamma_k(x)) \tag{12}$$

coefficients c_k are found by solving the linear system.

$$\begin{bmatrix} \phi(\|x_1 - x_1\|_2) & \cdots & \phi(\|x_1 - x_N\|_2) \\ \vdots & \ddots & \vdots \\ \phi(\|x_N - x_1\|_2) & \cdots & \phi(\|x_N - x_N\|_2) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix} \tag{13}$$

The work presented tests the scheme with several different RBFs. The basic functions used in the interpolator are presented in Table 1.

2 Methods

The algorithm used to calculate the Lyapunov exponents in this thesis is a data-driven scheme. No knowledge of an underlying function is in principle not necessary to know. The dynamical systems tested in this thesis however, are all known dynamical systems with known differential equations. In this work, all of the dynamical systems were solved using the 4th-order Runge-Kutta method. All of the systems are integrated $t = 0 \to 1000$ with a time step $h = 0.01$ to generate the data array for the system. For the Rossler and the RF2 attractor, an additional time step of $h = 0.05$ is tested as well. A finer time step is used in order to measure the fill distance of the attractor data. The fine data is calculated over the same time scale but $h = 0.001$. The initial condition for all systems is $(x_1, x_2, x_3) = (0, 1, 0)$.

2.1 Lyapunov Exponent Estimation

2.1.1 QR-Decomposition

In this work an algorithm developed by *Deici et al.* is used in order to check the results of the RBF interpolation method presented. The algorithm uses QR-decomposition of the a dynamical system's Jacobian matrix in order to calculate the Lyapunov exponents of said dynamical system. The method is not data driven and relies on the exact Jacobian of the system calculated from the differential equations. The work of this thesis uses the QR-decomposition of the Jacobian but uses time series data collected from numerically integration and feeding it into the MSDI-RBF algorithm in order to approximate the Jacobian matrix. A brief overview of the QR-algorithm developed by Deici will be presented. However, see *Deici et al.* [4] for a more comprehensive discussion.

The method described holds for all of the dynamical systems presented in this work. However, as an example consider the Lorenz system from equation (1). In this work the following values were chosen for the parameters of the Lorenz system: $\sigma = 16,\ r = 40,\ b = 4$. For a generic initial condition a general solution can be described by the flow map $\mathbf{X}(t) = \phi(t; \mathbf{x})$ where $\mathbf{x} = \{x_1, x_2, ..., x_N\}$. represents the data set. When numerically integrated using the 4th-order Runge-Kutta scheme, one obtains the famous Lorenz butterfly [Figure 1.2.1].

For a given solution one can linearize the system by computing the affiliated Jacobian of (1).

$$\dot{\mathbf{X}} = J(t)\mathbf{X}$$

where

$$J(t) = \begin{bmatrix} -\sigma & \sigma & 0 \\ r - z(t) & -1 & -x(t) \\ y(t) & x(t) & -b \end{bmatrix} \tag{14}$$

A discretized approximation to the solution of (1) is defined at discrete times $\{t_j\}_{j=0}^{N_T}$, $t-0 = 0$ and $t_j = j\delta t$. An approximation to the Lyapunov exponents is generated when solving for $t_j \leq t \leq t_{j+1}$

$$\frac{d}{dt}\mathbf{U}_j = J(t)\mathbf{U}_j,\ \mathbf{U}_j(t_j) = \mathbf{Q}_j \tag{15}$$

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

$$\mathbf{U}_j(t_{j+1}) = \mathbf{Q}_{j+1}\mathbf{R}_{j+1} \tag{16}$$

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

$$\lambda_m = \lim_{j\to\infty} \frac{1}{t_j} \sum_{k=1}^{j} \log |(\mathbf{R}_k)_{mm}| \tag{17}$$

2.1.2 Calculating the Jacobian

Deici et al.'s [4] method is not a data driven method. It relies on Jacobians obtained by linearizing the equations of a dynamical system. 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 present are data sets collected from an experimental system. Therefore, proposed in this work is a data driven method for calculating the Lyapunov exponents of a dynamical system.

To get estimates of Lyapunov experiments from data alone we need Jacobian map estimates that we otherwise can't access. *Abarbanel et al.* [1] dealt with this issue by using least-squares fitting to Taylor series expansions. This approach is algorithmically unwieldy. Additionally, because Taylor series expansions only converge locally, this approach may not be the most accurate approach. MSDI-RBFs converge globally and can be used to generate analytic approximations to the map at each time step using the nearest-neighbors to a given point $y(t_j)$. The method produces a sequence of approximate Jacobians, $\tilde{\mathbf{J}}_j$. From time-step to time-step, we have the analytic formula for the matricies $\mathbf{U}_j$

$$\mathbf{U}_j(t_{j+1}) = \mathbf{Q}_j + \int_{t_j}^{t_{j+1}} \tilde{\mathbf{J}}(t)\mathbf{U}_j(t)dt \tag{18}$$

The Trapezoid approximation scheme is used to get a stable approximation of (18)

$$\left(\mathbf{I} - \frac{\delta t}{2}\tilde{\mathbf{J}}_{j+1}\right)\mathbf{U}_j\left(t_{j+1}\right) = \left(\mathbf{I} - \frac{\delta t}{2}\tilde{\mathbf{J}}_j\right)\mathbf{Q}_j \tag{19}$$

In order to avoid transients, the first 400 points from the data set $\mathbf{x} = \{x_1, ..., x_N\}$ are removed to produce the reduced data set $\mathbf{x}_{red}$. A list of the neartest 100 neighbors to every point on the trajectory $\mathbf{x}_{red}$ is then generated. The data from $\mathbf{x}_{red}$ is placed into the distance matrix $\mathbf{\Phi}$ represented by (13). An RBF is chosen to represent $\phi(r, \epsilon)$. The linear system is then solved for $\mathbf{c} = [c_1, ..., c_N]^T$. One then obtains a new unit distance matrix $\mathbf{\Phi'}$ with $\hat{\phi}'(r, \epsilon) = \frac{\phi'(r,\epsilon)}{\|r\|_2}$ as its entries.

$$\mathbf{\Phi'} = \begin{bmatrix} \hat{\phi}'(\|k_1 - x_1\|_2) & \cdots & \hat{\phi}'(\|k_1 - x_N\|_2) \\ \vdots & \ddots & \vdots \\ \hat{\phi}'(\|k_N - x_1\|_2) & \cdots & \hat{\phi}'(\|k_N - x_N\|_2) \end{bmatrix} \tag{20}$$

where $\mathbf{k} = \{k_1, \ldots, k_N\}$ represents the indicies of the nearest neighbors to $\mathbf{x}_{red}$. We then obtain a matrix of distance vectors $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N]$. The linear equation

$$\mathbf{V} = \mathbf{c}^T \mathbf{\Phi'} \tag{21}$$

is then solved. The approximated Jacobian is then obtained by solving

$$\tilde{\mathbf{J}} = \mathbf{VK} \tag{22}$$

where $\mathbf{K}$ is represented by

$$\mathbf{K} = \begin{bmatrix} k_1 - x_1 & \cdots & k_1 - x_N \\ \vdots & \ddots & \vdots \\ k_N - x_1 & \cdots & k_N - x_N \end{bmatrix} \tag{23}$$

2.2 Shape Parameter

The accuracy of many RBF schemes is dependent on a shape parameter ϵ. 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* [6][17]. Other methods presented by Faushauer [6] will be briefly presented in order to contrast the weaknesses of the alternative approaches that led to Rippa's algorithm being chosen for 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. This fact means that if $\mathbf{A}$ is singular then it is non-invertible. Invertibility is often required for solving matrix equations since $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. Let $\mathbf{B}$ be an invertible matrix where det($\mathbf{B}$)$= 0 + \eta$ where $0 < \eta << 1$. Then $\mathbf{B}$ is said to be *ill-conditioned*. Ill-conditioned matricies are said to be *near-singular*, meaning that small perturbations in the matrix elements can cause the matrix to become singular [10]. 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 will be inaccurate, only that results cannot be trusted on their face.

The *condition number* of a matrix is a measure of how ill-conditioned a matrix is. 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 and thus, there is no guarantee that the scheme is stable for the given data set. This feature is what is called the *trade-off principle* [6]. Faushauer identifies three trade-off principles: *Accuracy vs. Stability*, *Accuracy and Stability vs. Problem Size*, *Accuracy vs. Efficiency*. The first principle has been the topic of this section. The second will be briefly discussed in the following sections, and the third will not be discussed in this thesis but the interested reader may refer to Faushauer. In the following subsections, various schemes for finding an optimal ϵ are presented.

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 [6]. 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 generating data on a chaotic attractor is almost never known. Additionally, underlying functions of real-world systems are also rarely known leaving this approach unsuitable for the this work.

2.2.3 LOOCV "Drop-One" Algorithm

Rippa 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 performing a least-squares error fit based on the data point that was dropped [6]. The algorithm is presented in this section. This 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 (12) one arrives at

$$\mathcal{P}_f^{[k]}(x) = \sum_{j=1}^{N} c_j^{[k]} \phi(\gamma_k(x)), \;\; j \neq k \tag{24}$$

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

$$E_k = f_k - \mathcal{P}_f^{[k]}(x_k) \tag{25}$$

This implementation of the algorithm is expensive [6]. However, E_k can be simplified into

$$E_k = \frac{c_k}{A_{kk}^{-1}} \tag{26}$$

with c_k representing the kth coefficient of $\mathcal{P}_f$ and $A_{kk^{-1}}$ represents the kth diagonal element of the inverse interpolation matrix. The derivation for this formula is given in [17].

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

2.3 Fill Distance

The attractors in this thesis have different geometries and to get a sense of how they distribute data 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 $\mathbf{x}$ and is given by

$$h = h_{\chi,\Omega} = \sup_{x\in\Omega} \min_{x_j\in\chi} \|x - x_j\| \tag{27}$$

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 [6]. The data set worked with in [6] 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 is used in order to calculate the fill distances on the attractor.

In this thesis an absolute fill distance, corresponding to the smallest fill distance, is calculated for each attractor and a fill distances calculated as a function of time are calculated and plotted. The fill distance plots represent how the fill distance changes as trajectories move around the attractor.

3 Results

The results presented in this section will present

4 Discussion

5 Conclusions

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 failed. It may be the case that "flat and peaky" attractors such as Rossler and some of the attractors created by Rabinovich-Fabrikant, may cause the MSDI-RBF scheme to perform poorly. Additionally, this scheme is sensitive to the fineness of data. The researcher using this scheme will have to calibrate the data resolution to neither be too fine or too course. In future work, developing an algorithm to determine an optimal time step would be of use.

Some preconditioning may help the results of syatems such as Rossler and RF2. 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.* [3] and *Gilpin* [10] 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.

References

“

1. Abarbanel, H.D.I., Brown, R., Kennel, M.B. Local Lyapunov exponents computed from observed data. J Nonlinear Sci 2, 343–365 (1992). https://doi.org/10.1007/BF01208929

2. Alligood KT, Sauer TD, Yorke JA. Chaos An Introduction to Dynamical Systems.New York: Springer-Verlag.; 1996.

3. Curtis, C., Alford-Lago J., Issan O. Deep Learning Enhanced Dynamic Mode Decomposition. ArXiv, arXiv:2108.04433 (2021). https://arxiv.org/pdf/2108.04433v3.pdf

4. Dieci, Luca. Russell, Robert. Vleck, Erik. (1997). On the Compuation of Lyapunov Exponents for Continuous Dynamical Systems. Siam Journal on Numerical Analysis - SIAM J NUMER ANAL. 34. 10.1137/S0036142993247311.

5. Eckmann, Jean-Pierre Kamphorst, Sylvie Ruelle, D Ciliberto, Sergio. (1987). Liapunov exponents from time series. Physical review. A. 34. 4971-4979. 10.1103/PhysRevA.34.4971.

6. Fasshauer GE. Meshfree Approximation Methods with MATLAB. Singapore: World Scientific Publishing Co.; 2007.

7. Gilpin, W. Deep reconstruction of strange attractors from time series. ArXiv, arXiv:2002.05909 (2020). https://arxiv.org/abs/2002.05909

8. Gleick, J., Chaos: Making a New Science, New York: Open Road Media; 2011

9. Haberman R. Applied Partial Differential Equations with Fourier Series and Boundary Value Problems. 5th ed. Pearson.; 2013

10. Lay, David C. Linear Algebra and its approximations. 4th wd. Pearson; 2012.

11. Leon O. Chua (2007) Chua circuit. Scholarpedia, 2(10):1488.

12. Li, Chunbiao. Sprott, Julien Clinton. Thio, Wesley. Zhu, Huanqiang. (2015). A unique signum switch for chaos and hyperchaos.

13. Matsumoto, T. (1984) A Chaotic attractor from Chua's Circuit, IEEE Transaction on Circuits and Systems, 31 : 1055-1058.

14. Meador, Clyde Sarra, Scott. (2022). A Comparison of two 4th-Order Numerical Ordinary Differential Equation Methods Applied to the Rabinovich-Fabrikant Equations.

15. Meador, Clyde-Emmanuel Estorninho, "Numerical Calculation of Lyapunov Exponents for Three-Dimensional Systems of Ordinary Differential Equations" (2011). Theses, Dissertations and Capstones. 107. https://mds.marshall.edu/etd/107

16. Rabinovich, Mikhail I.; Fabrikant, A. L. (1979). "Stochastic Self-Modulation of Waves in Nonequilibrium Media". Sov. Phys. JETP. 50: 311

17. Rippa, Shmuel. (1999). An algorithm for selecting a good parameter c in radial basis function interpolation. Advances in Computational Mathematics. 11. 193-210. 10.1023/A:1018975909870.

18. Rossler, O.E. (1976) An Equation for Continuous Chaos. Physics Letters, Vol. 57A number 5.

19. Strogatz SH. Nonlinear Dynamics and Chaos. 2nd ed. Boulder, CO: Westview Press; 2015.

20. Wolf, Alan. Swift, Jack. Swinney, Harry. Vastano, John. (1985). Determining Lyapunov Exponents From a Time Series. Physica D: Nonlinear Phenomena. 16. 285-317. 10.1016/0167-2789(85)90011-9.

6 Appendix

6.1 Attractor Plots

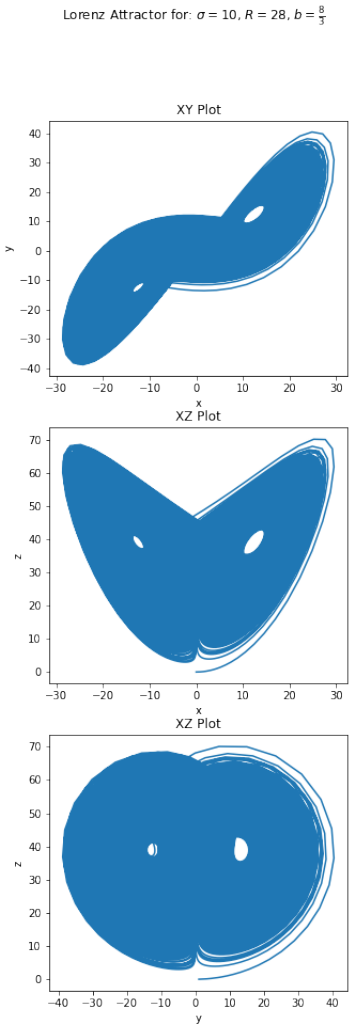

Figure 5: Lorenz

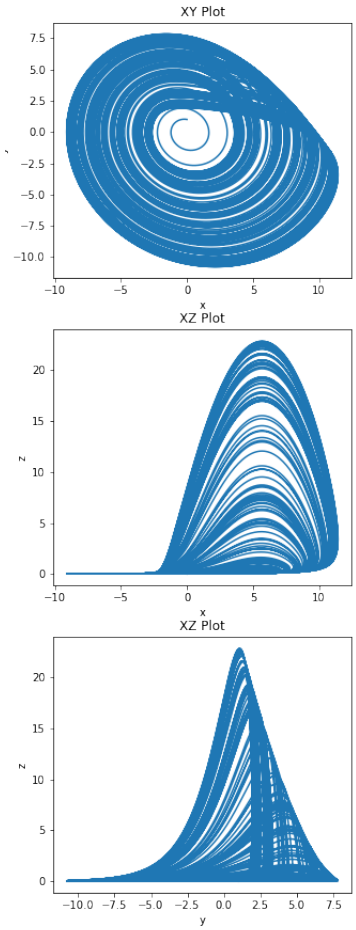

Figure 6: Rossler

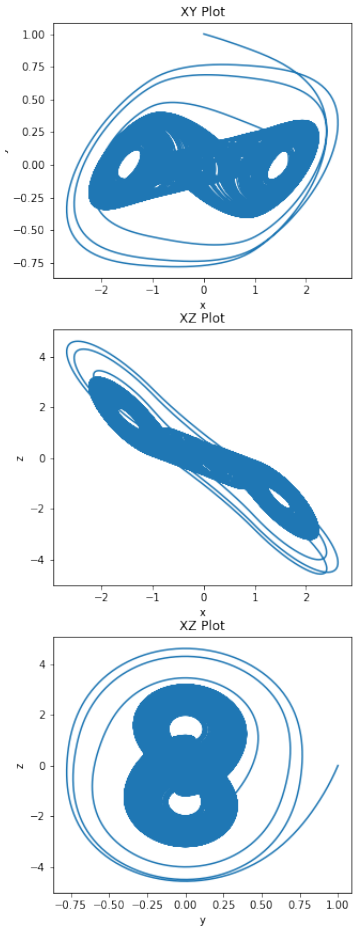

Figure 7: Chua

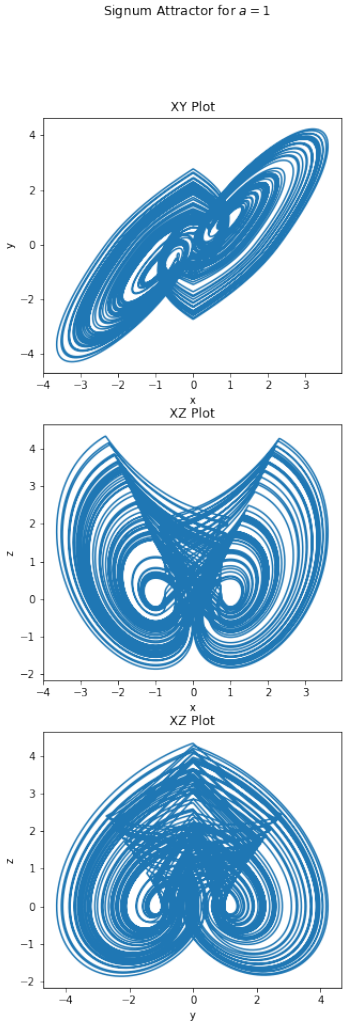

Figure 8: Signum

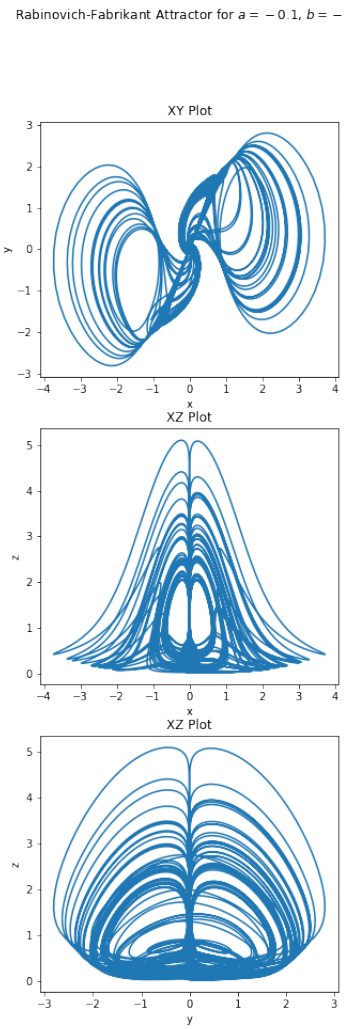

Figure 9: Rabinovich-Fabrikant 1

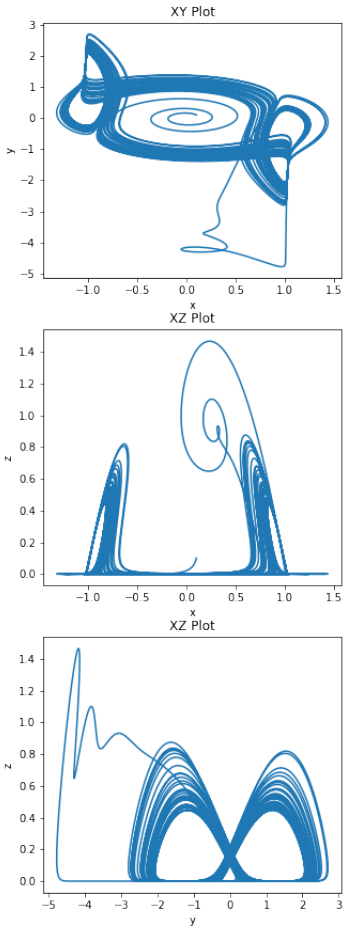

Figure 10: Rabinovich-Fabrikant 2

6.2 Condition Number Plots

6.2.1 Lorenz

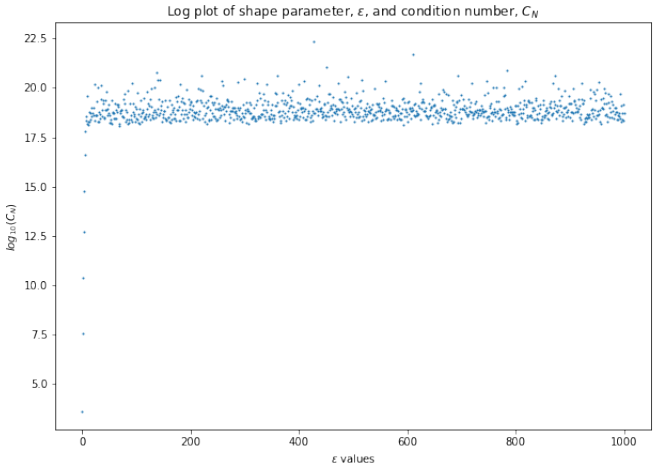

Figure 11: Multiquadric

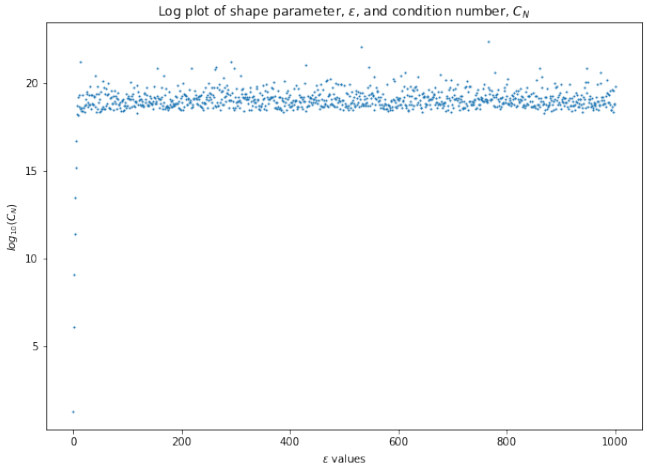

Figure 12: Inverse

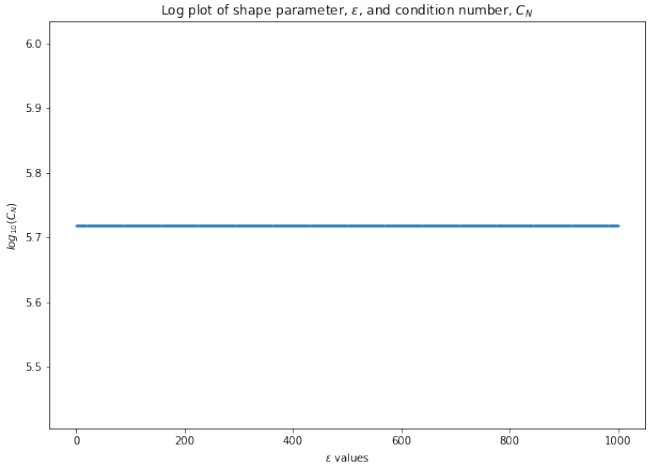

Figure 13: Cubic

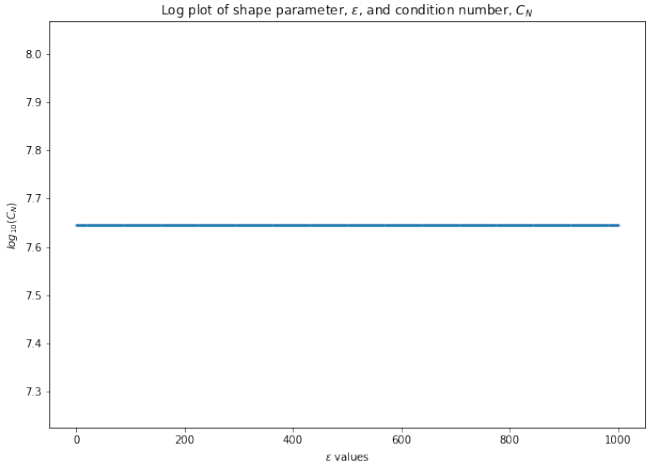

Figure 14: Quintic

6.2.2 Rossler

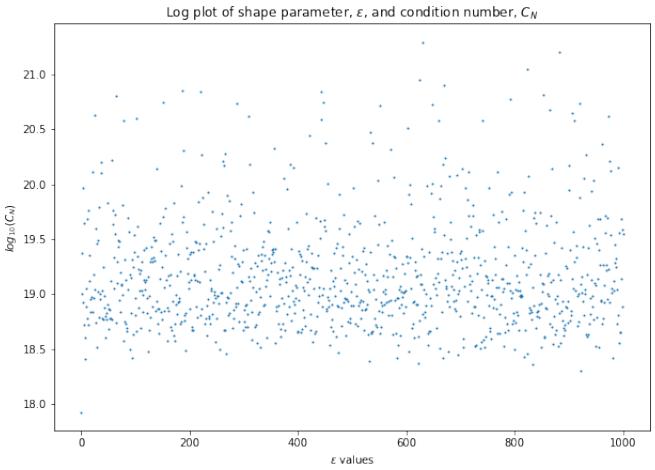

Figure 15: Gaussian

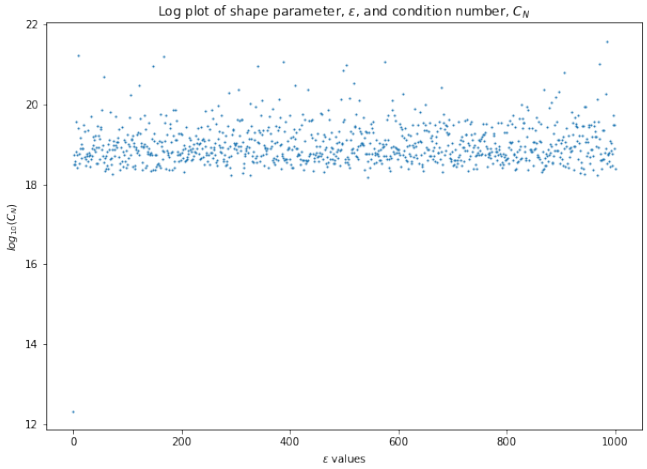

Figure 16: Multiquadric

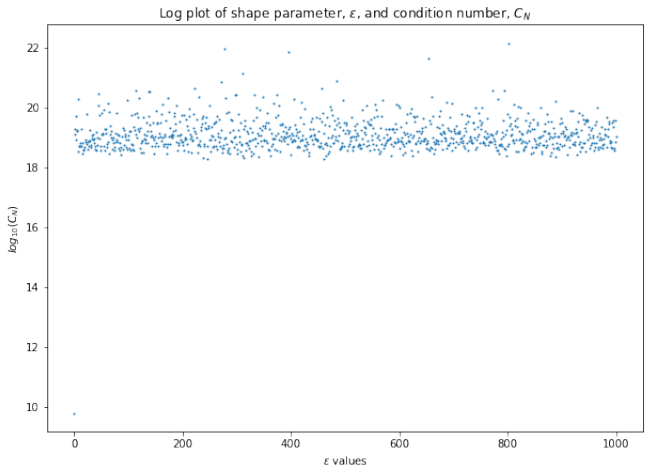

Figure 17: Inverse

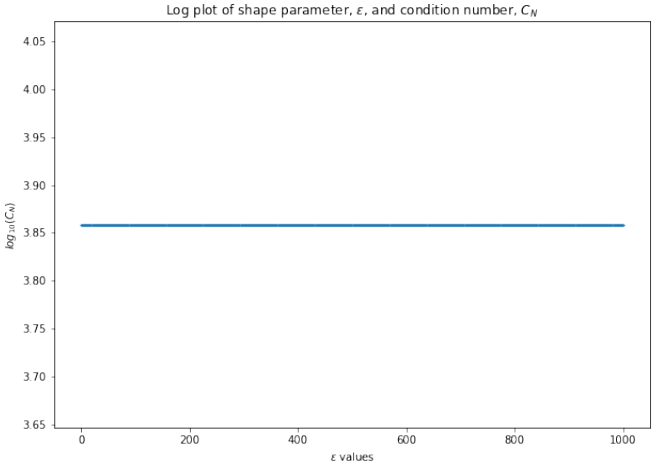

Figure 18: Linear

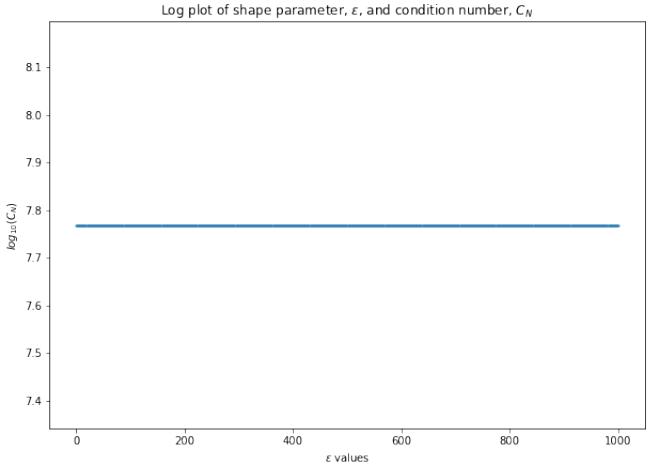

Figure 19: Cubic

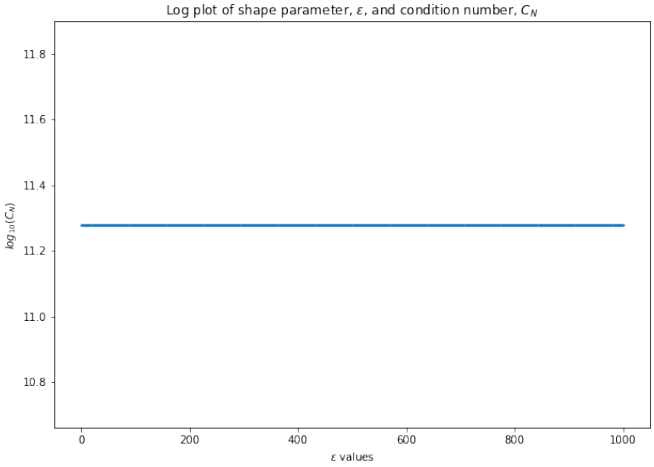

Figure 20: Quintic

6.2.3 Chua

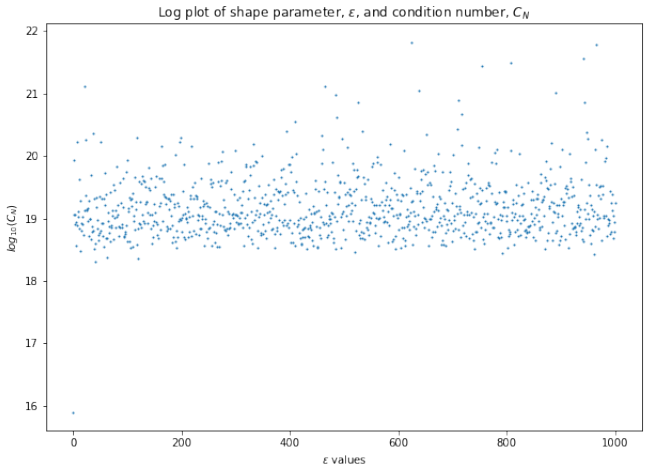

Figure 21: Gaussian

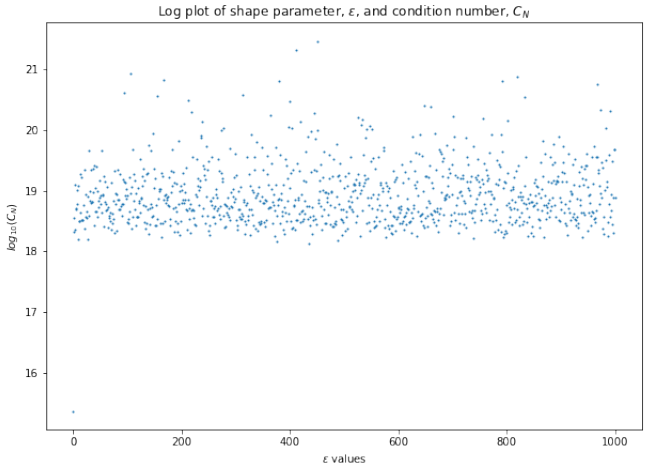

Figure 22: Multiquadric

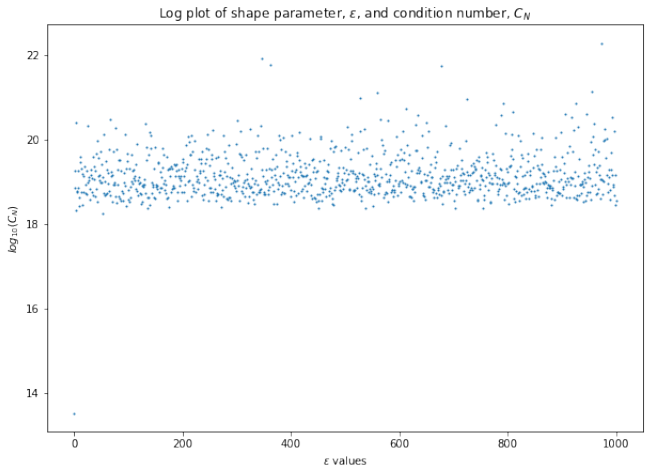

Figure 23: Inverse

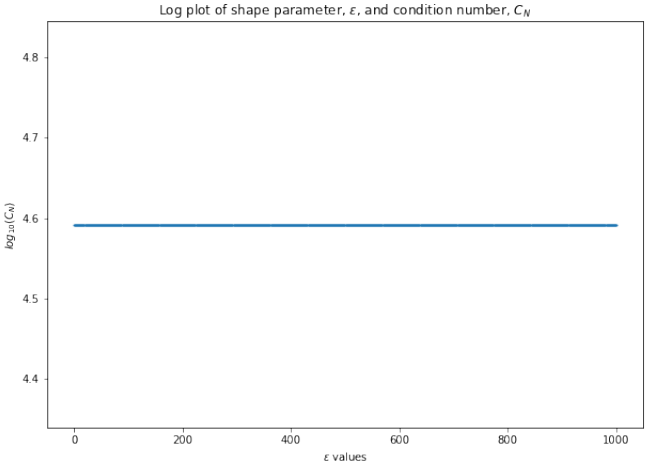

Figure 24: Linear

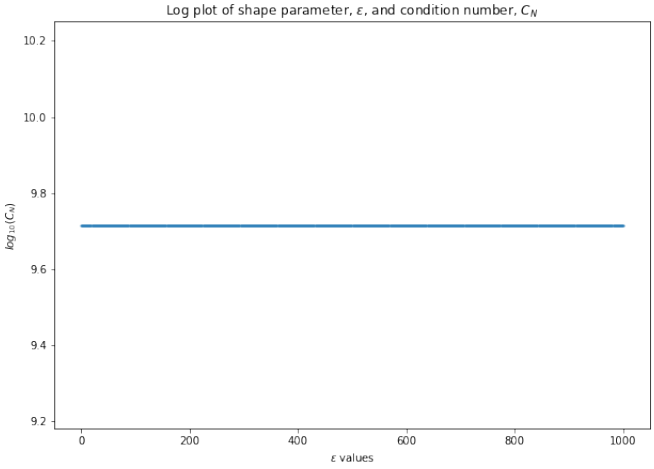

Figure 25: Cubic

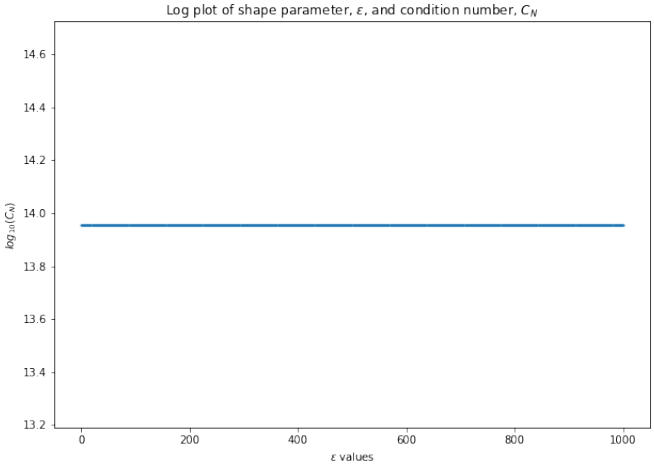

Figure 26: Quintic

6.2.4 Signum

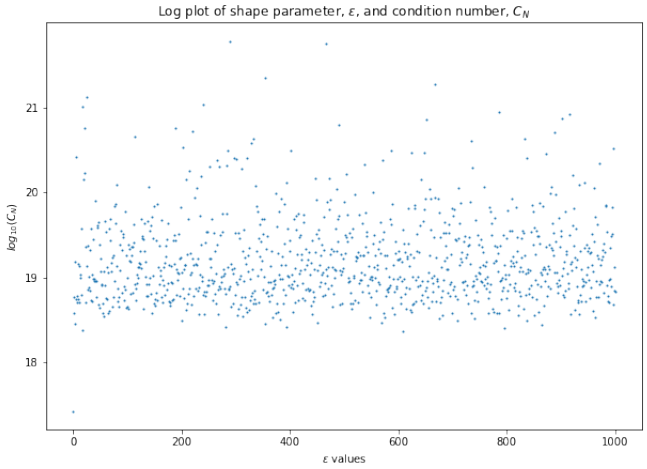

Figure 27: Gaussian

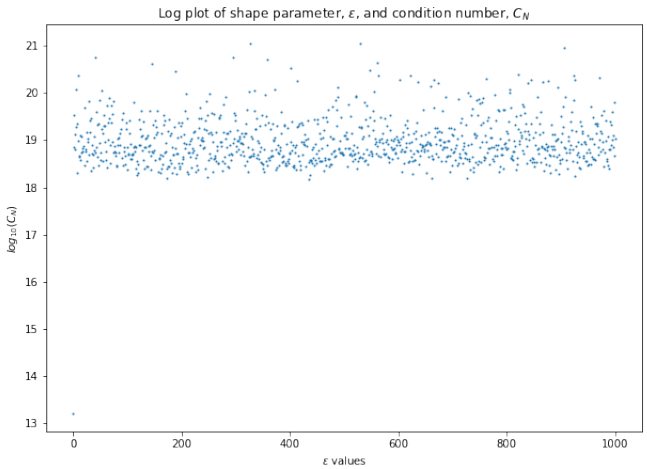

Figure 28: Multiquadric

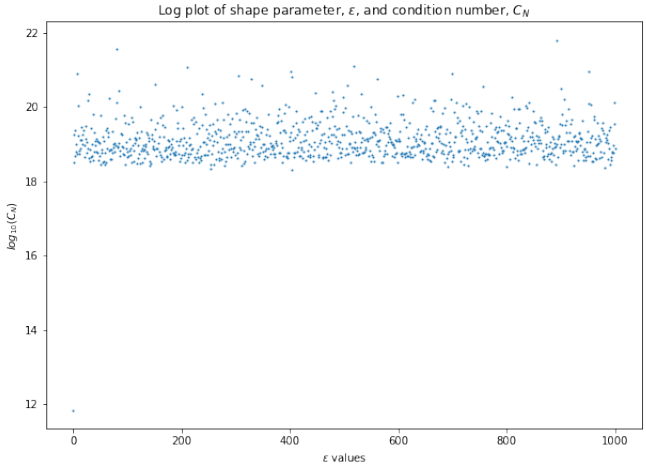

Figure 29: Inverse

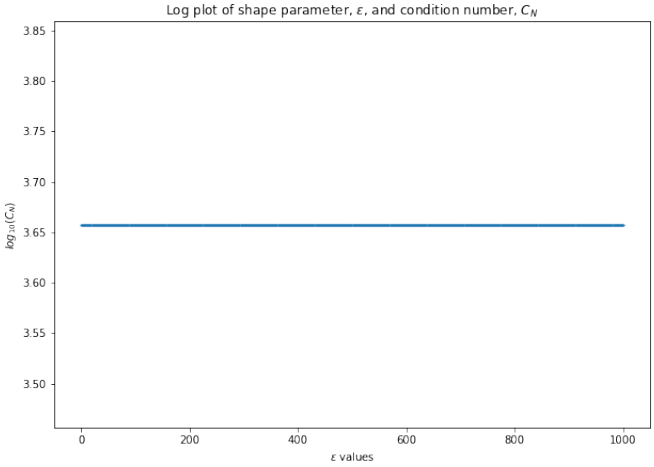

Figure 30: Linear

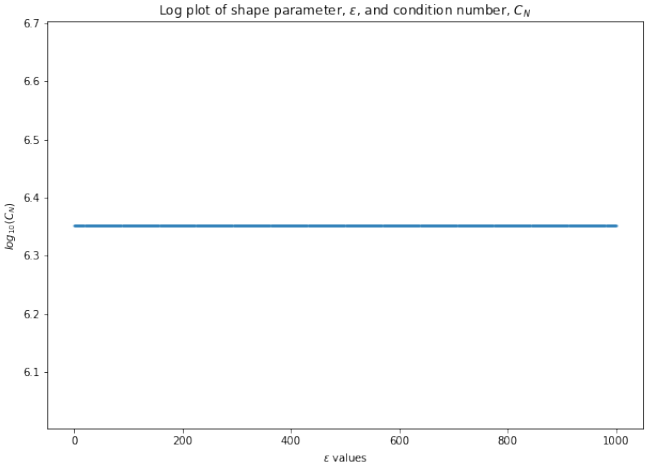

Figure 31: Cubic

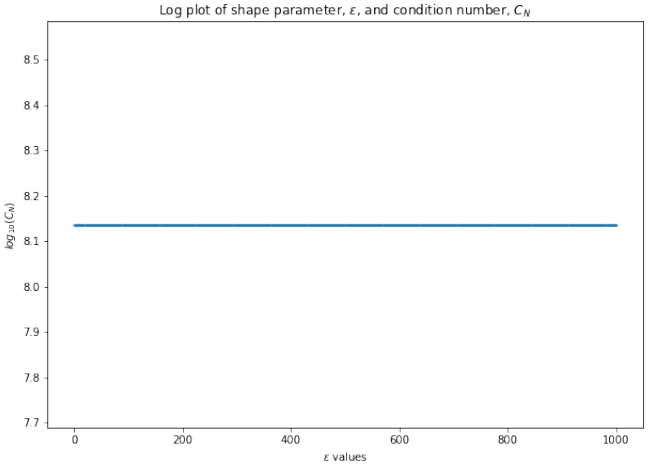

Figure 32: Quintic

6.2.5 RF1

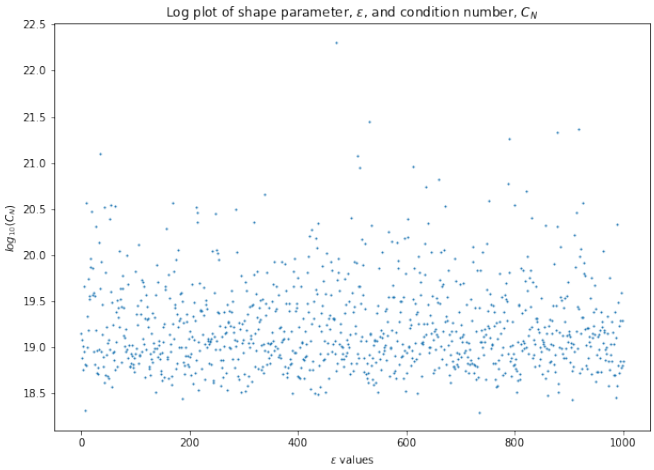

Figure 33: Gaussian

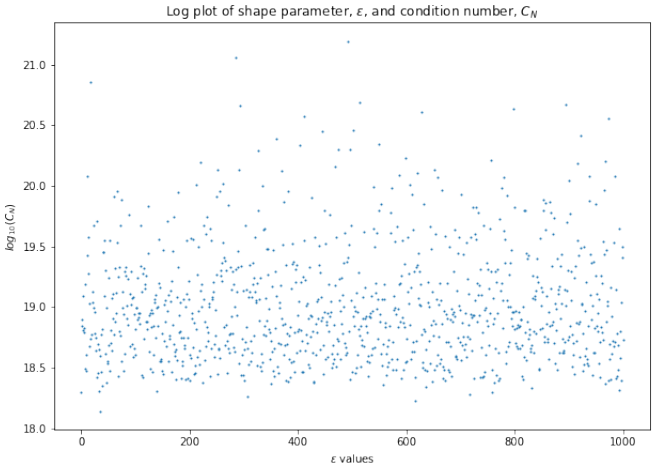

Figure 34: Multiquadric

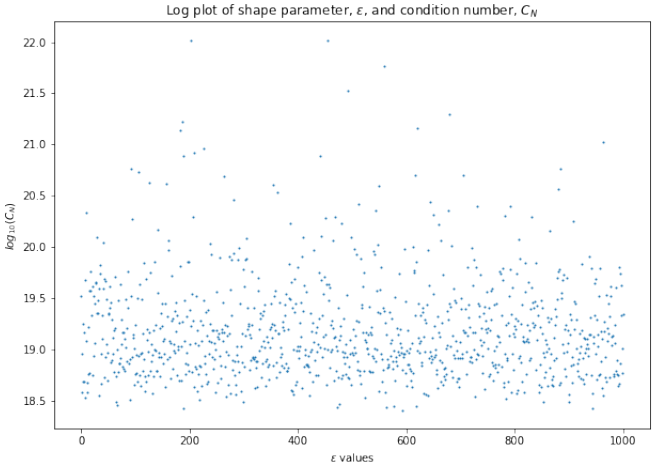

Figure 35: Inverse

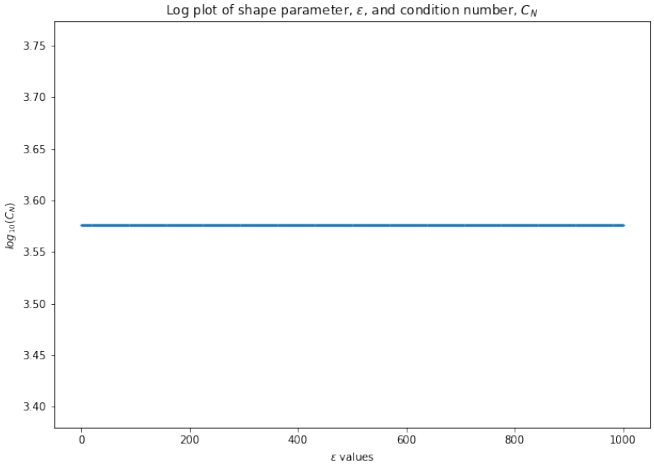

Figure 36: Linear

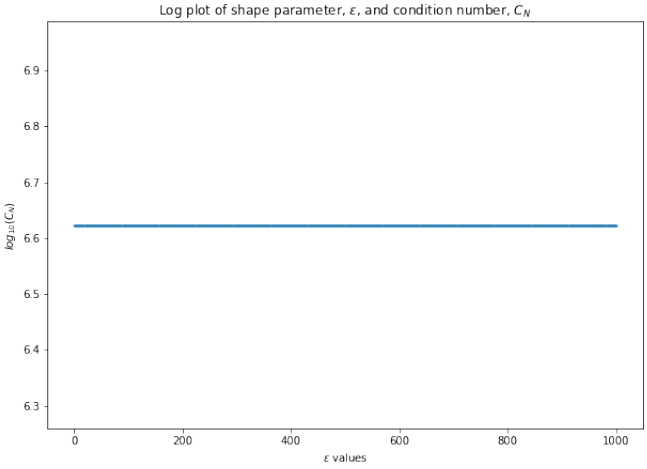

Figure 37: Cubic

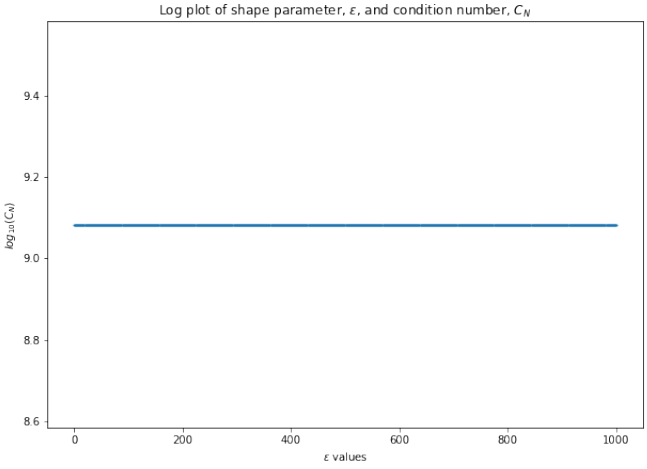

Figure 38: Quintic

6.2.6 RF2

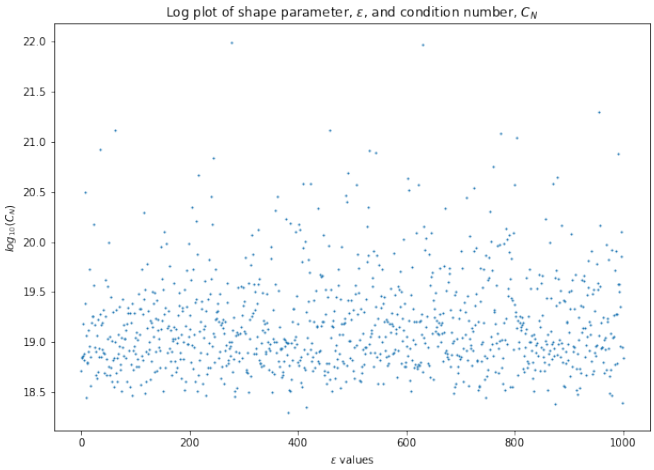

Figure 39: Gaussian

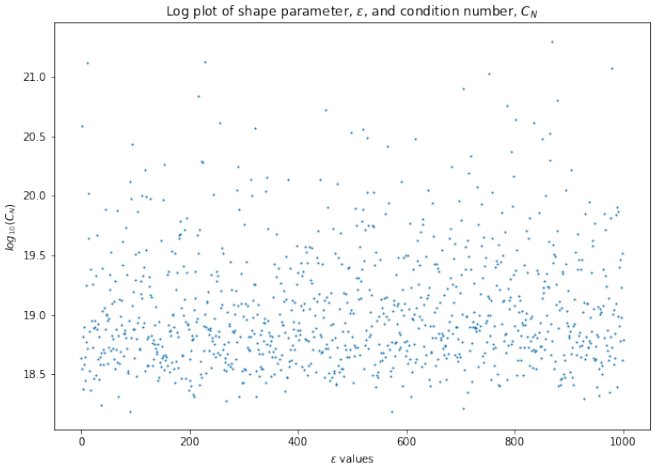

Figure 40: Multiquadric

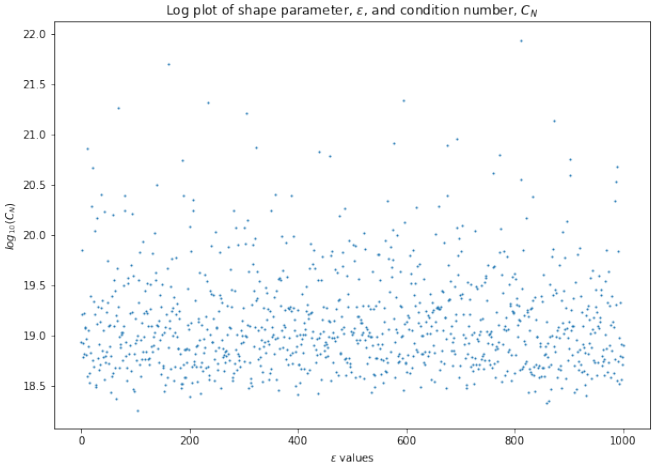

Figure 41: Inverse

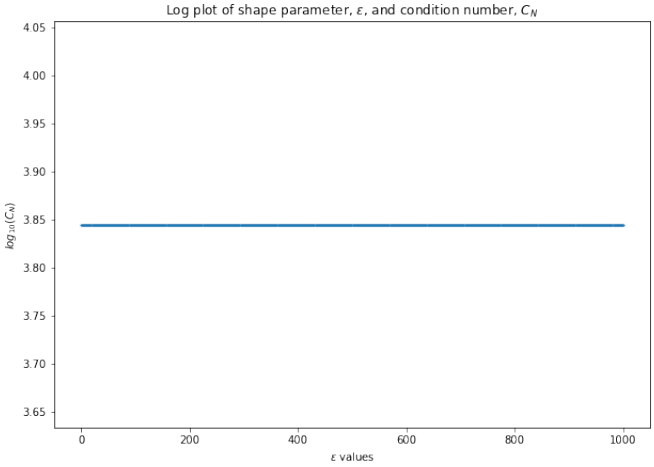

Figure 42: Linear

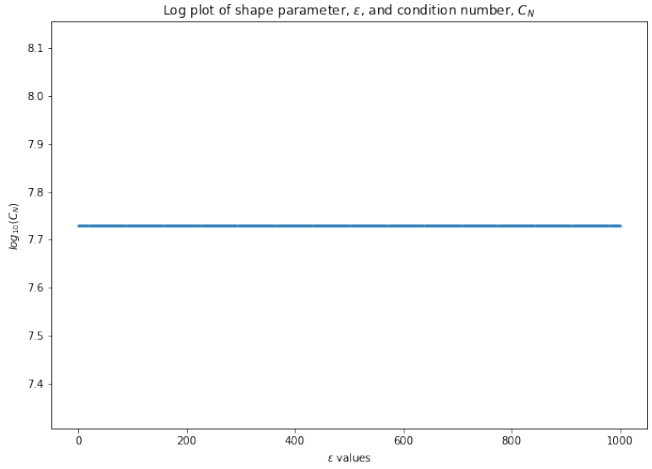

Figure 43: Cubic

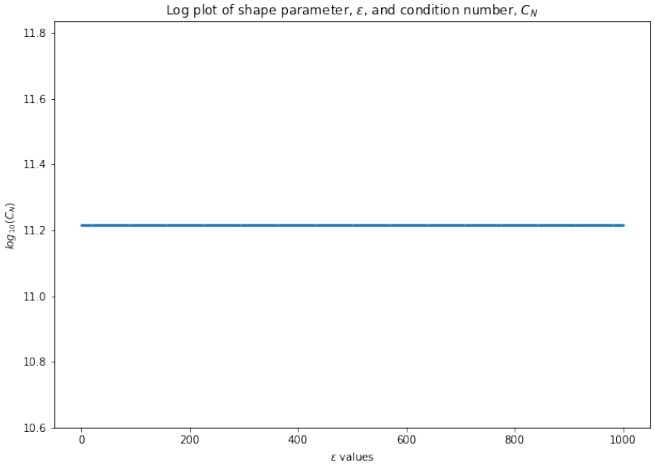

Figure 44: Quintic

6.3 ϵ vs error plots

6.3.1 Lorenz

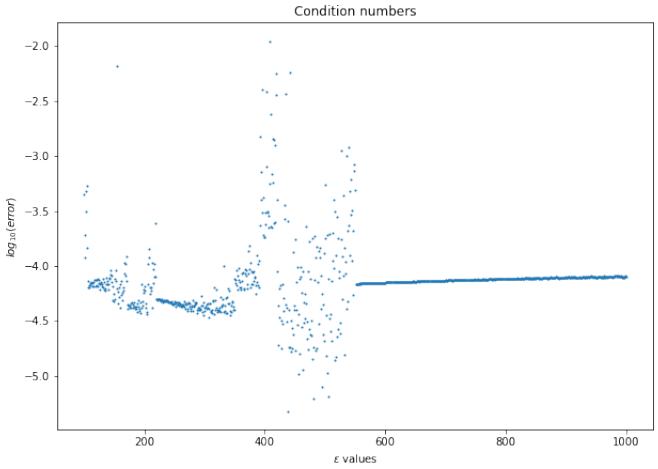

Figure 45: Multiquadric

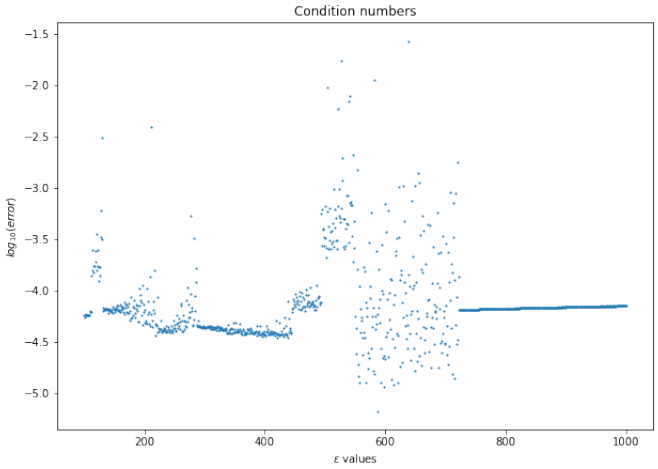

Figure 46: Inverse

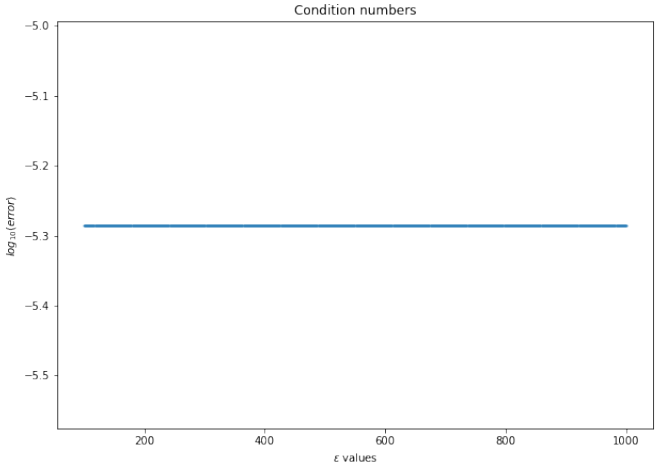

Figure 47: Cubic

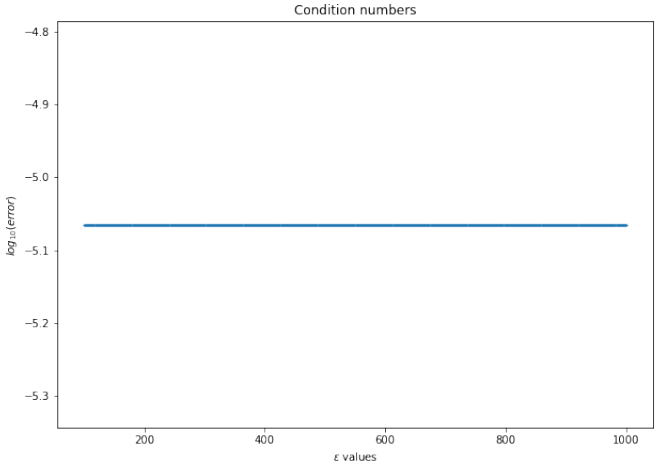

Figure 48: Quintic

6.3.2 Rossler

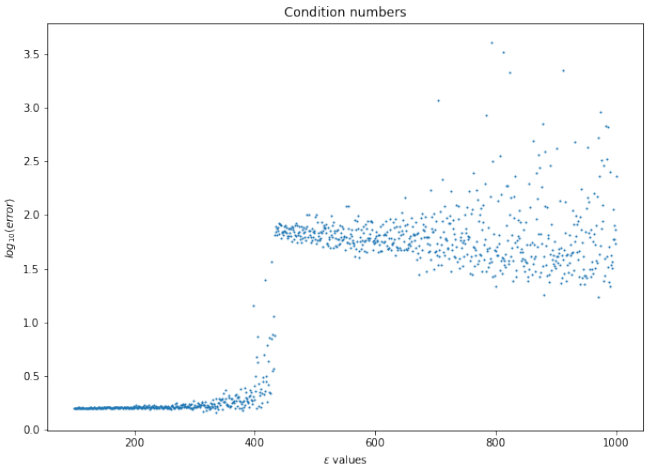

Figure 49: Gaussian

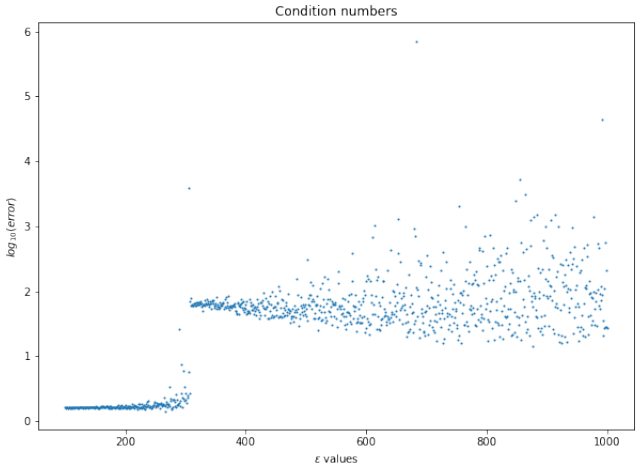

Figure 50: Multiquadric

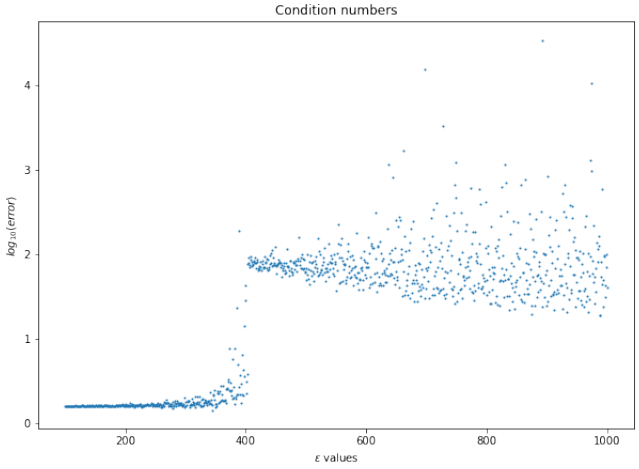

Figure 51: Inverse

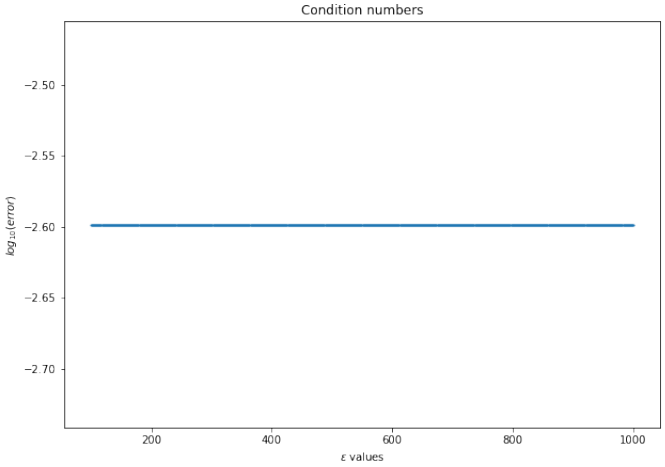

Figure 52: Linear

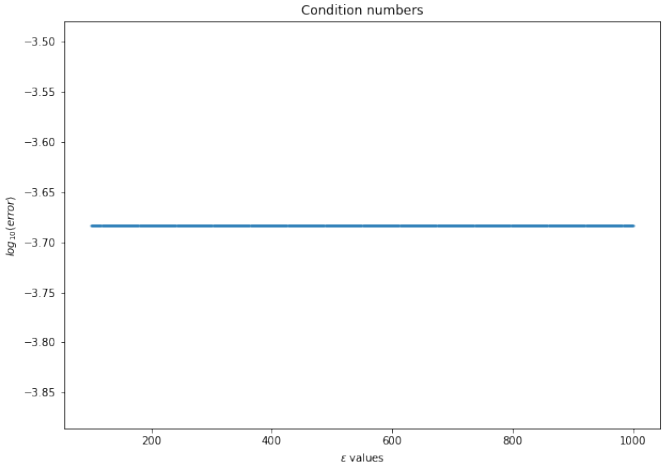

Figure 53: Cubic

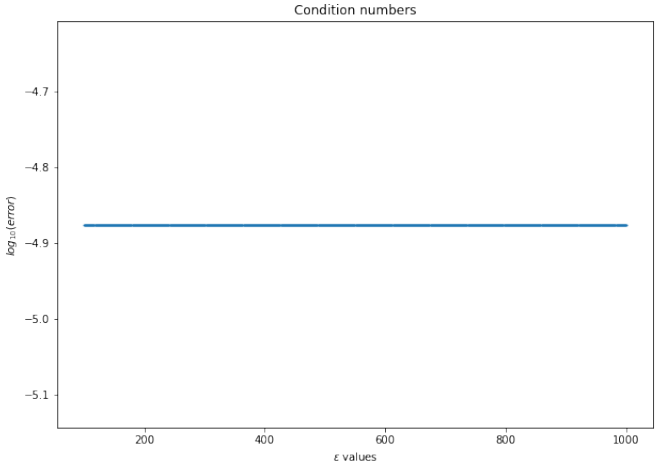

Figure 54: Quintic

6.3.3 Chua

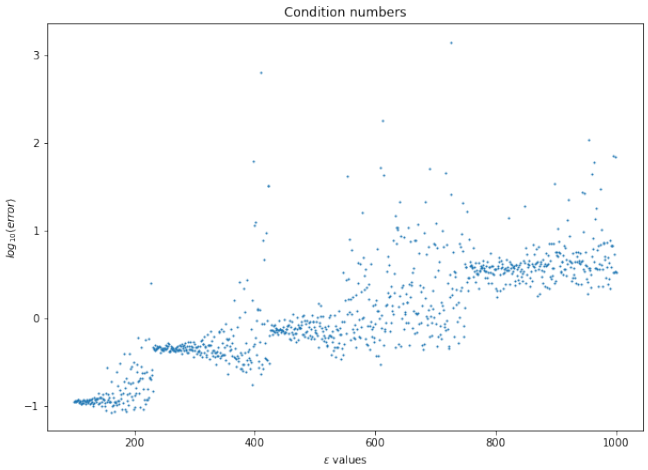

Figure 55: Gaussian

/Epsilon/Chua multiquadric.png

Figure 56: Multiquadric

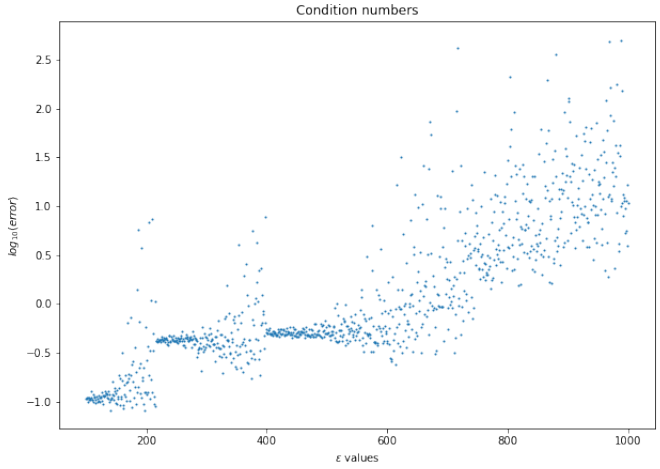

Figure 57: Inverse

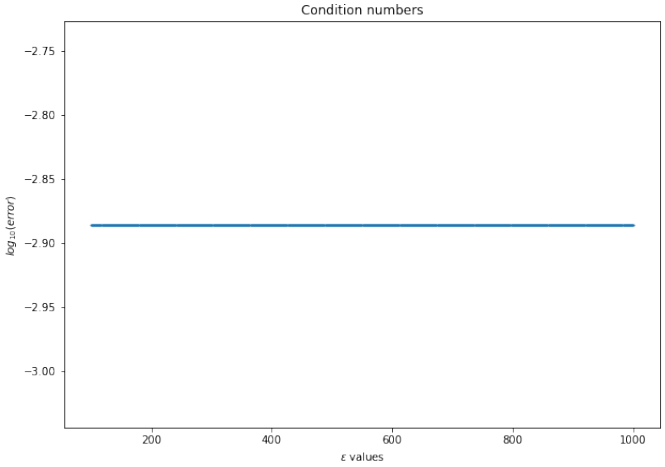

Figure 58: Linear

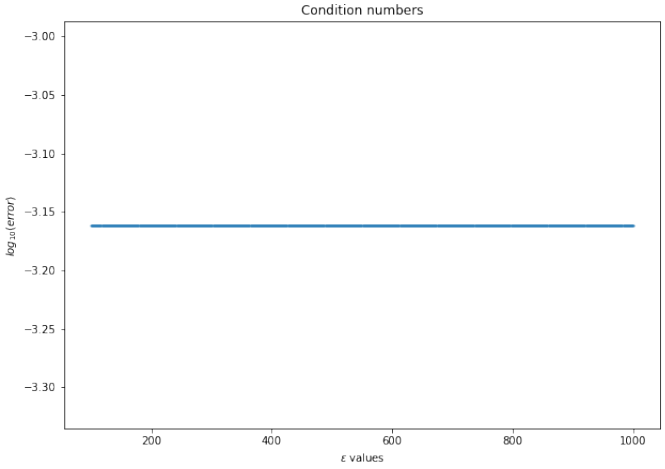

Figure 59: Cubic

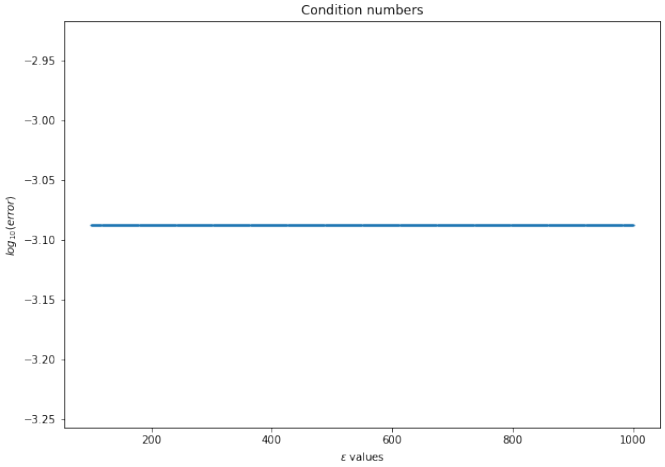

Figure 60: Quintic

6.3.4 Signum

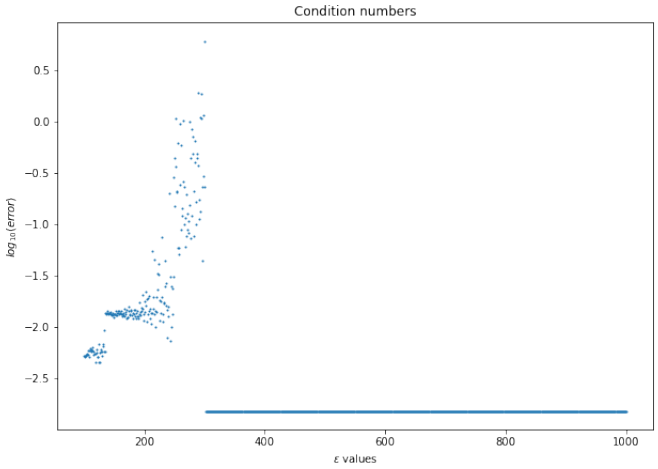

Figure 61: Gaussian

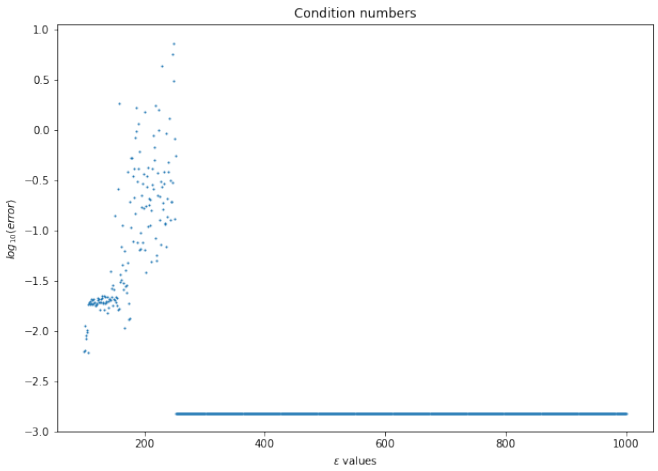

Figure 62: Multiquadric

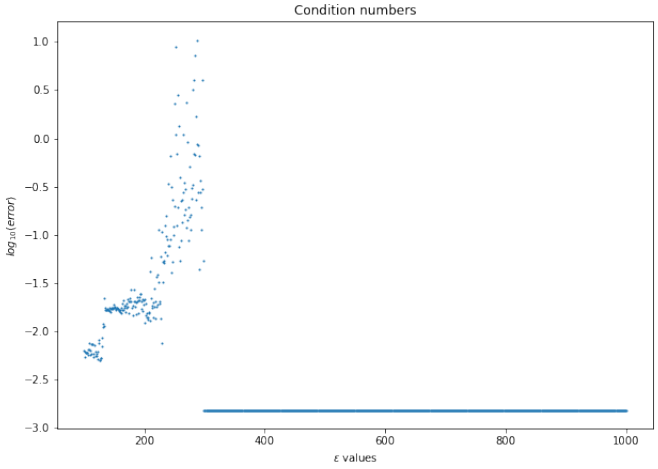

Figure 63: Inverse

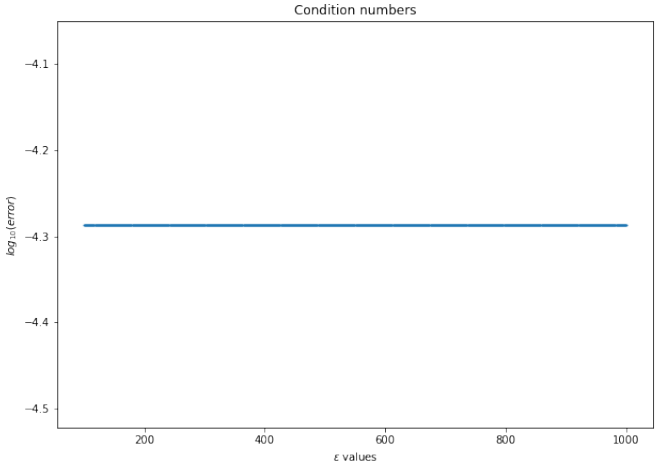

Figure 64: Linear

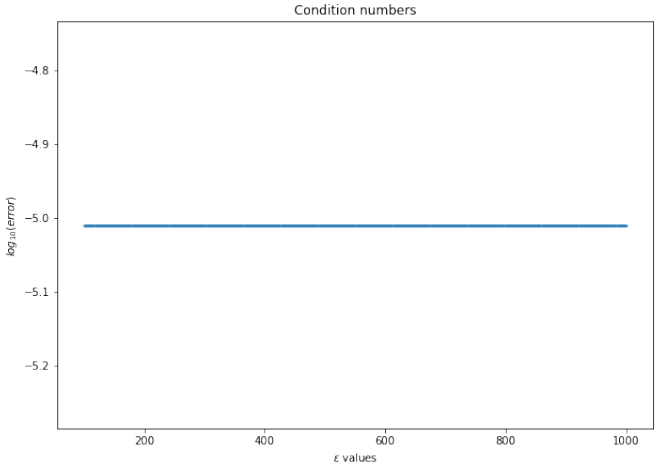

Figure 65: Cubic

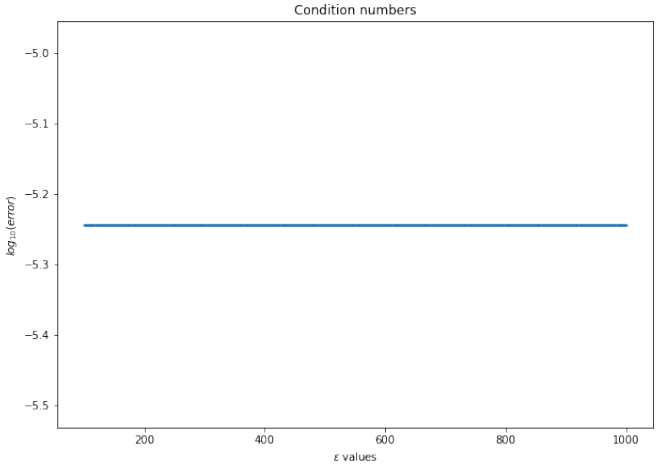

Figure 66: Quintic

6.3.5 RF1

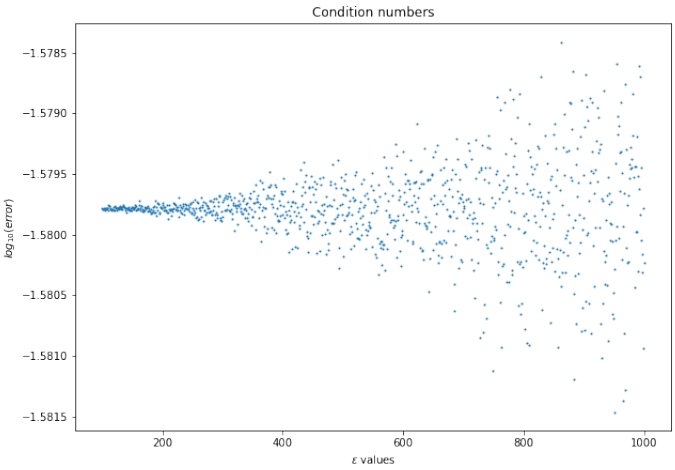

Figure 67: Gaussian

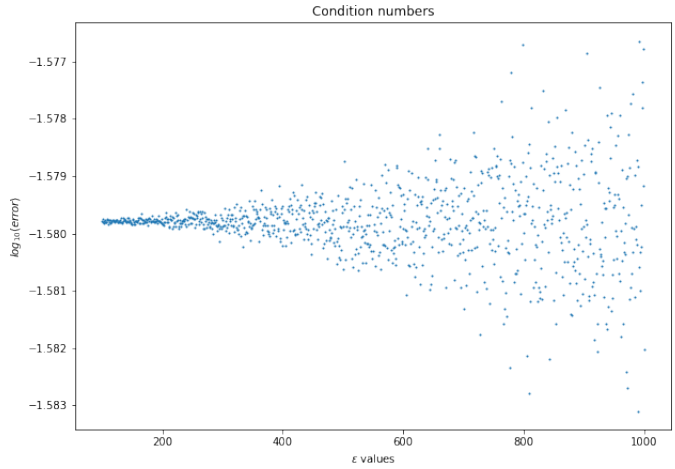

Figure 68: Multiquadric

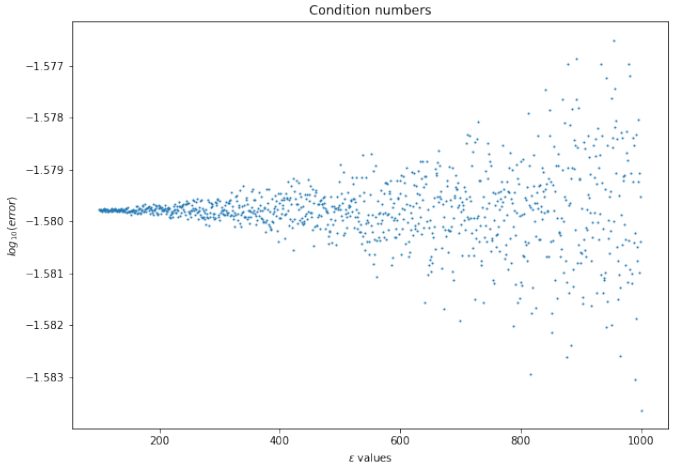

Figure 69: Inverse

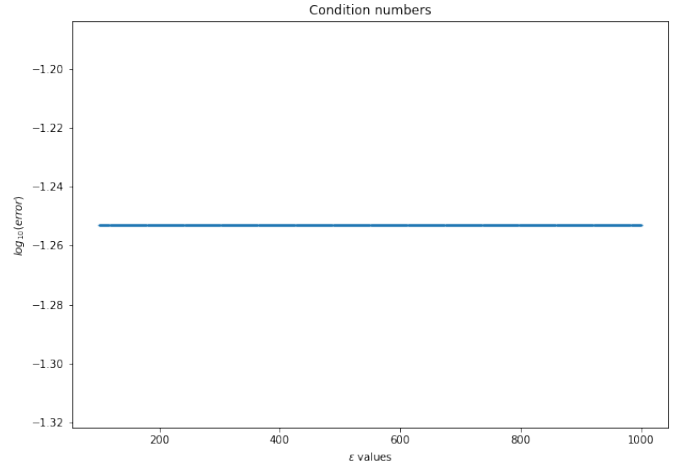

Figure 70: Linear

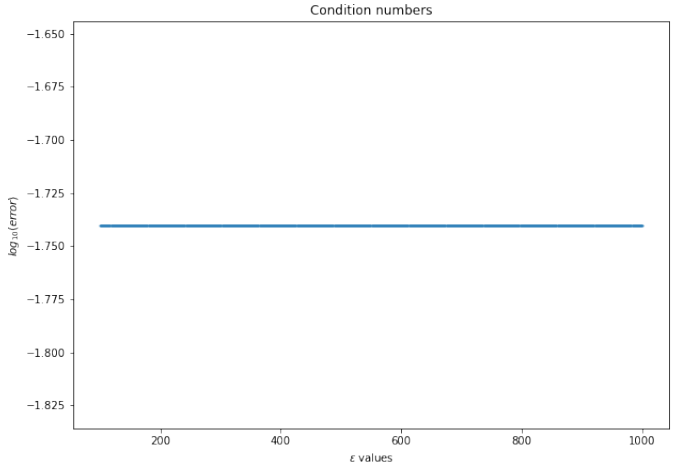

Figure 71: Cubic

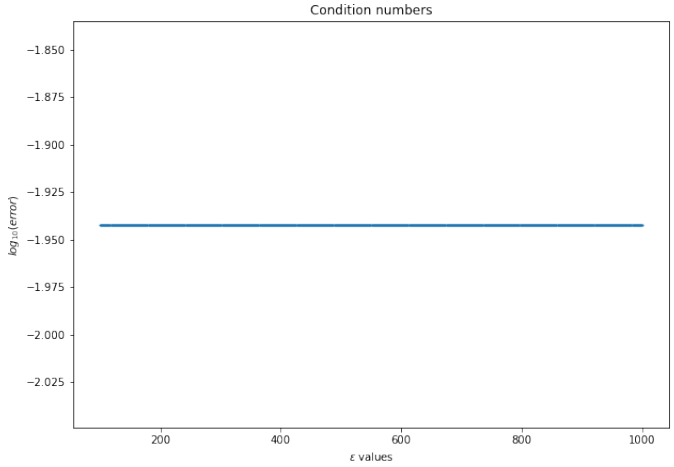

Figure 72: Quintic

6.3.6 RF2

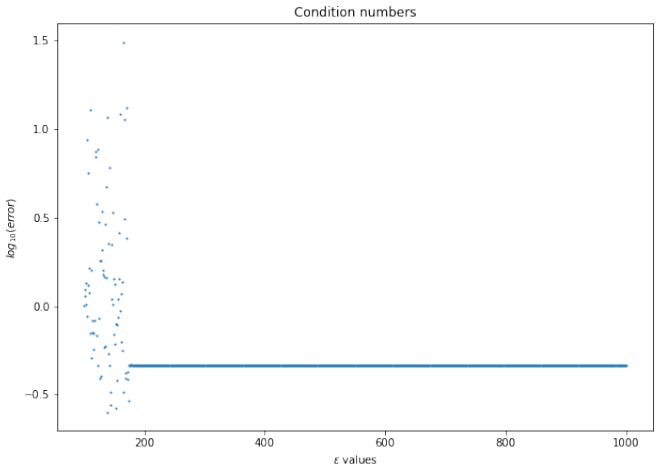

Figure 73: Gaussian

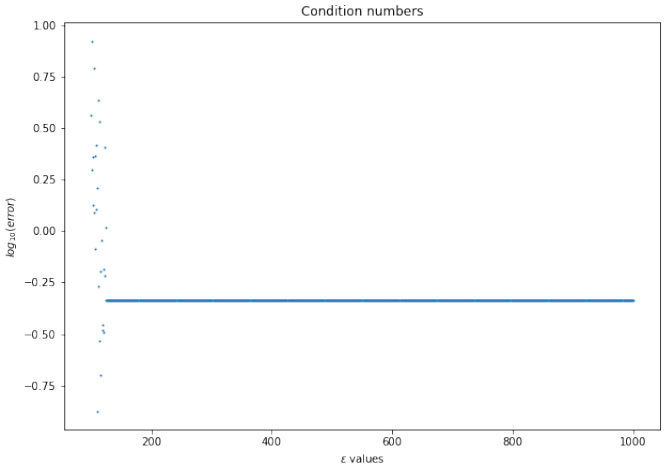

Figure 74: Multiquadric

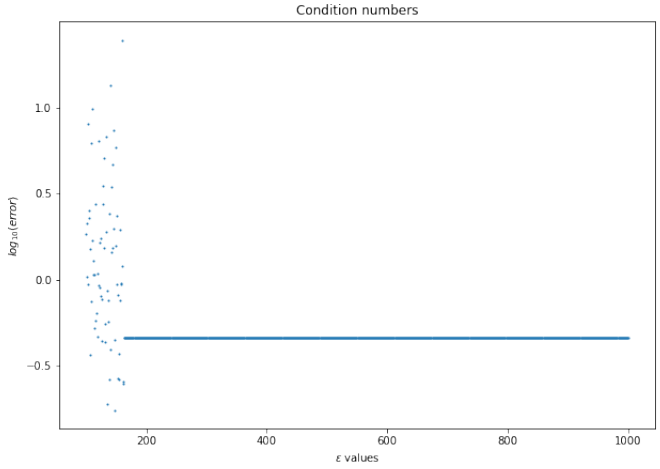

Figure 75: Inverse

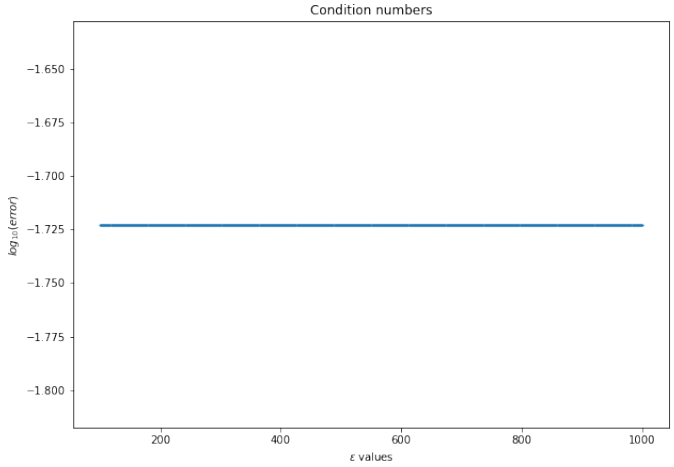

Figure 76: Linear

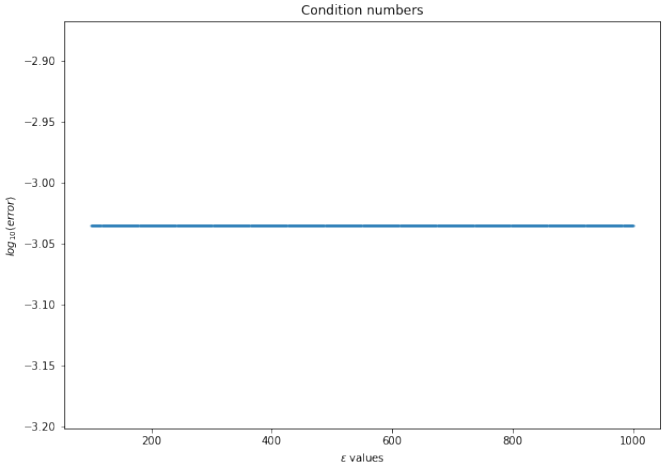

Figure 77: Cubic

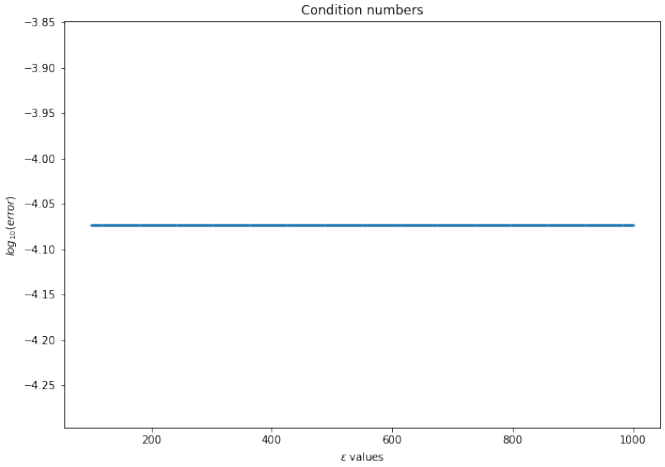

Figure 78: Quintic

6.4 Jacobian

6.4.1 Analytic Jacobians in General Form

$$J(t) = \begin{pmatrix} 0 & -1 & -1 \\ 1 & a & 0 \\ y_3(t) & 0 & y_1(t) - c \end{pmatrix}$$

Rossler Jacobian

$$J(t) = \begin{pmatrix} -a(1 + f'(y_1(t))) & a & 0 \\ 1 & -1 & 1 \\ 0 & -b & 0 \end{pmatrix}$$

$$f'(y_1) = \begin{cases} d & \text{for} \quad |y_1| \geq 1 \\ c & \text{otherwise} \end{cases}$$

Chua Jacobian

$$J(t) = \begin{pmatrix} 1 & -1 & 0 \\ -2y_3\delta(y_1(t)) & 0 & -sgn(y_1(t)) \\ sgn(y_2(t)) & 2y_1\delta(y_2(t)) & 0 \end{pmatrix}$$

Signum Jacobian

$$J(t) = \begin{pmatrix} 2y_2(t)y_1(t) + b & y_3(t) - 1 + y_1^2(t) & y_2(t) \\ 3y_3(t) + 1 - 3y_1^2(t) & b & 3y_1(t) \\ -2y_2(t)y_3(t) & -2y_1(t)y_3(t) & -2a \end{pmatrix}$$

Rabinovich-Fabrikant Jacobian

6.4.2 Exact Numerical Jacobians

$$J(t) = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0.2 & 0 \\ 0.04270733 & 0 & -4.441831 \end{bmatrix}$$

Rossler Jacobian

$$J(t) = \begin{bmatrix} -2.57142857 & 9 & 0 \\ 1 & -1 & 1 \\ [0 & -14.28571429 & 0 \end{bmatrix}$$

Chua Jacobian

$$J(t) = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

Signum Jacobian

$$J(t) = \begin{bmatrix} -1.00000464 & -7.74222986 * 10^{-1} & -2.43218040 * 10^{-3} \\ 1.67732559 & -1.00000000 & 2.85886457 * 10^{-3} \\ 1.09825644 * 10^{-3} & -4.30308873 * 10^{-4} & 2.00000000 * 10^{-1} \end{bmatrix}$$

RF1 Jacobian

$$J(t) = \begin{bmatrix} 8.89165841 * 10^{-2} & -9.99348167 * 10^{-1} & -2.22526028 * 10^{-1} \\ 9.98234351 * 10^{-1} & 1.00000000 * 10^{-1} & 7.47109185 * 10^{-2} \\ 1.40821997 * 10^{-5} & -1.57598654 * 10^{-6} & -1.96000000 \end{bmatrix}$$

RF2 Jacobian

6.4.3 Approximated Jacobians

Lorenz

$$\tilde{J}(t) = \begin{bmatrix} -16.04812754 & 16.00291924 & -0.07300597 \\ -2.95404503 & -0.85013214 & 12.92109608 \\ -10.49598568 & -13.02128017 & -3.91024594 \end{bmatrix}$$

Multiquadric

$$\tilde{J}(t) = \begin{bmatrix} -16.06382566 & 16.01117978 & -0.08011912 \\ -2.79549923 & -0.93416466 & 12.99170684 \\ -10.5943563 & -12.96925189 & -3.95403625 \end{bmatrix}$$

Inverse

$$\tilde{J}(t) = \begin{bmatrix} 0.9254092 & 6.75689194 & 6.29264008 \\ -4.57144641 & -0.18830851 & 10.98751532 \\ -7.16658644 & -14.64684367 & -1.71317161 \end{bmatrix}$$

Linear

$$\tilde{J}(t) = \begin{bmatrix} 4.57985159 & 4.91547367 & 8.51263915 \\ -5.62531897 & 0.5226613 & 11.73918563 \\ -6.07070391 & -15.34611613 & -2.01336146 \end{bmatrix}$$

Cubic

$$\tilde{J}(t) = \begin{bmatrix} 12.91159115 & 0.1850583 & 11.72122653 \\ -5.59566463 & 0.60050831 & 11.86011325 \\ -5.35821154 & -15.833899 & -1.83279406 \end{bmatrix}$$

Quintic

Rossler

$$\tilde{J}(t) = \begin{bmatrix} 1.79622572 * 10^3 & -2.95753330 * 10^2 & -1.50565867 * 10^5 \\ 8.88951491 * 10^3 & -1.45834515 * 10^3 & -7.45063034 * 10^5 \\ 14.9800016 & -2.46211855 & -1.25542970 * 10^3 \end{bmatrix}$$

Gaussian

$$\tilde{J}(t) = \begin{bmatrix} -1.39776540 * 10^4 & -2.07067532 * 10^4 & 3.44693200 * 10^6 \\ -6.91662823 * 10^4 & -1.02460564 * 10^5 & 1.70568717 * 10^7 \\ -1.16521739 * 10^2 & -1.72621357 * 10^2 & 2.87356793 * 10^4 \end{bmatrix}$$

Multiquadric

$$\tilde{J}(t) = \begin{bmatrix} 69.9835206 & -2.32786413 * 10^3 & 2.19006036 * 10^5 \\ 3.47339645 * 10^2 & -1.15140896 * 10^4 & 1.08373455 * 10^6 \\ 5.87221591 * 10^{-1} & -19.4043551 & 1.82585578 * 10^3 \end{bmatrix}$$

Inverse

$$\tilde{J}(t) = \begin{bmatrix} -1.06054177 & 3.43173949 * 10^{-2} & 2.06961883 * 10^{-2} \\ -3.15467203 & 4.29997354 & 1.35333428 * 10^{-1} \\ -5.05456873 * 10^{-3} & 3.28352760 * 10^{-3} & 1.96745786e * 10^{-4} \end{bmatrix}$$

Linear

$$\tilde{J}(t) = \begin{bmatrix} 1.41337992 * 10^2 & -1.40596381 * 10^2 & -3.07195312 \\ 6.99580867 * 10^2 & -6.89723828 * 10^2 & -15.1868408 \\ 1.18273063 & -1.16973244 & -2.55659687 * 10^{-2} \end{bmatrix}$$

Cubic

$$\tilde{J}(t) = \begin{bmatrix} -4.37159343 * 10^3 & 4.31845866 * 10^3 & 67.4005873 \\ -2.16340221 * 10^4 & 2.13772556 * 10^4 & 3.33526529 * 10^2 \\ -36.4394415 & 36.0032976 & 5.61947133 * 10^{-1} \end{bmatrix}$$

Quintic

Chua

$$\tilde{J}(t) = \begin{bmatrix} -2.57291703 & 9.00091731 & -6.38630869 * 10^{-4} \\ 9.99862214 * 10^{-1} & -1.00001732 & 9.998328638 * 10^{-1} \\ 8.59709110 * 10^{-4} & -14.2845127 & 1.96555124 * 10^{-4} \end{bmatrix}$$

Gaussian

$$\tilde{J}(t) = \begin{bmatrix} -2.57083458 & 8.99901411 & 2.44101136 * 10^{-4} \\ 1.00038370 & -1.00050447 & 1.00005694 \\ -1.85930262 * 10^{-3} & -14.2820661 & -9.46202426 * 10^{-4} \end{bmatrix}$$

Multiquadric

$$\tilde{J}(t) = \begin{bmatrix} -2.57281114e & 9.00076920 & -5.74870610 * 10^{-4} \\ 9.99921582 * 10^{-1} & -1.00005220 & 9.99855270 * 10^{-1} \\ 8.82956745 * 10^{-4} & -14.2843896 & 1.63822373 * 10^{-4} \end{bmatrix}$$

Inverse

$$\tilde{J}(t) = \begin{bmatrix} 0.73978075 & 3.508917 & 2.11766257 \\ -0.35084736 & -0.54230817 & 0.62010845 \\ -3.65871129 & -7.61791821 & -2.59044314 \end{bmatrix}$$

Linear

$$\tilde{J}(t) = \begin{bmatrix} -18.71407558 & 25.74232335 & -7.41075773 \\ 5.57804063 & -3.87086153 & 2.62228712 \\ 20.49465546 & -34.64906911 & 9.18100244 \end{bmatrix}$$

Cubic

$$\tilde{J}(t) = \begin{bmatrix} 396.22848162 & -416.8601514 & 186.2043317 \\ -93.92893243 & 95.71041535 & -42.13398506 \\ -534.02338287 & 549.96464581 & -247.80757016 \end{bmatrix}$$

Quintic

Signum

$$\tilde{J}(t) = \begin{bmatrix} -1.00002923 & 1.00001404 & 1.48883000 * 10^{-5} \\ 6.91624668 * 10^{-6} & -1.00730466 * 10^{-5} & -9.99995646 * 10^{-1} \\ 1.00001942 & -1.86233592 * 10^{-5} & -1.79291817 * 10^{-5} \end{bmatrix}$$

Gaussian

$$\tilde{J}(t) = \begin{bmatrix} -1.00000111 & 9.99995242 * 10^{-1} & 2.92605501 * 10^{-6} \\ 5.76446398 * 10^{-6} & -9.44436207 * 10^{-6} & -9.99995329 * 10^{-1} \\ 9.99998575 * 10{-}1 & -4.61615038 * 10^{-6} & -8.96164728e * 10^{-6} \end{bmatrix}$$

Multiquadric

$$\tilde{J}(t) = \begin{bmatrix} -1.00004434 & 1.00002321 & 2.11022441 * 10{-}5 \\ 1.93603025 * 10^{-5} & -1.82244697 * 10^{-5} & -1.00000108 \\ 1.00002364 & -2.08555876 * 10^{-5} & -1.94895757 * 10^{-5} \end{bmatrix}$$

Inverse

$$\tilde{J}(t) = \begin{bmatrix} -0.47380449 & 0.17414827 & -0.62817169 \\ -0.03055163 & -0.14425667 & -1.00830906 \\ 0.62190103 & 0.69668386 & 0.47446241 \end{bmatrix}$$

Linear

$$\tilde{J}(t) = \begin{bmatrix} -0.99547768 & 1.00123644 & 0.00384099 \\ -0.00881607 & 0.00285065 & -1.001823 \\ 1.00128501 & -0.00239026 & -0.00186339 \end{bmatrix}$$

Cubic

$$\tilde{J}(t) = \begin{bmatrix} -1.00345437 & 1.00095992 & -1.05060912 * 10^{-3} \\ -7.81136969 * 10^{-3} & 5.26304047 * 10^{-3} & -9.98140935 * 10^{-1} \\ 1.00688949 & -3.68039131 * 10^{-3} & -2.90699031 * 10^{-4} \end{bmatrix}$$

Quintic

RF1

$$\tilde{J}(t) = \begin{bmatrix} -1.00972544 & -7.76529090 * 10^{-1} & 6.68411064 * 10^{-3} \\ 1.69282276 & -9.98895165 * 10^{-1} & -1.50358219 * 10^{-2} \\ 1.43326675 * 10^{-4} & -3.11890089 * 10^{-4} & 2.00091465 * 10^{-1} \end{bmatrix}$$

Gaussian

$$\tilde{J}(t) = \begin{bmatrix} -9.99548211 * 10^{-1} & -7.74661804 * 10^{-1} & -1.13004432 * 10^{-3} \\ 1.67704048 & -1.00015384 & 1.47179671 * 10^{-3} \\ 3.51775374 * 10^{-4} & -2.15723983 * 10^{-4} & 2.00004162 * 10^{-1} \end{bmatrix}$$

Multiquadric

$$\tilde{J}(t) = \begin{bmatrix} -1.01626176 & -7.76738319 * 10^{-1} & -2.02012974 * 10^{-3} \\ 1.66752830 & -1.00204446 & 1.34063340 * 10^{-3} \\ -6.00394677 * 10^{-4} & -3.14141220 * 10^{-4} & 1.99971317 * 10^{-1} \end{bmatrix}$$

Inverse

$$\tilde{J}(t) = \begin{bmatrix} -0.75769736 & -0.32839828 & 0.00625178 \\ 1.23576002 & -1.22867699 & -0.08229904 \\ 0.00404666 & -2.03997243 & 0.38959716 \end{bmatrix}$$

Linear

$$\tilde{J}(t) = \begin{bmatrix} -1.01424658 & -0.84435181 & 0.0041378 \\ 1.67509359 & -1.1227702 & 0.01386094 \\ -0.02463622 & 0.143143 & 0.18763557 \end{bmatrix}$$

Cubic

$$\tilde{J}(t) = \begin{bmatrix} -9.72581768 * 10^{-1} & -8.02701305 * 10 - 1 & -4.59102693 * 10^{-4} \\ 1.66348092 & -7.36147378 * 10^{-1} & -2.04202041 * 10^{-2} \\ 1.40078916 * 10^{-2} & 1.52610271 & 6.33904304 * 10^{-2} \end{bmatrix}$$

Quintic

RF2

$$\tilde{J}(t) = \begin{bmatrix} 4.32015660 * 10^{2} & -7.13805115 * 10^{3} & 6.82242916 * 10^{5} \\ -6.06902616 * 10^{2} & 1.00448991 * 10^{4} & -9.60195598 * 10^{5} \\ -3.47837304 * 10^{-2} & 5.82746371 * 10^{-1} & -55.8510818 \end{bmatrix}$$

Gaussian

$$\tilde{J}(t) = \begin{bmatrix} 3.57113204 * 10^{3} & -3.34932385 * 10^{4} & 7.41248000 * 10^{5} \\ -5.02494254 * 10^{3} & 4.71376089 * 10^{4} & -1.04323820 * 10^{6} \\ -2.91618690 * 10^{-1} & 2.73929264 & -60.6794385 \end{bmatrix}$$

Multiquadric

$$\tilde{J}(t) = \begin{bmatrix} 1.66051030 * 10^{3} & 4.42395192 * 10^{4} & 3.55098687 * 10^{6} \\ -2.33589301 * 10^{3} & -6.22645532 * 10^{4} & -4.99769848 * 10^{6} \\ -1.35291341 * 10^{-1} & -3.62026245 & -2.90531803 * 10^{2} \end{bmatrix}$$

Inverse

$$\tilde{J}(t) = \begin{bmatrix} 5.47181879 * 10^{-2} & 1.70313900 & 8.26516168 * 10^{-2} \\ 1.03568281 & -3.07451381 & -1.20050235 * 10^{-1} \\ 5.45492143 * 10^{-4} & -4.34505052 * 10^{-5} & -1.06818129 * 10^{-5} \end{bmatrix}$$

Linear

$$\tilde{J}(t) = \begin{bmatrix} 3.14661818 & -2.55036325 * 10^{2} & -1.82401078 \\ -3.29724376 & 3.56958565 * 10^{2} & 2.56661354 \\ 2.94138789 * 10^{-4} & 2.11464425 * 10^{-2} & 1.49121233 * 10^{-4} \end{bmatrix}$$

Cubic

$$\tilde{J}(t) = \begin{bmatrix} -74.6806563 & 6.58095948 * 10^{3} & 32.1543395 \\ 1.06252615 * 10^{2} & -9.26536848 * 10^{3} & -45.2548096 \\ 6.65824798 * 10^{-3} & -5.37853746 * 10^{-1} & -2.63026628 * 10^{-3} \end{bmatrix}$$

Quintic

6.5 Fill Distances

/Fill Distance/Rossler Fill Distance.png

Figure 79: Rossler

/Fill Distance/Chua Fill Distance.png

Figure 80: Chua

/Fill Distance/Signum Fill Distance.png

Figure 81: Signum

/Fill Distance/RF1 Fill Distance.png

Figure 82: RF1

6.6 Lyapunov Estimates

System	λ_1	λ_2	λ_3
Lorenz	1.11408	0.280419	-21.9047
Rossler	0.107945	-0.00615912	-1.2318
Chua	0.384737	-0.0261238	-2.61506
Signum "Li"	0.536935	-0.0137958	-1.45266
RF1	0.407014	-0.0761892	-0.714403
RF2	0.548837	0.0947014	-0.629973

Multiquadric Lyapunov exponent estimates

System	λ_1	λ_2	λ_3
Lorenz	1.08552	0.297218	-21.9361
Rossler	0.115125	-0.0102712	-1.18505
Chua	0.412058	0.000260987	-2.58251
Signum "Li"	0.515629	0.0192148	-1.52953
RF1	0.410951	-0.0781752	-0.68114
RF2	0.556254	0.0793044	-0.576992

Inverse Multiquadric Lyapunov exponent estimates

System	λ_1	λ_2	λ_3
Lorenz	1.13462	0.245332	-5.85492
Rossler	0.122288	-0.0258432	-0.370496
Chua	0.377898	-0.00943354	-1.44748
Signum "Li"	0.169619	-0.0137071	-0.378934
RF1	0.375769	-0.168181	-0.558479
RF2	0.394556	0.0182391	-0.533374

Linear RBF Lyapunov exponent estimates

System	λ_1	λ_2	λ_3
Lorenz	1.22321	0.187201	-11.2226
Rossler	0.0871546	-0.0289435	-0.793241
Chua	0.378835	-0.00438282	-2.52915
Signum "Li"	0.474979	0.00375356	-0.687091
RF1	0.276953	-0.0953691	-0.80238
RF2	0.356797	0.0583849	-0.67947

Cubic RBF Lyapunov exponent estimates

System	λ_1	λ_2	λ_3
Lorenz	1.22028	0.197824	-13.6087
Rossler	0.156222	-0.0264456	-0.797354
Chua	0.646272	-0.0115738	-2.44164
Signum "Li"	1.47271	-0.00476395	-2.06125
RF1	0.44085	-0.0710948	-0.731609
RF2	0.58379	0.0579223	-0.816723

Quintic RBF Lyapunov exponent estimates