Experimental Nonlinear Dynamics Lecture Notes

David Chelidze[1]

Department of Mechanical, Industrial & Systems Engineering

University of Rhode Island, Kingston, Rhode Island

Joseph P. Cusumano[2]

Department of Engineering Science & Mechanics

Pennsylvania State University, University Park, Pennsylvanis

Draft Copy, May 5, 2021

[1]email: chelidze@uri.edu ⋄ phone: 401.874.2356 ⋄ fax: 401.874.2355 ⋄ http://egr.uri.edu/nld/chelidze/

[2]email: jpc3@psu.edu ⋄ phone: 814.865.3179 ⋄ fax: 814.863.7967

Contents

Chapter 1

Introduction

These notes present and explore *experimental analysis* of *dynamical systems*. The first draft of the notes was developed by JPC and later modified and enlarged by DC. The material covered in here is at the graduate level and assumes certain amount of familiarity with basic calculus and linear algebra. Therefore, the details of some methods and concepts covered in undergraduate engineering and mathematics courses will not be provided. We start by describing what we mean by *dynamical systems*.

1.1 Dynamical Systems

Empirically, any collection of *interacting* objects that *changes over time* is a *dynamical system*, if we believe there is some "structure" to that change. Most generally, by "structure" we mean:

1. There is a *phase space* (or *state space*) $\mathcal{S}$ such that "points" or "elements" $x \in \mathcal{S}$ define the **current situation** in the system (i.e., its "state").

   - $\mathcal{S}$ may be a *discrete set* (e.g., symbols in an alphabet) but more typically, $\mathcal{S}$ is a *vector space* (e.g., $x \equiv \mathbf{x} \in \mathbb{R}^n$) or a *manifold* (a "surface" that locally looks like $\mathbb{R}^m$)
   - $\mathcal{S}$ can be finite dimensional or infinite dimensional (for the latter, consider any continuous field)

2. There is some notion of *time*, t. Time can be continuous ($t \in \mathbb{R}$, e.g.) or discrete ($t \in \{t_1, t_2, t_3, \ldots\}$).[1]

3. There is an *evolution law* that acts on $\mathcal{S}$ to take the *current state* into *future states*.

[1] It may be the case that $\Delta t_n = t_{n+1} - t_n \neq \Delta t_m$ for $n \neq m$!

- For a *continuous time* system, the evolution law will be a system of *differential equations* (in theory)

$$\dot{x} = f(x,t) \quad (x \in \mathcal{S}, t \in \mathbb{R}) \tag{1.1}$$

  Note: time dependence can be stochastic (random) and f may be a *partial differential operator* if Eq. (1.1) is a partial differential equation.

- For *discrete time* systems, it will be a *map*

$$x_{n+1} = f(x_n, n) \quad (x_n \in \mathcal{S}, n \in \mathbb{Z}) \tag{1.2}$$

  Note: t_n can be mapped on n.

> Thus, empirically, the notion of a **dynamical system** is a *hypothesis* on the object (or collection of objects) that you are observing.

Indeed the most *fundamental* type of experiment aims to learn about items 1 and 3 above (item 2 is usually clear from context):

- What is $\mathcal{S}$? In particular, what is its *dimension*?
- What can we say about the *evolution law*? Is it *linear* or *nonlinear*? Is it *deterministic* or *stochastic*?

> The three hypotheses listed above are the foundation of a *dynamical systems perspective* (DSP). Concepts like "stability," "chaos," "fractals," "attractors," etc. are *consequences* of this point of view, but *not essential* to it!

Main consequences of the dynamical systems perspective are:

1. *Trajectories* of the system (i.e., "solutions" to Eq. (1.1) or Eq. (1.2)) lie on (or near with small noise) *curves* in $\mathcal{S}$
2. Curves and surfaces left *invariant* by the action of the system play an *important role* in the dynamics
3. *Asymptotic behavior* (stability) can be precisely defined using a notion of distance ($\|x - y\|$) between points $x, y \in \mathcal{S}$

In short, the DSP converts the study of *time evolution* into the study of *geometrical properties* (at least to some extent).

1.2 Some Examples of Dynamical Systems

In this section we present several sample dynamical systems and discuss what type of questions will DSP be used to answer.

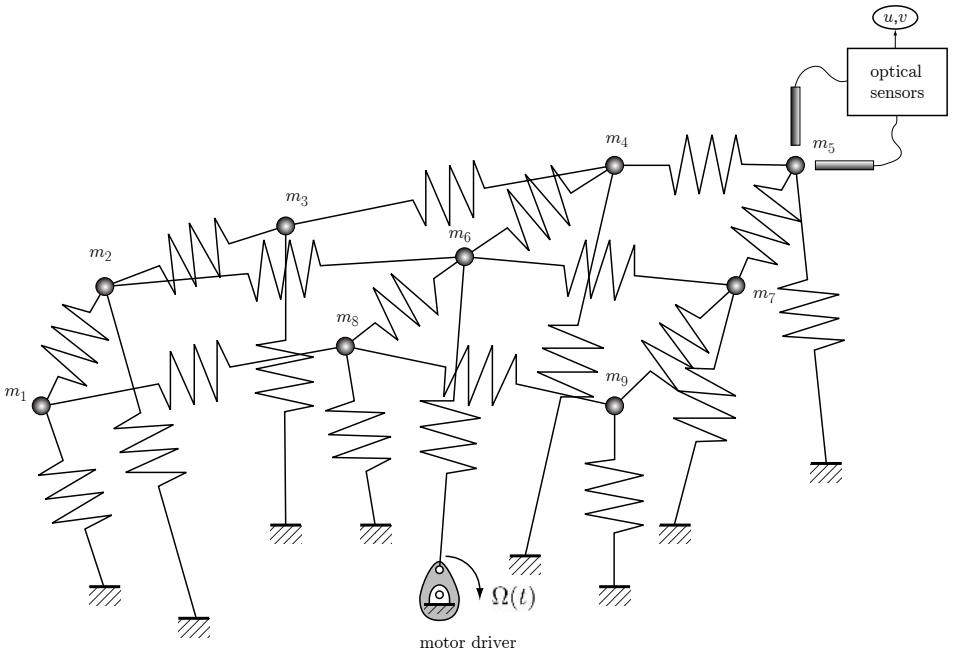

Figure 1.1: Schematic drawing of a mass-spring-model

1.2.1 Linear Mass-Spring System

Imagine that we have a collection of nine masses (m_i, $i = 1, \ldots, 9$) connected by springs and excited by some motor driver as shown in Fig. 1.1. All masses are constrained to move in a plane and we can measure the vertical and horizontal displacements of mass m_5 by means of optical sensors.

The dynamics of this system can be modeled by a system of linear ordinary differential equations, if we assume that the excitation amplitude and mass motions are sufficiently small:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}(t)\mathbf{x} + \mathbf{b}(t)\,, \\ \mathbf{u} &= \mathbf{g}(\mathbf{x})\,, \end{aligned} \tag{1.3}$$

where $\mathbf{A}(t) \in \mathbb{R}^{36\times 36}$ is usually called a state transitions matrix and its time dependence indicates parametric forcing, $\mathbf{b} \in \mathbb{R}^{36}$ is an external forcing function, $\mathbf{x} = [x_1, y_1, x_2, y_2, \ldots, \dot{x}_1, \dot{y}_1, \ldots]^T \in \mathbb{R}^{36}$ is a state variable, $\mathbf{g} : \mathbb{R}^{36} \to \mathbb{R}^2$ is the observation function, generating observations $\mathbf{u} = (x_5, y_5)^T$ in this particular case. Experimentally, we only have observations from the optical sensors sampled at some discrete time steps:

$$\begin{bmatrix} u_1 & u_2 & u_3 & \cdots \\ v_1 & v_2 & v_3 & \cdots \end{bmatrix} \tag{1.4}$$

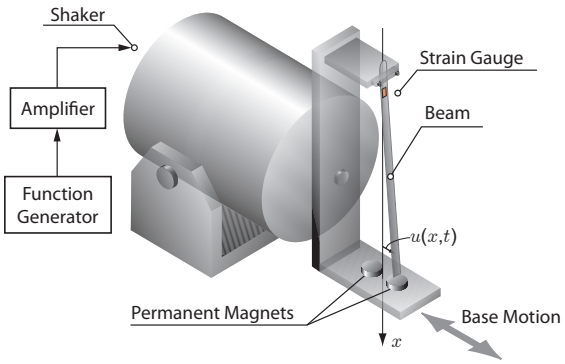

Figure 1.2: Schematic drawing of Moon-Holmes oscillator

Then the question is that if we can determine the mapping from one observation to another:

$$\begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} \triangleq \mathbf{u}_{n+1} = \mathbf{C} \begin{bmatrix} \mathbf{u}_n \\ \mathbf{u}_{n-1} \\ \vdots \\ \mathbf{u}_{n-m} \end{bmatrix} + \text{OT} \tag{1.5}$$

where $\mathbf{C} \in \mathbb{R}^{2\times(m+1)}$, and $\text{OT} \in \mathbb{R}^2$ indicates other possible terms such as bias. Other questions might be: what is the value for m? Can it be just 0? Is the dimensionality of $\mathbf{C}$ $(m+1)$ indicative of the dimensionality of $\mathcal{S}$? What can we say about $\mathcal{S}$ and its evolution law?

1.2.2 Moon-Holmes Oscillator

Let us consider the classic nonlinear dynamics experiments first investigated by Moon and Holmes [70] shown in Fig. 1.2. The apparatus consists of a cantilever beam fixed into a horizontally moving frame, where the beam's free end is buckled by a two-well potential (see Fig. 1.3) realized by two permanent earth magnets. The frame is driven by an electromagnetic shaker that can provide harmonic force excitation. From physics, we expect the partial differential equation (PDE) in $u(x,t) \in \mathcal{S}$ to model the transverse oscillations of the beam. Therefore, our system is infinite dimensional (u is a scalar field variable).

The dynamics of this system is measured using a strain gauge attached to the beam near its clamped end. The gauge provides strain amplified voltage $s(t)$ or $\{s_1, s_2, s_3, \ldots\}$, a *scalar* signal or time series. The objectives then could be: What is the actual form of the double well? What parameters determine the variation in dynamical behavior? Can we infer the dynamical properties of u such as its effective dimensionality, the corresponding predictive model, etc.

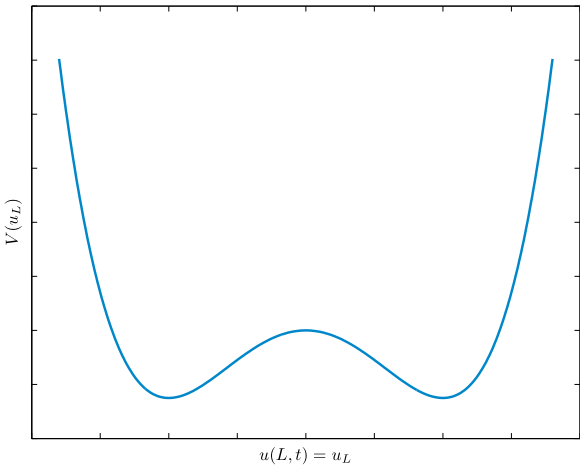

Figure 1.3: Double-well potential near the free end the the Moon beam

1.2.3 Chua-Matsumoto Circuit

Chua-Matsumoto circuit [66] is one of the simple electronic circuits that exhibit chaotic behavior. A version of the circuit is shown in Fig. 1.4, where the variables used in the following model are indicated near the components (in capital letters):

$$\dot{x} = \alpha(y - x - f(x)) \tag{1.6}$$

$$\dot{y} = x - y + z \tag{1.7}$$

$$\dot{z} = -\beta y \tag{1.8}$$

where the function $f(x)$ describes the behavior of the nonlinear negative resistor realized in the right part of the circuit, and parameters α and β can be altered by varying particular values of the circuit components R and C1. In this circuit the dashed rectangle encompasses elements that form

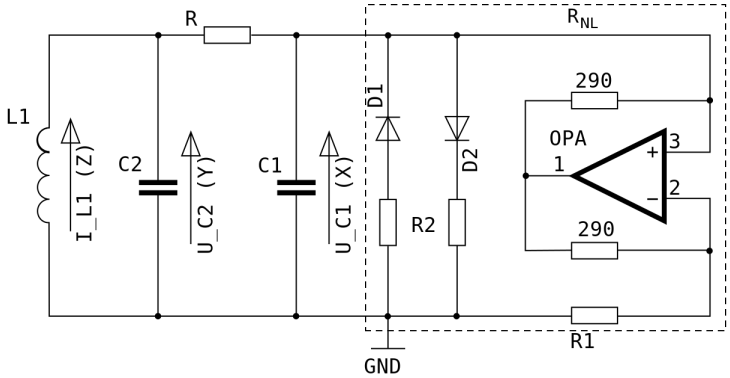

Figure 1.4: A version of the Chua-Matsumoto circuit

a *nonlinear negative* resistor R_{NL}, [2] which can be represented by the graph on Fig. 1.5.

[2] Usually $V = RI$, or voltage equals resistance times current, where $R > 0$.

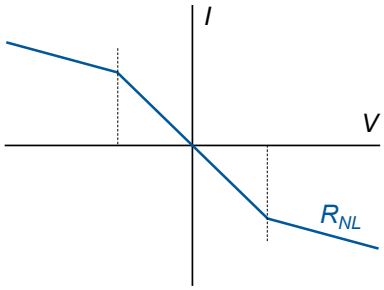

Figure 1.5: The current-voltage characteristic of the Chua-Matsumoto nonlinear resistor R_{NL}

1.2.4 Shooting an Arrow at a Target

Now let us consider a task of shooting an arrow at a target illustrated in Fig. 1.6. Here the objective of the task is to hit a bullseye. After each try we observe the error e in targeting and adjust our arrow release angle, tension in the string, or other targeting mechanisms. Thus we observe a sequence of $\{e_1, e_2, e_3, \ldots\}$, and ask a basic question: Is there a "first principles" model that can describe the targeting dynamics (i.e., learning dynamics)? Or more simply is there some general functional dependance between successive errors in targeting, $e_{n+1} = f(e_n, n)$? Further, we can investigate which parameters need more strict control for accurate targeting, and which are not that critical to control.

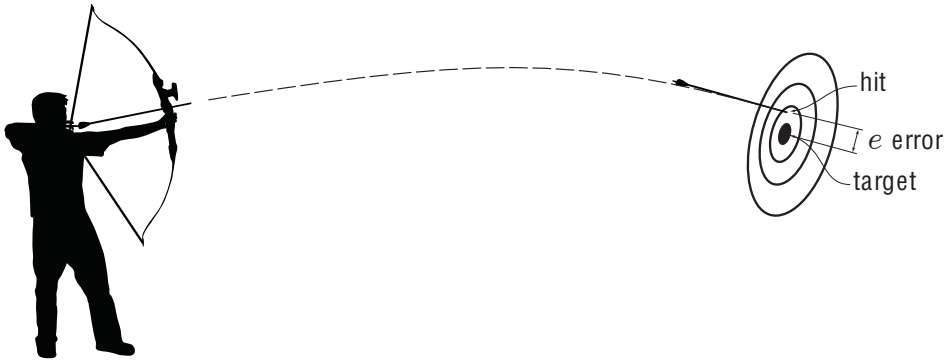

Figure 1.6: A schematic of shooting an arrow at a target

1.2.5 Fly Population Dynamics

One other interesting example of a nonlinear map model is the fly population dynamics model [10]. For example, consider a population of flies that lives in a confined space (e.g., a jar) and we control the amount or rate of the food supply. If we denote the population or number of n-th generation of flies as x_n, then a linear model of the population dynamics can be expressed as:

$$x_{n+1} = a x_n \,, \tag{1.9}$$

where the parameter a depends, for example, on the available food supply. However, we see the problem of the linear model Eq. (1.9): if $a > 0$, then $x_n \to \infty$ as $n \to \infty$ (unbounded growth); and if $a < 0$, then $x_n \to 0$ as $n \to 0$ (total extinction). Thus, for a linear model, we have no nonzero steady state population of flies!

To look for a nonzero steady state solution one may start to consider introducing nonlinearity into Eq. (1.9), e.g.:

$$x_{n+1} = ax_n + bx_n^2\,, \tag{1.10}$$

where the parameter b can be used to control the nonlinear cost of population growth. Or in general, we may look for a more appropriate model $x_{n+1} = f(x_n)$, where the form of f can be ascertained from the experimental observations or other physical/biological arguments.

There are many other examples of dynamical systems of interest such as commodity price dynamics, global warming, or protein folding.

1.3 Experimental Observation and Measurement Function

The difference between *discrete* and *continuous* systems is not as strong as it seems at first. Let us consider a continuous dynamical system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \tag{1.11}$$

with a solution starting from an initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$:

$$\mathbf{x}(t) = \mathbf{X}(t; t_0, \mathbf{x}_0) \tag{1.12}$$

then let us denote $\mathbf{x}_1 = \mathbf{X}(t_1; t_0, \mathbf{x}_0)$, $\mathbf{x}_2 = \mathbf{X}(t_2; t_0, \mathbf{x}_0)$, and

$$\mathbf{x}_{n+1} = \mathbf{X}(t_{n+1}; t_0, \mathbf{x}_0) \triangleq \mathbf{F}(\mathbf{x}_n, n). \tag{1.13}$$

or, thinking in terms of a numerical algorithm:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{f}(\mathbf{x}_n, t_n)\Delta t_n, \tag{1.14}$$

which is Euler step iteration. In this course, we will tend to think of all systems as "discrete" at the level of observations.

As we saw in the examples we do not in general observe $\mathbf{x} \in \mathcal{S}$ *directly*. Instead, we usually have:

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{f}(\mathbf{x}_n, n)\,, \quad \mathbf{x}_n \in \mathcal{S}\,, \\ \mathbf{u}_{n+1} &= \mathbf{g}(\mathbf{x}_{n+1})\,, \quad \text{(measurement system)} \end{aligned} \tag{1.15}$$

where $\mathbf{u}_n$ does not even have the same dimension as $\mathbf{x}_n$. Graphically we can picture $\mathbf{g}$ mapping as shown in Fig. 1.7. We usually like $\mathbf{g}$ to be *linear* (i.e. $\mathbf{u}_{n+1} = \mathbf{C}\mathbf{u}_n$), but at minimum we need it to be one-to-one.

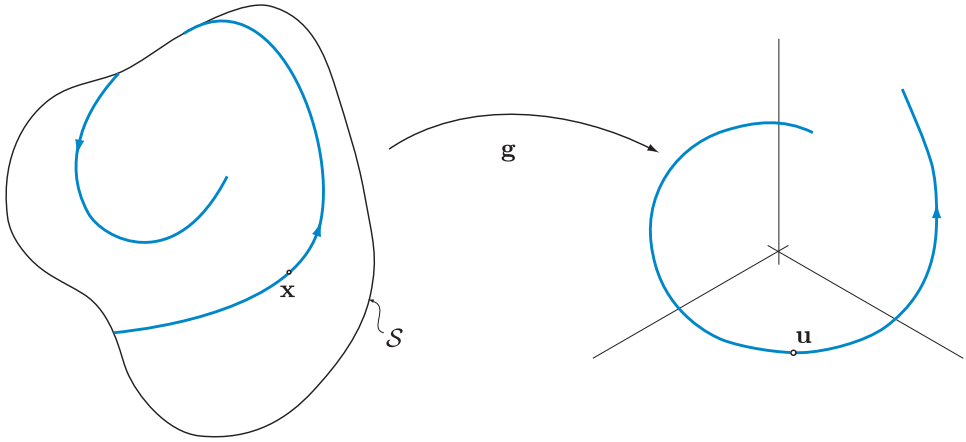

Figure 1.7: Mapping from the phase space to the measurement space

1.4 Experimental Data Analysis and Noise

Experimental data always contains some form of noise. Even the cleanest data sets have digitization noise. Additive noise can enter as process noise $\mathbf{p}(n)$:

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, n) + \mathbf{p}(n) \tag{1.16}$$

or as measurement noise $\mathbf{q}(n)$:

$$\mathbf{u}_{n+1} = \mathbf{g}(\mathbf{x}_{n+1}) + \mathbf{q}(n) \tag{1.17}$$

The challenge then is to learn as much about the system of equations Eq. (1.16) as we can given only the measurement time series $\{\mathbf{u}_n\}_{n=0}^{N}$. We do this with two general types of data analysis:

- Characterization:
  - What does the system do? How does it behave in general?
  - What is its "structure"?
- Prediction:
  - What will the system do in the future?
  - What is a good model for it?

Problems

In this class we will be using MATLAB extensively. We will start by familiarizing ourselves to simulations of continuous and discrete systems. Please refer to *ordinary differential equations* section in MATLAB documentation for simulating odes (https://www.mathworks.com/moler/odes.pdf). Specifically, how to choose an ODE solver in MATLAB and what options are available.

Problem 1.1

Simulate free response of damped harmonic oscillator

$$\ddot{x} + 2\zeta\dot{x} + x = 0$$

for different values of damping ration ζ and initial conditions. Plot the response $x(t)$ and $\dot{x}(t)$ versus time t and the corresponding trajectories in the phase space for underdamped, critically damped, and overdamped systems.

Do the same for the harmonically forced system:

$$\ddot{x} + 2\zeta\dot{x} + x = f\cos\omega t$$

for some choice of f and $\omega \neq 1$ and plot the solutions including transient response and some part of the steady state.

Problem 1.2

Simulate two-well Duffing oscillator

$$\ddot{x} + 0.25\,\dot{x} + x(x^2 - 1) = 0.3\cos t$$

using MATLAB's ode45 algorithm and using initial conditions $x(0) = -0.67079530230$ and $\dot{x}(0) = -0.36933661878$ over the time interval $t \in [0, 100]$. Sample the trajectory with the uniform sampling time interval $t_s = 0.1$. Now assume that our measurement from this system is $u_i = x_i$, and do the following:

- Plot the original systems phase space trajectory, by plotting x_i versus $\dot{x}_i$.
- Now plot our measurements versus their time-delayed version: x_i versus x_{i+12}.

How do these two plot compare? Can you say that our measurement can be used to reconstruct the original system's trajectory?

Chapter 2

Probability, Estimators, and Stationarity

Consider a signal generated by a dynamical process, $x(t) \in \mathbb{R}$, $t \in \mathbb{R}$. Considering $x(t)$ as a function of time t, we are operating in the *time domain*. A fundamental way to *characterize* the dynamics using the time domain data is in terms of *probability*. Here, we assume the existence of a *probability density* $p(x)$ such that

$$\text{Prob}\{a \leq x \leq b\} = \int_a^b p(x)\, dx\,, \tag{2.1}$$

with a constraint that $\int_{-\infty}^{\infty} p(x)\, dx = 1$ (refer to Fig. 2.1 for illustration).

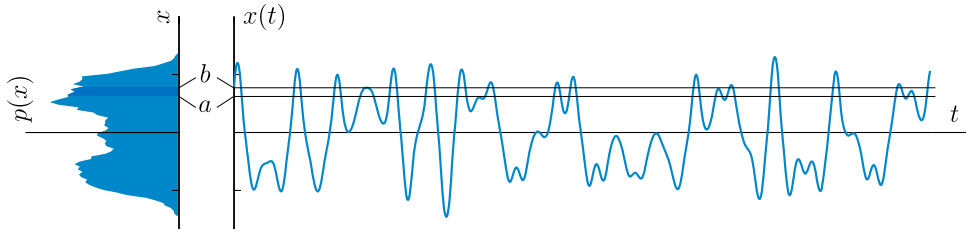

Figure 2.1: A generic sample of $x(t)$ signal and corresponding probability density function $p(x)$

Remark: In these notes *probability distribution* will be called

$$P(x) = \int_{-\infty}^{x} p(s)ds \tag{2.2}$$

so $p(x) = dP(x)/dx$. However, often $p(x)$ is called a "distribution," too, in which case $P(x)$ will be called the *cumulative distribution*.

Thinking of $x(t)$ in terms of probability density or distribution is a *point of view* that considers it as a *random process*—but, it does not imply *anything* about the actual process generating it, which may indeed be *deterministic*.

2.1 Description of Probability

Given $p(x)$, we define the *expected value* of any function of x, $f(x)$, as

$$E\left[f(x)\right] \triangleq \int_{-\infty}^{\infty} f(x)\, p(x)\, dx\,. \tag{2.3}$$

2.1.1 Mean or First Moment of $p(x)$

In the absence of other information, $E[f(x)]$ is a good guess at the value of $f(x)$. A special case is when $f(x) \equiv x$, so for it we define the *mean* value of x or the *first moment* of $p(x)$:

$$\mu = E[x] = \int_{-\infty}^{\infty} x\, p(x)\, dx\,. \tag{2.4}$$

Remark 1: Other possible "good measures" for $f(x) = x$ might be

- the *median* $x = \mu_{1/2}$, defined by $\int_{-\infty}^{\mu_{1/2}} x\, p(x)\, dx = \int_{\mu_{1/2}}^{\infty} x\, p(x)\, dx$, or
- the *mode* $x = \mu_{\max}$, defined by $p(\mu_{\max}) \geq p(x) \quad \forall x$.

Remark 2: For the Gaussian (Normal) density function:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{2.5}$$

$\mu = \mu_{1/2} = \mu_{\max}$. In general this is not true, unless $p(x)$ is symmetric with respect to $x = \mu$.

2.1.2 Variance or Second Central Moment of $p(x)$

Another important expected value is for $f(x) = (x-\mu)^2$, which is called *variance* of x or a *second central momentum* of $p(x)$:

$$\sigma^2 \triangleq E\left[(x-\mu)^2\right] = \int_{-\infty}^{\infty} (x-\mu)^2\, p(x)\, dx\,. \tag{2.6}$$

The square root of variance, or σ, is called *standard deviation*. σ^2 measures the "spread" of the random process about its mean, μ.[1] Please also note that:

$$\begin{aligned}
\sigma^2 &= \int_{-\infty}^{\infty} (x^2 - 2\mu x + \mu^2)\, p(x)\, dx && (2.7)\\
&= E\left[x^2 - 2\mu x + \mu^2\right] && (2.8)\\
&= E\left[x^2\right] - 2\mu E[x] + \mu^2 E[1] && (2.9)\\
&= E\left[x^2\right] - 2\mu^2 + \mu^2 && (2.10)\\
&= E\left[x^2\right] - \left(E[x]\right)^2\,, && (2.11)
\end{aligned}$$

where $E[1]$ and $E\left[x^2\right]$ are the zeroth and the second moments of $p(x)$, respectively.

[1] The variance is also the *covariance* of x with itself (auto-covariance in Chapter 3).

2.1.3 Higher Order Expected Values

Other important and frequently used expected values are *skewness*:

$$\alpha_3 = \gamma_1 = \frac{E\left[(x-\mu)^3\right]}{\left(E\left[(x-\mu)^2\right]\right)^{3/2}}\,, \tag{2.12}$$

which is based on the third central moment and measures *asymmetry* of the density function, and *kurtosis*:

$$\alpha_4 = \beta_2 = \frac{E\left[(x-\mu)^4\right]}{\left(E\left[(x-\mu)^2\right]\right)^{2}}\,, \tag{2.13}$$

which is based on the fourth central moment and measures *peakedness* of the density function. In addition we have *kurtosis excess* $\gamma_2 = \beta_2 - 3$. For a Gaussian distribution/density functions $\gamma_1 = \gamma_2 = 0$.

2.2 Estimators

μ, σ^2, γ_1, β_2 and other moment related quantities are *properties* of $p(x)$ and, indeed, *define* $p(x)$ via *characteristic function*[2]. However, we usually do not know probability distribution $p(x)$. What we usually *do* have are samples of $x(t)$:

$$\{x(t_1), x(t_2), x(t_3), \ldots\} \equiv \{x_1, x_2, x_3, \ldots\}\,. \tag{2.14}$$

Thus the question is, given the finite sequence $\{x_i\}_{i=1}^N$ what are the best estimates of the above described statistics?

2.2.1 Mean Estimator

Let us begin with the best estimate of the mean. Let $\widehat{\mu}$ be the estimate of μ. Then the error in the estimate is:

$$e = \sum_{i=1}^{N} \left(x_i - \widehat{\mu}\right)^2\,, \tag{2.15}$$

and we seek $\widehat{\mu}$ that minimizes e:

$$e = \sum_{i=1}^{N} \left(x_i^2 - 2x_i\,\widehat{\mu} + \widehat{\mu}^2\right) = \sum_{i=1}^{N} x_i^2 - 2\widehat{\mu}\sum_{i=1}^{N} x_i + N\widehat{\mu}^2\,. \tag{2.16}$$

Thus minimization yields:

$$\frac{de}{d\widehat{\mu}} = 0 = -2\sum_{i=1}^{N} x_i + 2N\widehat{\mu}\,, \tag{2.17}$$

which gives the following

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i\,, \tag{2.18}$$

where $\widehat{\mu}$ is the *sample mean* or an *estimator* of μ.

[2] refer to the stochastic systems or random processes textbooks

Remark 1: An estimator $\widehat{a}$ is *consistent* if

$$\lim_{N\to\infty} \widehat{a} = a\,. \tag{2.19}$$

For $\widehat{\mu} = \frac{1}{N}\sum x_i \to \mu$ as $N \to \infty$ by the Law of Large Numbers (or Chebishev Inequality).

Remark 2: An estimator $\widehat{a}$ is *unbiased* if

$$E\left[\widehat{a}\right] = a\,. \tag{2.20}$$

For $\widehat{\mu} = \frac{1}{N}\sum x_i$, $E[\widehat{\mu}] = \frac{1}{N}\sum E[x_i] = \frac{1}{N}(N\mu) = \mu$, thus it is unbiased.

2.2.2 Central Limit Theorem

Gaussian (Normal) densities, Eq. (2.5), play an important role in experimental measurements because of the *central limit theorem* (CLT).

CLT: If $\widehat{\mu}$ is the sample mean of $\{x_i\}_{i=1}^N$ from *any* probability density with mean μ and variance σ^2, then the normalized random variable

$$y = \frac{\sum_{i=1}^N x_i - N\mu}{\sigma\sqrt{N}} = \frac{\widehat{\mu}-\mu}{\sigma/\sqrt{N}} \tag{2.21}$$

has probability density[3]

$$p(y) \to \mathcal{N}(0,1) \tag{2.22}$$

as $N \to \infty$, where $\mathcal{N}$ is the Normal (i.e., Gaussian) density with zero mean and unit variance.

Remark:

$$p(y) \sim \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2\right) = \frac{1}{\sqrt{2\pi}(\sigma/\sqrt{N})}\exp\left[-\frac{1}{2}\left(\frac{\widehat{\mu}-\mu}{\sigma/\sqrt{N}}\right)^2\right] \tag{2.23}$$

The CLT says that the *distribution of estimates* $\widehat{\mu} \to$ Gaussian *no matter* what the original parent distribution is, and that the "mean of the means" $E[\widehat{\mu}] = \mu$, with $\sigma_{\widehat{\mu}} = \sigma/\sqrt{N}$. Thus, $\sigma_{\widehat{\mu}} \to 0$ as $N \to \infty$, too.

2.2.3 Variance, Skewness, and Kurtosis Estimators

The obvious guess for the variance σ^2 estimator, given the logic of $\widehat{\mu}$, would be:

$$\widehat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^N (x_i - \widehat{\mu})^2 = \frac{1}{N}\sum_{i=1}^N \left(x_i^2 - 2x_i\,\widehat{\mu} + \widehat{\mu}^2\right) = \frac{1}{N}\sum_{i=1}^N x_i^2 - \widehat{\mu}^2\,. \tag{2.24}$$

[3] y is a random variable since $\widehat{\mu}$ is a random variable itself, which varies for different sample sets.

However,

$$\begin{aligned} E\left[\widehat{\sigma}^2\right] &= \frac{1}{N}\sum_{i=1}^{N} E\left[x_i^2\right] - E\left[\widehat{\mu}^2\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} E\left[x^2\right] - E\left[\widehat{\mu}^2\right] \\ &= E\left[x^2\right] - E\left[\widehat{\mu}^2\right] \\ &= \underbrace{E\left[x^2\right] - E[x]^2}_{\sigma^2} + \underbrace{E\left[\widehat{\mu}\right]^2 - E\left[\widehat{\mu}^2\right]}_{-\sigma_{\widehat{\mu}}^2} \\ &= \sigma^2 - \sigma_{\widehat{\mu}}^2 = \sigma^2 - \frac{\sigma^2}{N} \quad \text{using CLT}\,. \end{aligned} \tag{2.25}$$

where we have used the facts that $E\left[x_i^2\right] = E\left[x^2\right]$, and $E[x]^2 = E\left[\widehat{\mu}\right]^2$ since $\widehat{\mu}$ is unbiased. Therefore, $E\left[\widehat{\sigma}^2\right] = \frac{N-1}{N}\sigma^2 \neq \sigma^2$ and Eq. (2.25) is a *biased* estimator. However, it is clear that the following is *unbiased sample variance*:

$$\widehat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \widehat{\mu}\right)^2\,. \tag{2.26}$$

The following estimators for *skewness* and *kurtosis* are also unbiased:

$$\widehat{\gamma}_1 = \frac{k_3}{k_2^{3/2}}, \quad \widehat{\beta}_2 = \frac{k_4}{k_2^2}, \tag{2.27}$$

where

$$\begin{aligned} k_2 &= \widehat{\sigma}^2, \\ k_3 &= \frac{N}{(N-1)(N-2)}\sum_{i=1}^{N}\left(x_i - \widehat{\mu}\right)^3, \\ k_4 &= \frac{N[(N+1)\sum_{i=1}^{N}\left(x_i - \widehat{\mu}\right)^4 - 3(N-1)^2 k_2]}{(N-1)(N-2)(N-3)}\,. \end{aligned} \tag{2.28}$$

2.3 Stationarity

Intuitively, stationarity means no change over time, or properties are constant over time. Specifically:

SS: A process $x(t)$ is *strictly stationary* if

$$p(x(t_1), x(t_2), \ldots, x(t_N)) = p(x(t_1+T), x(t_2+T), \ldots, x(t_N+T)),$$

where $\{t_1, t_2, \ldots, t_N\}$ is an arbitrary set of sample times and T is an arbitrary time shift.

Here, p is a *joint* probability distribution:

$$\begin{aligned} &\text{Prob}\left\{a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2, \ldots, a_N \leq x_N \leq b_N\right\} \\ &\qquad = \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_N}^{b_N} p(x_1, x_2, \ldots, x_N)\, dx_N \cdots dx_2\, dx_1\,. \end{aligned} \tag{2.29}$$

Remark: SS is a very strong assumption. It implies, among other things, that *all moments* are constant.

WSS: A process $x(t)$ is *wide sense (weakly) stationary* if:

1. $E[x(t_1)] = E[x(t_2)] = \mu \ \ \forall t_1, t_2$
2. $E[x(t_1)x(t_2)] = E[x(t_1)x(t_1+\tau)] \triangleq R(\tau)$, where $\tau = t_2 - t_1$, it is t_1 independent, and $R(\tau)$ is called the *autocorrelation*.
3. $R(0) = E[x^2] < \infty$, (i.e., $x(t)$ is mean square integrable or has bounded "power")

Remark 1: WSS contains only the second order statistics and is adequate for *linear statistics*, i.e. the investigation of *linear relationships* between properties of the system at two different times.[4]

Remark 2: Heuristically speaking, a stationary signal comes from a *steady state motion*. A bit more precisely, it is governed by an *invariant measure* on an *attractor*.

The difficulty in getting a handle on the concept of stationarity is that it is a *hypothesis* that is difficult to test. It is also highly context dependent and depends on what the *sample functions* of a random process are in the probability space.

Consider the following examples

1. $x(t) = a\sin t + n(t)$, where $n(t)$ is Gaussian white noise. Examine Fig. 2.2, where 100 sample functions are plotted on top of each other. The whole function is not stationary since the mean of $x(t)$ changes with time. However, $a\sin T + n(t)$ is stationary for any fixed T.

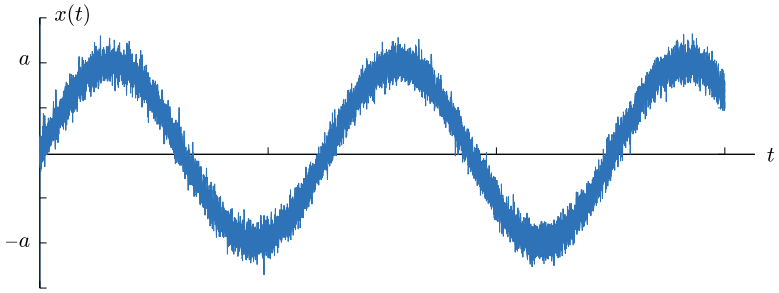

Figure 2.2: $a\sin t + n(t)$ for 100 different sample functions plotted together

2. $x(t) = a\sin(t + t_0)$ is stationary for t_0 random (arbitrary) initial time or phase (Fig. 2.3).
3. Now lets go back to $x(t) = a\sin t + n(t)$ case. Here, averages over time scale $\Delta T_1 < 1$ period will not be stationary. However, averages over $\Delta T_2 \gg 1$ period will be (approximately) stationary (Fig. 2.4). It is hard to figure out if this is the same idea of stationarity.

[4] One still might use "linear statistics" even if the process generating $x(t)$ is *nonlinear* (e.g., random number generator).

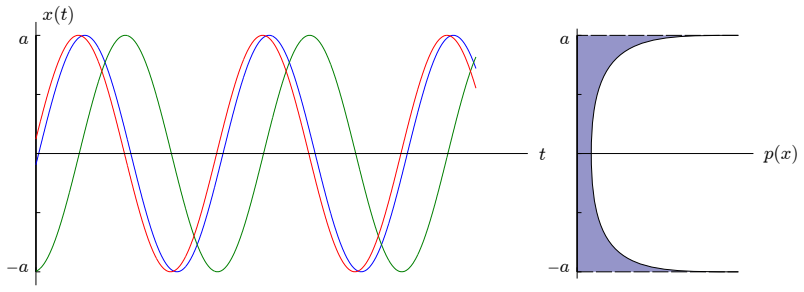

Figure 2.3: $a\sin(t+t_0)$ for 3 different t_0 random initial time

4. Usually transient processes are not stationary, but if we let the transients to die out we get approximate stationarity of steady state. For example, a motion of particle in a double-well potential (refer to Fig. 2.5) with dissipation can be described by:

$$\ddot{x} + \gamma\dot{x} + x(x^2 - 1) = 0. \tag{2.30}$$

Now let $x(t) = X(t, x_0)$, where the initial condition x_0 is randomly selected from the basin of attraction $\mathcal{B}_1$ for $x = -1$ equilibrium point. Two different sample functions (solutions to two different initial conditions: $(2.2, 0)$ red line and $(0, -1)$ blue line) are shown in Fig. 2.5. Here, it is clear that the process is *not* stationary. However, we can wait until "transients die off" and then we get approximate stationarity.

Simple Stationarity Test

A simple test for the stationarity of a scalar time series $\{x_n\}_{n=1}^N$ is the examination of *moving averages* with different window lengths $\Delta t = t_w$ containing $M \ll N$ samples, refer to Fig. 2.6. For the data points in each sliding time window estimate sample mean μ_i and its corresponding standard deviation $\sigma_{\widehat{\mu},i} = \widehat{\sigma}_i/\sqrt{N}$. If the sample means $\widehat{\mu}_i$ stay mainly inside the $\pm$ one standard deviation $\sigma_{\widehat{\mu},i}$ of each other, we can say that our time series is approximately stationary. If, on the other hand,

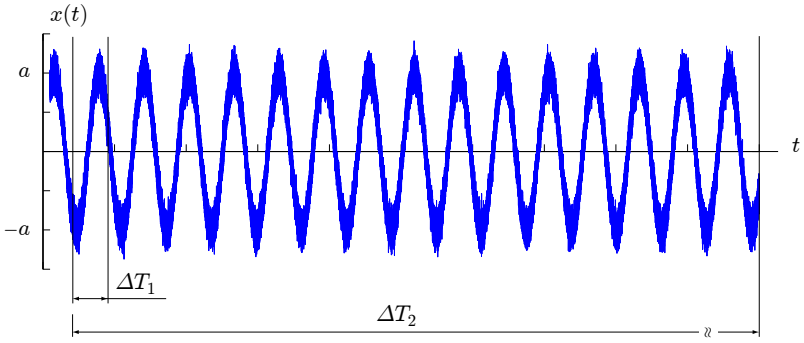

Figure 2.4: Different time scale for $x(t) = a\sin t + n(t)$ signal

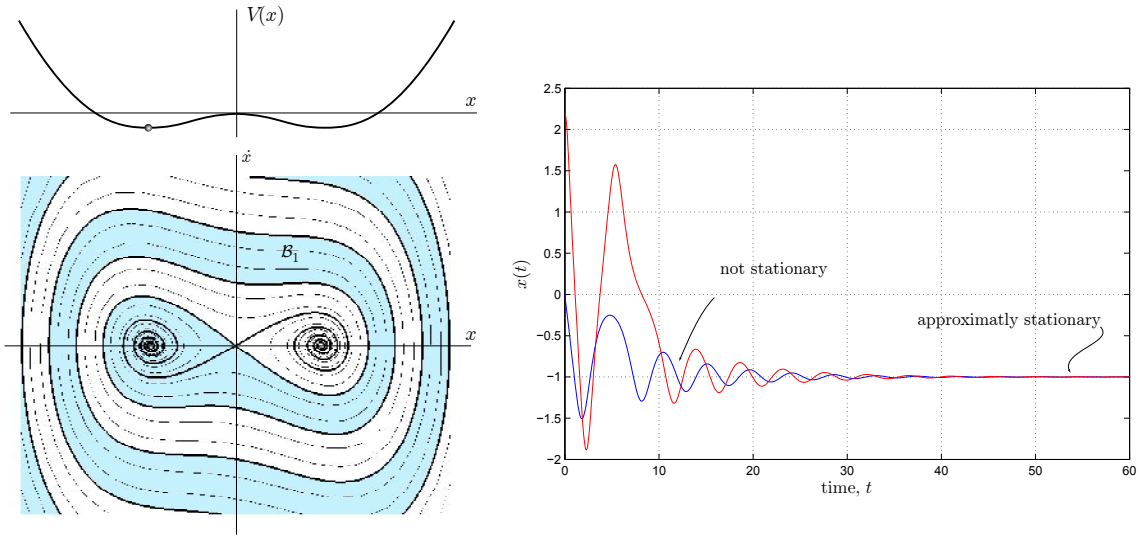

Figure 2.5: Top left plot represents a two-well potential $V(x) = -\frac{x^2}{2} + \frac{x^4}{4}$, the bottom left plot shows the basins of attraction for the equilibrium points $x = \pm 1$, and the right plot shows two sample function from the $\mathcal{B}_1$ basin of attraction for $x = -1$ equilibrium.

sample means show significant variation in time than our time series, it is not stationary. For the illustration of this concept please refer to the right plots in Fig. 2.6. Please note that our results will depend on the choice of time window t_w as was illustrated in the third example of the previous section.

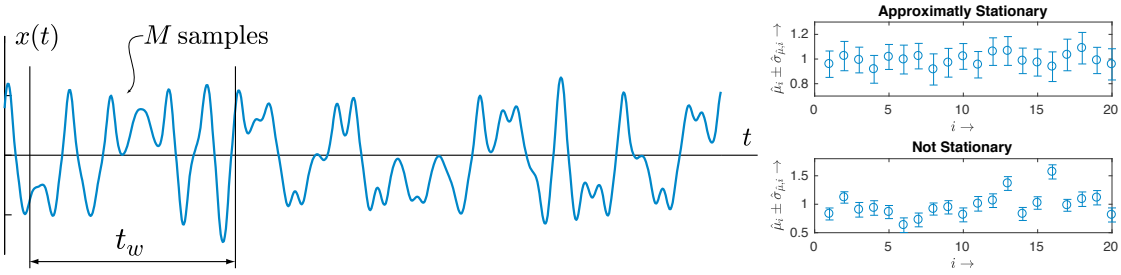

Figure 2.6: Moving averages with different window length t_w (left plot), and simple stationarity test (right plots)

2.4 Ergodicity

General "signal" $x(t)$ is a *sample function* of a *random process*. Thus every trial will generate a new function. In general, we have been considering two types of averaging: *ensemble averaging* and *time averaging*.

Ensemble average is the average of a quantity over all realizations (or trials) of a random variable. For example, if $x(t)$ is a Gaussian random variable, $p(x(t))$ for each fixed t is Gaussian. If it is a stationary random process, then $p(x(t_1)) = p(x(t_2))$. Now let us look at the mean value of our random process for some fixed time t_j:

$$\widehat{\mu}_j \triangleq \frac{1}{N_T} \sum_{i=1}^{N_T} x_i(t_j)\,, \tag{2.31}$$

where t_j is fixed, and we have total of N_T trials of our random process. In other words the $\widehat{\mu}_j$ is an estimate for

$$\mu_j = E\left[x(t_j)\right] = \int_{-\infty}^{\infty} x(t_j)\, p(x(t_j))\, dx(t_j)\,. \tag{2.32}$$

Thus, the *ensemble average* is the expected value (or an estimate of) a random process for a particular fixed time. Now a good question is: does $\widehat{\mu}_j = \widehat{\mu}_k$ for $j \neq k$? The answer is yes for a stationary process.

Time average is the average using a *single* sample function over some time interval. For example, given any function f,

$$\bar{f} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} f(x(t))\, dt \tag{2.33}$$

In particular, for the time mean value estimates:

$$\bar{\mu} = \frac{1}{T} \int_0^T x(t)dt, \quad \text{or} \quad \bar{\mu} = \frac{1}{N_s} \sum_{i=1}^{N_s} x(t_i)\,, \tag{2.34}$$

where N_s is the number of time samples of $x(t)$.

Definition: We say that a random process is *ergodic* if ensemble averages are equivalent to time averages. More precisely, given f the stationary process is ergodic if $\widehat{\mu}_f = \bar{\mu}_f$, or

$$\int_{-\infty}^{\infty} f(x(t))\, p(x(t))\, dx(t) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} f(x(t))\, dt\,. \tag{2.35}$$

Please note that in the above equation on the left hand side the t is *fixed* and we have an ensemble average, and on the right hand side we have a time average of a single sample function (see Fig. 2.7).

Random processes can be ergodic for some f and not for others. For example, consider $x(t) = A\sin(t + B)$ random process, where A and B are Gaussian random variables. Then, this process is *ergodic in the mean* ($f(x) = x$), but not *ergodic in the power* ($f(x) = x^2$).

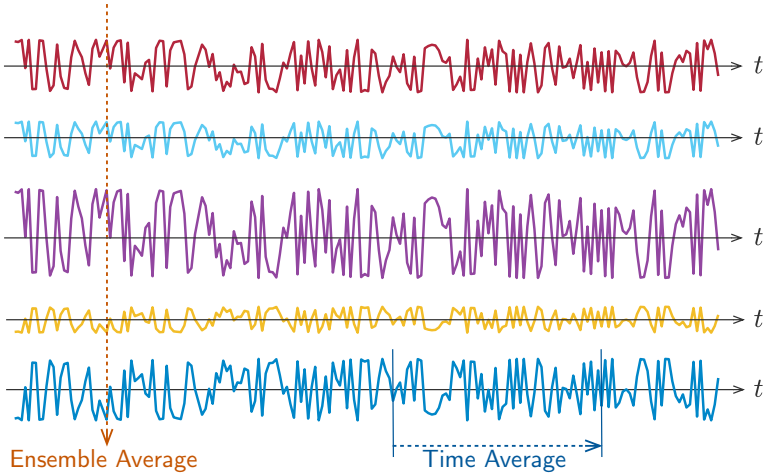

Figure 2.7: Ensemble and time averages

Problems

Problem 2.1

In Matlab, generate M trials of random processes with the forms

1. $x(t) = \cos(t) + n(t)$;
2. $x(t) = \cos(t + \theta) + n(t)$; and
3. $x(t) = A\cos(t + \theta) + n(t)$,

where $0 \leq t \leq T$ for some sample time T, $n(t)$ is a normally-distributed (Gaussian) random process with zero mean and variance σ, θ is a uniformly-distributed constant random variable on $[0; 2\pi]$, and A is a normally-distributed constant with zero mean and unit variance. In each case, generate uniformly-sampled time series $\{x_1; x_2; \ldots; x_N\}$ where $x_i = x(t_i)$.

Use the above-generated time series to examine the *stationarity* and *ergodicity* of the different processes. Experiment with different values of M, N, T, and σ to make sure you get good results. Discuss your findings.

Note: Think carefully about your sampling time interval and the length of the time series (relationship between N and T).

Problem 2.2

Generate one long time series of the random process (1) in Exercise 1. Estimate the mean of the process many times from different subsets of the time series, and show that the distribution of $\widehat{\mu}_x$ is approximately Gaussian with mean and variance, as predicted by the Central Limit Theorem.

Note: the way in which you make the samples used for each estimate can have a large impact on your results!

Chapter 3

Correlation and Auto-Correlation

Given two random variables $x(t)$ and $y(t)$

$$E\left[x(t_1)y(t_2)\right] = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x(t_1)y(t_2)p(x(t_1),y(t_2))\,dx(t_1)dy(t_2) \tag{3.1}$$

is called *cross correlation* $R_{xy}(t_1,t_2) = R\left(x(t_1),y(t_2)\right)$ between $x(t_1)$ and $y(t_2)$, $\forall\ t_1,t_2$ fixed.[1] If the generating processes are *stationary* and $t_1 \equiv t$, $t_2 \equiv t+\tau$

$$\begin{aligned} E\left[x(t)y(t+\tau)\right] &= R_{xy}(\tau) \\ &= E\left[x(t-\tau)y(t)\right] \\ &= E\left[y(t)x(t-\tau)\right] \\ &= R_{yx}(-\tau)\,. \end{aligned} \tag{3.2}$$

But, in general, for a *nonstationary* processes $R_{xy}(\tau) \neq R_{yx}(-\tau)$! If, in addition, both processes are statistically independent, then $R_{xy}(\tau) = R_{yx}(\tau) = R_{xy}(-\tau)$.

Given two sample sets $\{x_i(t)\}_{i=1}^N$ and $\{y_i(t+\tau)\}_{i=1}^N$, we estimate cross correlation $R_{xy}(\tau)$ as

$$\widehat{R}_{xy}(\tau,t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t)y_i(t+\tau)\,. \tag{3.3}$$

If the generating processes are stationary then the correlation is independent of t and we can write

$$\widehat{R}_{xy}(\tau) = \frac{1}{N}\sum_{i=1}^{N} x_i y_i\,, \tag{3.4}$$

where x_i and y_i are assumed to be sampled some fixed τ time apart. If we subtract off the means from these variables, then

$$\begin{aligned} C_{xy}(\tau,t) &= E\left[(x(t)-\mu_x)(y(t+\tau)-\mu_y)\right] \\ &= E\left[(x(t)y(t+\tau) - \mu_x y(t+\tau) - \mu_y x(t) + \mu_x\mu_y)\right] \\ &= R_{xy}(\tau,t) - \mu_x\mu_y \end{aligned} \tag{3.5}$$

[1] Cross correlation is a function of two variables and represents a surface.

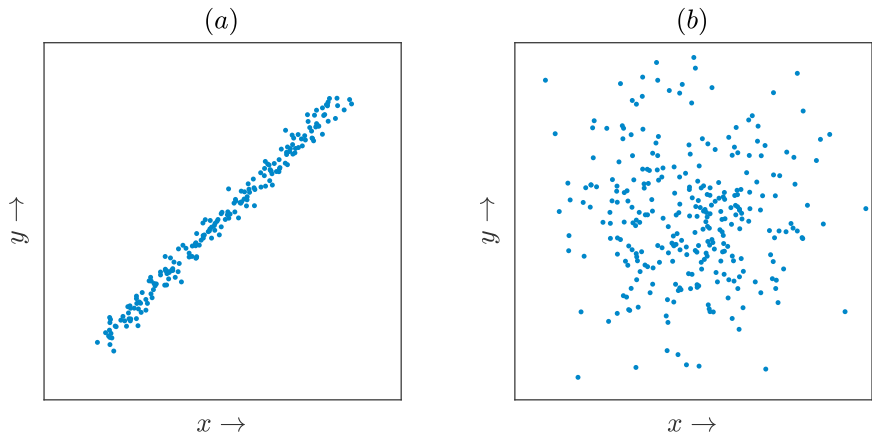

Figure 3.1: Highly correlated data (a) and uncorrelated data (b)

is called the *cross-covariance*, which—for stationary processes—can be estimated from the above sample sets as:

$$\begin{aligned}\widehat{C}_{xy}(\tau) &= \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\widehat{\mu}_x)\,(y_i-\widehat{\mu}_y)\\ &= \widehat{R}_{xy}(\tau)-\widehat{\mu}_x\widehat{\mu}_y,,\end{aligned} \tag{3.6}$$

where, again, x_i and y_i are assumed to be sampled some fixed τ time apart. If we normalize cross-covariance C_{xy} by σ_x and σ_y we get the *correlation coefficient*:[2]

$$\rho_{xy}(\tau) = \frac{C_{xy}(\tau)}{\sigma_x\sigma_y}\,, \tag{3.7}$$

which means that $-1 \le \rho_{xy} \le 1$. We say that x and y are (linearly) uncorrelated if $\rho_{xy} = C_{xy} = 0$ (see Fig. 3.1(b)). Perfectly correlated (i.e., identical) processes have $\rho_{xy} = \pm 1$ (see Fig. 3.1(a)). For mean zero processes $R_{xy} = C_{xy}$.

Special case of the cross-correlation is when $x = y$ and R_{xx} is called an *autocorrelation*. For a stationary process

$$\begin{aligned}R_{xx}(\tau) &= E[x(t)x(t+\tau)]\\ &= E[x(t-\tau)x(t)]\\ &= E[x(t)x(t-\tau)]\\ &= R_{xx}(-\tau)\,.\end{aligned} \tag{3.8}$$

Thus, the autocorrelation is always symmetric. For a purely random signal, $R_{xx}(\tau) \cong 0$ if $\tau \neq 0$. However, in general for random signals we see autocorrelation similar to the one shown in Fig. 3.2. The first zero crossing of R_{xx} is the *correlation time*, τ_c. The idea is that for $\tau > \tau_c$, $x(t)$ and $x(t+\tau)$ act like independent variables.

[2]The terminology use here and in the literature is a little confusing: the cross-correlation of the *centered random variables* is called the cross-covariance, and the *normalized cross-covariance* is called the *correlation coefficient*.

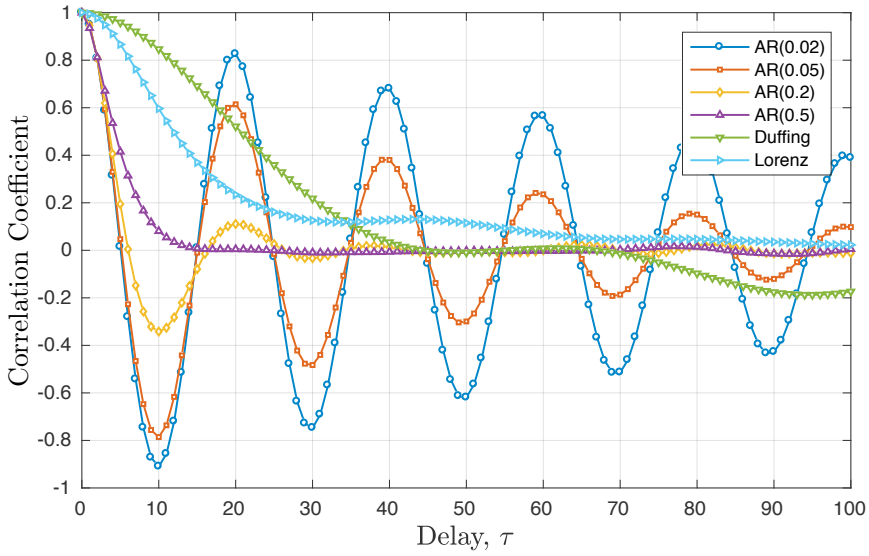

Figure 3.2: Correlation coefficient $\rho_{xx}(\tau)$ versus delay τ for a colored random processes described by $x_{n+1} = (2-(\pi/10)^2-\rho)x_n + (\rho-1)x_{n-1} + \eta_n$, where η_n is white noise and $\rho = [0.02, 0.05, 0.2, 0.5]$. Each case is labeled as AR(ρ). In addition, autocorrelations of x time series from Duffing ($\ddot{x} + 0.2\,\dot{x} - x + x^3 = 0.33\cos t$) and y time series from Lorenz ($\dot{x} = -\frac{8}{3}x + yz,\ \dot{y} = -10\,(y-z)\,,$ and $\dot{z} = -xy + 28y - z$) oscillators are plotted.

In many cases we deal with a collection of "signals" $x_1(t), x_2(t), \ldots x_n(t)$, for which they can be collected in a vector-valued signal $\mathbf{x}(t) \in \mathbb{R}^n$. Now if we are given a vector-valued time series $\{\mathbf{x}^{(k)}\}_{k=1}^N$, where $\mathbf{x}^{(k)} = \mathbf{x}(t_k)$, it can be arranged into a coordinate matrix $\mathbf{X} \in \mathbb{R}^{N\times n}$ as:

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_n^{(N)} \end{bmatrix} \tag{3.9}$$

Now we can define a *cross-correlation matrix* for $\mathbf{X}$ as

$$\widehat{\mathbf{R}} = \frac{1}{N}\mathbf{X}^T\mathbf{X}, \quad \text{or} \quad \widehat{R}_{ij} = \frac{1}{N}\sum_{k=1}^{N} x_i^{(k)} x_j^{(k)}\,. \tag{3.10}$$

Similarly, the corresponding *cross-covariance* matrix is

$$\widehat{\mathbf{C}} = \frac{1}{N-1}\bar{\mathbf{X}}^T\bar{\mathbf{X}}, \quad \text{or} \quad \widehat{C}_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}(x_i^{(k)} - \widehat{\mu}_i)(x_j^{(k)} - \widehat{\mu}_j) = \begin{cases} \widehat{\sigma}_i^2, & i = j \\ \widehat{c}_{ij}, & i \neq j \end{cases}, \tag{3.11}$$

where $\bar{\mathbf{X}}$ has zero mean columns.

Now, if we have a two-dimensional random variable with scalar components x and y, then our

cross-covariance matrix will be two dimensional:

$$\mathbf{C} \cong \begin{bmatrix} \sigma_x^2 & c_{xy} \\ c_{yx} & \sigma_y^2 \end{bmatrix}, \tag{3.12}$$

and if x and y are (linearly) uncorrelated, we have

$$\mathbf{C} \cong \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}, \tag{3.13}$$

Problems

Problem 3.1

[Linear Prediction] Suppose we wish to predict the values of a variable Y by observing (recording) the values of another variable X. In particular, the available data (see Fig. 3.3) suggests that a good prediction model for Y is the linear function

$$Y_p \triangleq \alpha X + \beta.$$

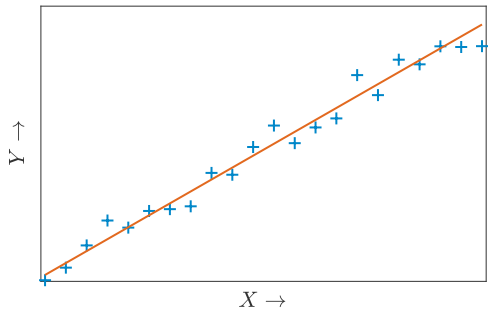

Figure 3.3: Pairwise observations on (X, Y) constitute a scatter diagram and straight line approximates the relationship between the X and Y.

Now although Y can be related to X, the values it takes on may be influenced by other sources that do not affect X. Thus, in general, $|\rho_{XY}| \neq 1$ and we expect that there will be an error between the predicted value of Y, that is, Y_p and the value that Y actually assumes. Our task is to adjust the coefficients α and β in order to minimize the mean-square error

$$\epsilon \triangleq E[(Y - Y_p)^2].$$

In other words, to come up with *optimum linear prediction* or *linear regression*. Do the following:

- Find expressions that need to be satisfied to minimize ϵ with respect to α and β.
- Solve this expressions for estimated $\hat{\alpha}$ and $\hat{\beta}$.

- Write the expression for the best predictor and the corresponding $\epsilon_{\min}$.
- What happens if $\rho_{XY} = 0$? In this case, what is the best predictor for Y and how does it relate to X?
- How can we use the correlation coefficient to establish the relationship between two variables?

Problem 3.2

Generate the data for the systems shown in Fig. 3.2. That is, generate 40,000 point records for each of these:

1. Colored random processes described by $x_{n+1} = (2 - \omega_n^2 - \rho)x_n + (\rho - 1)x_{n-1} + \eta_n$, where $\omega_n = 2\pi/20$ (rad/s), η_n is Gaussian noise with unit variance (use `randn` function), and for $\rho = [0.02, 0.05, 0.2, 0.5]$ values. For the initialization use $x_1 = 0$ and $x_2 = 0$.
2. x time series from the Duffing's oscillator

$$\ddot{x} + 0.2\,\dot{x} - x + x^3 = 0.33\cos t\,,$$

   starting from $x(0) = -0.6973$ and $\dot{x}(0) = 0.0729$ with 0.1 sampling time.
3. y time series from the Lorenz oscillator

$$\dot{x} = -\frac{8}{3}x + yz, \quad \dot{y} = -10\,(y - z)\,, \quad \text{and} \quad \dot{z} = -xy + 28y - z\,,$$

   staring from $x(0) = 20$, $y(0) = 5$, and $z(0) = -5$ initial condition and with 0.02 sampling time.

Now calculate and plot:

1. Autocorrelations for all the time series just as shown in Fig. 3.2
2. Cross-covariances between the Duffing and Lorenz time series, and between AR(0.2) and AR(0.5) time series.
3. Cross-covariances between the Duffing and AR(0.5), and Lorentz and AR(0.5) time series.

What can you infer from all these plots?

Chapter 4

Fourier Analysis and Power Spectral Density

4.1 Fourier Series and Transforms

Recall Fourier series for periodic functions

$$x(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left[a_n \cos\frac{2\pi nt}{T} + b_n \sin\frac{2\pi nt}{T}\right] \tag{4.1}$$

for $x(t+T) = x(t)$, where

$$\begin{aligned} a_0 &= \frac{2}{T}\int_0^T x(t)\,dt \quad \left(\frac{a_0}{2} = \bar{x}\right) \\ a_n &= \frac{2}{T}\int_0^T x(t)\cos n\omega t\,dt \quad \left(\omega = \frac{2\pi}{T}\right) \\ b_n &= \frac{2}{T}\int_0^T x(t)\sin n\omega t\,dt\,. \end{aligned} \tag{4.2}$$

Dirichlet Theorem: For $x(t)$ periodic on $0 \le t < T$, if $x(t)$ is bounded, has a finite number of maxima, minima, and discontinuities, then the Fourier Series Eq. (4.1) converges $\forall t$ to $\frac{1}{2}[x(t^+) + x(t^-)]$.

Complex form of Eq. (4.1) is better for experimental applications. Using Euler's (or de Moivre's) formulas we get:

$$\begin{aligned} \cos\omega t &= \frac{1}{2}\left(e^{i\omega t} + e^{-i\omega t}\right) \\ \sin\omega t &= \frac{1}{2i}\left(e^{i\omega t} - e^{-i\omega t}\right)\,. \end{aligned} \tag{4.3}$$

Using above Eq. (4.1) can be rewritten as:

$$x(t) = \sum_{-\infty}^{\infty} X_n\, e^{in\omega t}\,, \tag{4.4}$$

where

$$\begin{aligned} X_0 &= \frac{a_0}{2} \\ X_{\pm n} &= \frac{1}{2}\left(a_n \mp i b_n\right) . \end{aligned} \tag{4.5}$$

Please also note that $X_n = X^*_{-n}$. Therefore:

$$X_n = \frac{1}{T}\int_0^T x(t)\, e^{-in\omega t}\, \mathrm{d}t = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\, e^{-in\omega t}\, \mathrm{d}t\,. \tag{4.6}$$

If "signal" $x(t)$ is not periodic, we let $f_n = \frac{n}{T}$ (i.e., in Eq. (4.6) $n\omega = 2\pi f_n$). Now, we define a function $X(f)$ by $X(f_n) = TX_n$ (i.e., $X_n = X(f_n)/T$) to get the following:

$$x(t) = \sum_{-\infty}^{\infty} X_n\, e^{in\omega t} = \sum_{n=-\infty}^{\infty} \frac{1}{T} X(f_n)\, e^{i2\pi f_n t} = \sum_{n=-\infty}^{\infty} X(f_n)\, e^{i2\pi f_n t} \Delta f_n\,, \tag{4.7}$$

where we used the fact that $\Delta f_n = f_{n+1} - f_n = \frac{n+1}{T} - \frac{n}{T} = \frac{1}{T}$. Therefore, in the limit as $T \to \infty$ and $\Delta f_n \to 0$ in Eq. (4.7), we get our signal in the *time domain* as

$$x(t) = \int_{-\infty}^{\infty} X(f)\, e^{i2\pi f t}\, \mathrm{d}f\,. \tag{4.8}$$

Now, assuming that everything converges and using Eq. (4.6) we get the corresponding *frequency domain* expression

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-i2\pi f t}\, \mathrm{d}t\,. \tag{4.9}$$

Therefore, $x(t) \leftrightarrow X(f)$ are Fourier Transform pair, where $x(t)$ is in *time domain* and $X(f)$ is in *frequency domain*.

4.1.1 Several Important Properties of Fourier Transforms

We denote a Fourier transform (FT) as $X(f) = \mathcal{F}(x(t))$ and $x(t) = \mathcal{F}^{-1}(X(f))$. Now, we can write several of the properties of FT:

1. **Linearity:** $\mathcal{F}[\alpha x(t) + \beta y(t)] = \alpha X(f) + \beta Y(f)$
2. **Duality:** $x(t) \leftrightarrow X(f) \quad \Rightarrow \quad X(t) \leftrightarrow x(-f)$
3. **Conjugation:** $x(t) \leftrightarrow X(f) \quad \Rightarrow \quad x^*(t) \leftrightarrow X^*(-f)$.

   Therefore, for real signal $x(t)$, $X(f) = X^*(-f)$. This instead gives:

   $$|X(f)|^2 = X(f)X^*(f) = X * (-f)X(-f) = |X(-f)|^2\,, \tag{4.10}$$

   i.e., for real $x(t)$, $|X(f)|$ is symmetric.
4. **Convolution:**

   $$\mathcal{F}\left[\int_{-\infty}^{\infty} x(\tau)y(t-\tau)\, d\tau\right] = X(f)Y(f) \triangleq \mathcal{F}[x * y]\,,$$

where $x * y$ indicates time convolution between $x(t)$ and $y(t)$. In addition,

$$\mathcal{F}[xy] = \int_{-\infty}^{\infty} X(\phi)Y(f-\phi)\,d\phi \triangleq X * Y\,,$$

where $X * Y$ indicates frequency convolution between $X(f)$ and $Y(f)$ (also, $X * Y = Y * X$).

5. **Differentiation:**

$$\mathcal{F}\left[\frac{\mathrm{d}^k x}{\mathrm{d}t^k}\right] = (i2\pi f)^k X(f)\,,$$

6. **Time Scaling and Shifting:**

$$x(at+b) \leftrightarrow \frac{e^{2\pi i f \frac{b}{a}}}{|a|} X\left(\frac{f}{a}\right)\,.$$

Theorem: Provided $x(t) \in \mathcal{L}^1$ (i.e., $\int_{-\infty}^{\infty} |x(t)|\mathrm{d}t < \infty$ and $x(t)$ has a finite number of maxima, minima, and discontinuities) $X(f)$ exists, and

$$x(t) = \begin{cases} \mathcal{F}^{-1}\left[X(f)\right] & \text{for } x \text{ continuous at } t\,, \\ \frac{1}{2}\left[x(t^+) + x(t^-)\right] & \text{for } x \text{ discontinuous at } t\,. \end{cases}$$

.

There is a problem with the above theorem if we consider the following:

$$\int_{-\infty}^{\infty} |\sin t|\,\mathrm{d}t = \infty\,,$$

which can be fixed using theory of *generalized functions (or distributions)*, *duality* and other basic properties.

4.1.2 Basic Fourier Transform Pairs

1. **Delta (δ) "function":** This actually is a generalized function or distribution defined as:

$$\int_{-\infty}^{\infty} \delta(t)\mathrm{d}t = 1\,. \tag{4.11}$$

Now, by definition of $\delta(t)$, $\Delta(f) = \int_{-\infty}^{\infty} e^{-2\pi i f t}\delta(t-t_0)\mathrm{d}t = e^{-2\pi i f t_0}$, also called "sifting property." Note that Δ is a complex constant with $|\Delta| = 1$. Therefore:

$$\delta(t-t_0) \leftrightarrow e^{-2\pi i f t_0}\,, \tag{4.12}$$

and in particular, $\delta(t) \leftrightarrow 1$. Therefore, by duality property

$$e^{-2\pi i f_0 t} \leftrightarrow \delta(f-f_0)\,, \tag{4.13}$$

and in particular, $1 \leftrightarrow \delta(f)$.

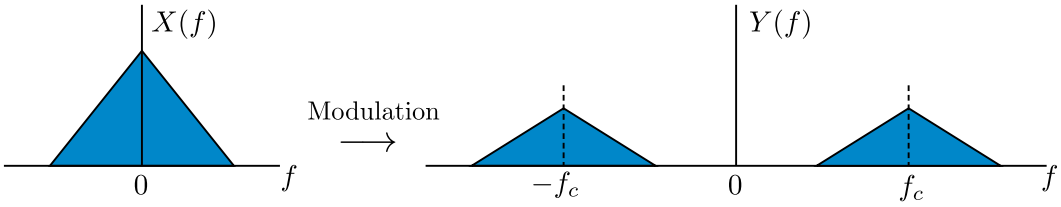

Figure 4.1: Signal modulation in the frequency domain

2. **Trigonometric functions:**

$$\cos(2\pi f_0 t) = \frac{1}{2}\left(e^{2\pi i f_0 t} + e^{-2\pi i f_0 t}\right) , \tag{4.14}$$

so

$$\mathcal{F}\left[\cos(2\pi f_0 t)\right] = \frac{\delta(f-f_0)+\delta(f+f_0)}{2} . \tag{4.15}$$

Similarly,

$$\mathcal{F}\left[\sin(2\pi f_0 t)\right] = \frac{\delta(f-f_0)-\delta(f+f_0)}{2i} . \tag{4.16}$$

3. **Modulated trigonometric functions:** As an example consider $x(t)\cos(2\pi f_c t) \leftrightarrow X(f) * C(f)$, where f_c is called *carrier frequency* and $C(f)$ is given by Eq. (4.15). Then,

$$x(t)\cos(2\pi f_c t) \leftrightarrow \int_{-\infty}^{\infty} X(s)\frac{\delta(f-f_c-s)+\delta(f+f_c-s)}{2}\mathrm{d}s\,, \tag{4.17}$$

where on the right hand side we have a convolution integral, which gives:

$$x(t)\cos(2\pi f_c t) \leftrightarrow \frac{X(f-f_c)+X(f+f_c)}{2} . \tag{4.18}$$

Therefore, if we already know $X(f)$, modulation scales and shifts it to $\pm f_c$ as shown in Fig. 4.1.

4.2 Power Spectral Density

The autocorrelation of a real, stationary signal $x(t)$ is defined by Eq. (3.8) as $R_x(\tau) = E[x(t)x(t+\tau)]$, where we replaced double x in the subscript with just single x (i.e., $R_{xx}(\tau) = R_x(\tau)$). *Power Spectral Density* (PSD), $S_x(f)$, can be defined as a Fourier transform of $R_x(\tau)$:

$$S_x(f) = \int_{-\infty}^{\infty} R_x(\tau)\, e^{-i2\pi f t}\,\mathrm{d}\tau\,. \tag{4.19}$$

The question is: what is the PSD? What does it mean? What is a "spectral density," and why is S_x called a *power* spectral density?

To answer this question, recall that

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-i2\pi f t}\,\mathrm{d}t\,. \tag{4.20}$$

To avoid convergence problems, we consider only a version of the signal observed over a finite-time T,[1] $x_T = x(t)w_T(t)$, where

$$w_T = \begin{cases} 1 & \text{for } 0 \le |t| \le T/2\,, \\ 0 & \text{for } \le |t| > T/2\,. \end{cases} \tag{4.21}$$

Then x_T has the Fourier transform

$$X_T(f) = \int_{-\infty}^{\infty} x_T(t)\, e^{-i2\pi ft}\, \mathrm{d}t\,, \tag{4.22}$$

$$= \int_{-T/2}^{T/2} x(t)\, e^{-i2\pi ft}\, \mathrm{d}t\,, \tag{4.23}$$

and so

$$X_T X_T^* = \left[\int_{-T/2}^{T/2} x(t)\, e^{-i2\pi ft}\, \mathrm{d}t\right]\left[\int_{-T/2}^{T/2} x^*(s)\, e^{i2\pi fs}\, \mathrm{d}s\right]\,, \tag{4.24}$$

$$= \int_{-T/2}^{T/2}\int_{-T/2}^{T/2} x(t)x(s)\, e^{-i2\pi f(t-s)}\, \mathrm{d}t\mathrm{d}s\,, \tag{4.25}$$

where the star denotes complex conjugation and for compactness the frequency argument of X_T has been suppressed. Taking the expectation of both sides of Eq. (4.26)[2], we get

$$E\left[X_T X_T^*\right] = \int_{-T/2}^{T/2}\int_{-T/2}^{T/2} E\left[x(t)x(s)\right]\, e^{-i2\pi f(t-s)}\, \mathrm{d}t\mathrm{d}s\,. \tag{4.26}$$

Letting $s = t + \tau$, one sees that $E[x(t)x(s)] \triangleq E[x(t)x(t+\tau)] = R_x(\tau)$, and thus

$$E\left[X_T X_T^*\right] = \int_{-T/2}^{T/2}\int_{-T/2}^{T/2} R_x(\tau)\, e^{-i2\pi f(t-s)}\, \mathrm{d}t\mathrm{d}s\,. \tag{4.27}$$

To actually evaluate the above integral, *both* variables of integration must be changed. Let

$$\tau = f(t,s) = s - t \quad \text{(as already defined for Eq. (4.30))} \tag{4.28}$$

$$\eta = g(t,s) = s + t\,. \tag{4.29}$$

Then, the integral of Eq. (4.30) is transformed (except for the limits of integration) using the *change of variables formula*:[3]

$$\int_{-T/2}^{T/2}\int_{-T/2}^{T/2} R_x(\tau)\, e^{-i2\pi f(t-s)}\, \mathrm{d}t\mathrm{d}s = \int_{?}^{?}\int_{?}^{?} R_x(\tau)\, e^{-i2\pi f\tau}\, |J|^{-1}\, \mathrm{d}\tau\mathrm{d}\eta\,, \tag{4.30}$$

where $|J|$ is the absolute value of the Jacobian for the change of variables Eq. (4.28) given by

$$J = \begin{vmatrix} \frac{\mathrm{d}f}{\mathrm{d}t} & \frac{\mathrm{d}f}{\mathrm{d}s} \\ \frac{\mathrm{d}g}{\mathrm{d}t} & \frac{\mathrm{d}g}{\mathrm{d}s} \end{vmatrix} = \begin{vmatrix} -1 & 1 \\ 1 & 1 \end{vmatrix} = -2\,. \tag{4.31}$$

[1] This restriction is necessary because not all of our signals will be square integrable. However, they will be mean square integrable, which is what we will take advantage of here.

[2] To understand what this means, remember that Eq. (5) holds for any x(t). So imagine computing Eq. (6) for different x(t) obtained from different experiments on the same system (each one of these is called a sample function). The expectation is over all possible sample functions. Since the exponential kernel inside the integral of Eq. (6) is the same for each sample function, it can be pulled outside of the expectation.

[3] This is a basic result from multivariable calculus. See, for example, I.S. Sokolnikoff and R.M. Redheffer, *Mathematics of Physics and Modern Engineering*, 2nd edition, McGraw- Hill, New York, 1966.

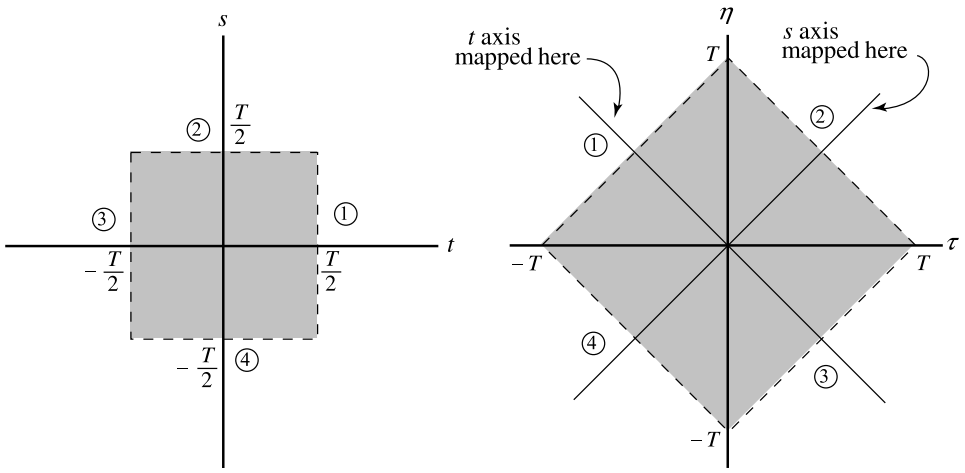

Figure 4.2: The domain of integration (gray regions) for the Fourier transform of the autocorrelation Eq. (7): (left) for the original variables, t and s; (right) for the transformed variables, η and τ, obtained by the change of variables Eq. (4.28). Notice that the square region on the left is not only rotated (and flipped about the t axis), but its area is increased by a factor of $|J| = 2$. The circled numbers show where the sides of the square on the left are mapped by the change of variables. The lines into which the t and s axes are mapped are also shown.

To determine the limits of integration needed for the right hand side of Eq. (4.31), we need to refer to Fig. 4.2, in which the domain of integration is plotted in both the original (t, s) variables and the transformed (τ, η) variables. Since we wish to integrate on η first, we hold τ fixed. For $\tau > 0$, a vertical cut through the diamond-shaped region in Fig. 4.2 (right) shows that $-T + \tau \le \eta \le T - \tau$, whereas for $\tau < 0$ one finds that $-T - \tau \le \eta \le T + \tau$. Putting this all together yields:

$$E\left[X_T X_T^*\right] = \int_{-T}^{T} \int_{-(T-|\tau|)}^{T-|\tau|} R_x(\tau)\, e^{-i2\pi f\tau}\, \mathrm{d}\eta \mathrm{d}\tau = T \int_{-T}^{T} \left[1 - \frac{|\tau|}{T}\right] R_x(\tau)\, e^{-i2\pi f\tau}\, \mathrm{d}\tau\,. \tag{4.32}$$

Finally, dividing both sides of Eq. (4.32) by T and taking the limit as $T \to \infty$ gives

$$\begin{aligned} \lim_{T\to\infty} \frac{1}{T} E\left[X_T X_T^*\right] &= \lim_{T\to\infty} \int_{-T}^{T} \left[1 - \frac{|\tau|}{T}\right] R_x(\tau)\, e^{-i2\pi f\tau}\, \mathrm{d}\tau \\ &= \lim_{T\to\infty} \int_{-T}^{T} R_x(\tau)\, e^{-i2\pi f\tau}\, \mathrm{d}\tau \\ &= \int_{-\infty}^{\infty} R_x(\tau)\, e^{-i2\pi f\tau}\, \mathrm{d}\tau \\ &= S_x(f)\,. \end{aligned} \tag{4.33}$$

Thus, in summary, the above demonstrates that

$$S_x(f) = \lim_{T\to\infty} \frac{1}{T} E\left[|X_T(f)|^2\right]\,. \tag{4.34}$$

Recalling that $X_T(f)$ has units SU/Hz (where SU stands for "signal units," i.e., whatever units the

signal $x_T(t)$ has), it is clear that $E\left[|X_T(f)|^2\right]$ has units $(\text{SU/Hz})^2$. However, $1/T$ has units of Hz, so that Eq. (4.33) shows that the PSD has units of $(\text{SU}^2)/\text{Hz}$.[4]

Although it is not always literally true, in many cases the mean square of the signal is proportional to the amount of *power* in the signal.[5] The fact that S_x is therefore interpreted as having units of "power" per unit frequency explains the name Power Spectral Density.

Notice that power at a frequency f_0 that does not repeatedly reappear in $x_T(t)$ as $T \to \infty$ will result in $S_x(f_0) \to 0$, because of the division by T in Eq. (4.34). In fact, based on this idealized mathematical definition, any signal of finite duration (or, more generally, any mean square integrable signal), will have power spectrum identical to zero! In practice, however, we do not let T extend much past the support $[T_{\min}, T_{\max}]$ of $x_T(t)$ ($T_{\min\,/\,\max}$ is the minimum (respectively, maximum) T for which $x_T(t) = 0$). Since all signals that we measure in the laboratory have the form $y(t) = x(t) + n(t)$, where $n(t)$ is broadband noise, extending T to infinity for any signal with finite support will end up giving $S_x \approx S_n$.

We conclude by mentioning some important properties of S_x. First, since S_x is an average of the magnitude squared of the Fourier transform, $S_x(f) \in \mathbb{R}$ and $S_x(f) \geq 0$ for all f. A simple change of variables in the definition Eq. (4.19) shows that $S_x(f) = S_x(f)$.

Given the definition Eq. (4.19), we also have the dual relationship

$$R_x(\tau) = \int_{-\infty}^{\infty} S_x(f)\, e^{i2\pi f\tau}\, \mathrm{d}f\,. \tag{4.35}$$

Setting $\tau = 0$ in the above gives

$$R_x(0) = E\left[x(t)^2\right] = \int_{-\infty}^{\infty} S_x(f)\, \mathrm{d}f\,, \tag{4.36}$$

which, for a mean zero signal gives

$$\sigma_x^2 = \int_{-\infty}^{\infty} S_x(f)\, \mathrm{d}f\,, \tag{4.37}$$

Finally, if we assume that $x(t)$ is ergodic in the autocorrelation, that is, that

$$R_x(\tau) = E[x(t)x(t+\tau)] = \lim_{T\to\infty} \int_{-T/2}^{T/2} x(t)x(t+\tau)\mathrm{d}t\,,$$

where the last equality holds for any sample function $x(t)$, then Eq. (4.37) can be rewritten as

$$\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)^2 \mathrm{d}t = \int_{-\infty}^{\infty} S_x(f)\, \mathrm{d}f\,. \tag{4.38}$$

The above relationship is known as *Parsevals Identity*.

[4] Of course, the units can also be determined by examining the definition of Eq. (4.19).

[5] This comes primarily from the fact that, in electrical circuits, the power can be written in terms of the voltage as V^2/Z , or in terms of the current as $I^2 Z$, where Z is the circuit impedance. Thus, for electrical signals, it is precisely true that the mean square of the signal will be proportional to the power. Be forewarned, however, that the mean square of the scaled signal, expressed in terms of the actual measured variable (such as displacement or acceleration), will *not* in general be equal to the average mechanical power in the structure being measured.

This last identity makes it clear that, given any two frequencies f_1 and f_2, the quantity

$$\int_{f_1}^{f_2} S_x(f)\,\mathrm{d}f$$

represents the portion of the average signal power contained in signal frequencies between f_1 and f_2, and hence S_x is indeed a "spectral density."

4.3 Sample Power Spectra

1. **White noise:** $S_{xx}(f) = 1$, where we have power at all frequencies. The corresponding autocorrelation is $R_{xx}(\tau) = \delta(\tau)$, see Fig. 4.3.

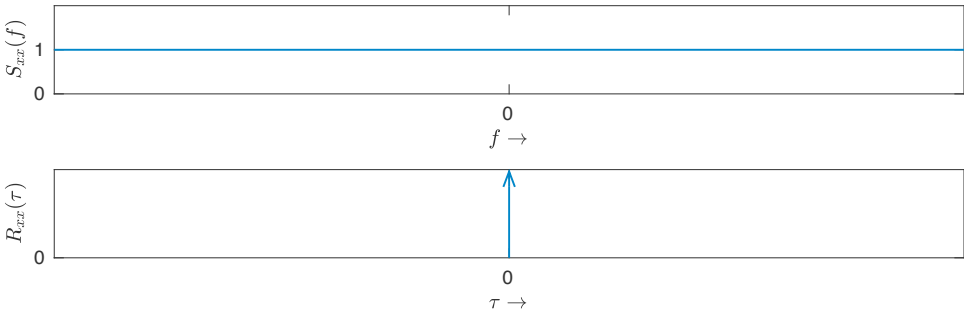

Figure 4.3: White noise signal has power at all frequencies and is uncorrelated for $\tau \neq 0$.

2. **Band limited noise:** $S_{xx}(f) = W(f) = \begin{cases} 1, & 0 \leq |f| \leq f_{BW} \\ 0, & \text{otherwise} \end{cases}$

$$\begin{aligned} R_{xx}(\tau) &= \int_{-\infty}^{\infty} W(f) e^{i2\pi f\tau}\mathrm{d}f \\ &= \int_{-f_{BW}}^{f_{BW}} e^{i2\pi f\tau}\mathrm{d}f \\ &= \frac{1}{2i\pi\tau}\left[e^{i2\pi f_{BW}\tau} - e^{-i2\pi f_{BW}\tau}\right] \\ &= \frac{\sin(2\pi f_{BW}\tau)}{\pi\tau}\,. \end{aligned} \tag{4.39}$$

For the corresponding graphs refer to Fig. 4.4.

Problems

Problem 4.1

Create a sample $\{x_n\}_{n=1}^{1024+14}$[6] of uncorrelated Gaussian random variables (command `randn` in Matlab). Now apply the moving average filter $s_n = 1/15\sum_{i=-7}^{7} x_{n+i}$ to obtain 1024 correlated Gaussian

[6] we only need 1024 for the spectral analysis, additional 14 is to generate 1024 filtered time series

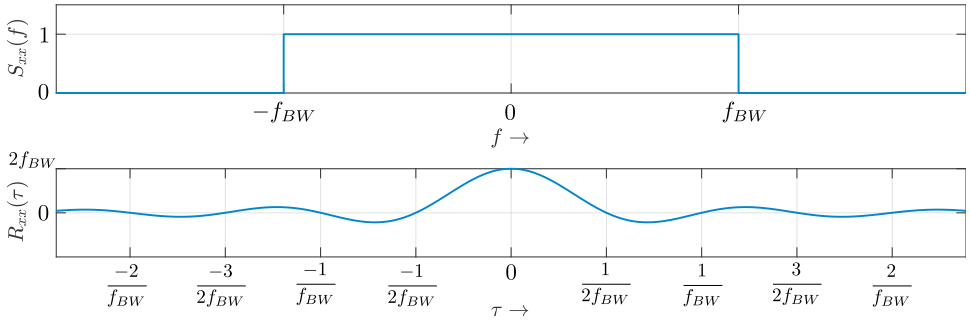

Figure 4.4: Band limited noise has a correlation time of $\frac{1}{2f_{BW}}$.

variates. Estimate the power spectrum (type `> help pwelch` in Matlab) for both data sequences and observe the differences.

Problem 4.2

Create two time series: (1) $\{x_n\}_{n=1}^{4096}$ of uncorrelated Uniform random variables (command `rand` in Matlab), and (2) deterministic evolution of the *Ulam map* $\{y_n\}_{n=1}^{4096}$, which follows the rule $y_0 = 0.1$ and $y_{n+1} = 1 - 2y_n^2$. The values of y_n are measured through a nonlinear observation function $s_n = \arccos(-y_n)/\pi$. Compare the mean, variance and the power spectra of the two time series.

Chapter 5

Experimental Fourier Transforms

5.1 Sampling and Aliasing

Given $x(t)$, we observe only *sampled data* $x_s(t) = x(t)s(t;T_s)$ (Fig. 5.1), where s is called sampling or "comb" function and can be described as

$$s(t;T_s) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s)\,, \tag{5.1}$$

where T_s is *sampling period* and $f_s = 1/T_s$ is *sampling frequency* (Fig. 5.2).

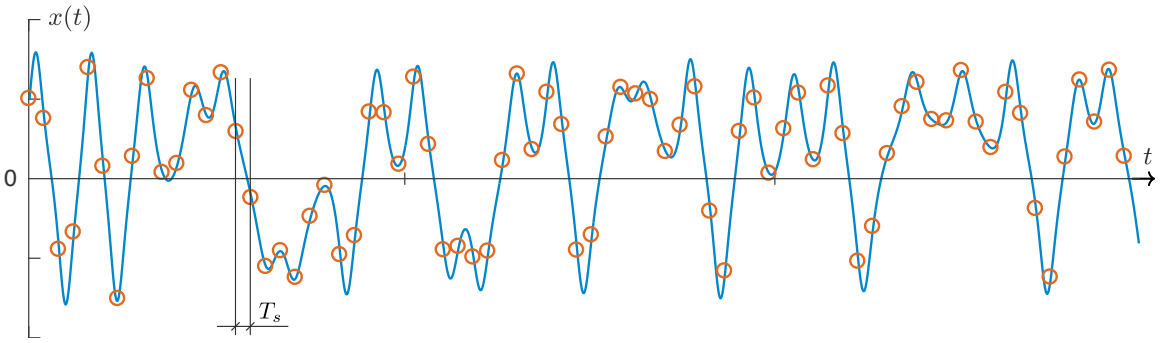

Figure 5.1: We observe only sampled data of a continuous signal every sampling period of T_s

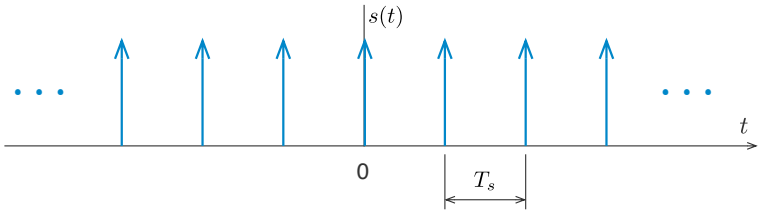

Figure 5.2: Sampling or comb function $s(t)$

Therefore, using the *sifting property* of the Dirac delta function, we get

$$x_s(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s)\,. \tag{5.2}$$

To take the Fourier Transform of x_s, it is useful to note that $s(t;T_s)$ is *periodic* with period T_s. Thus, the corresponding Fourier Series will be:

$$s(t;T_s) = \sum_{n=-\infty}^{\infty} X_n e^{in\frac{2\pi}{T_s}t}\,, \tag{5.3}$$

where

$$X_n = \frac{1}{T_s}\int_{-T_s/2}^{T_s/2}\left[\sum_{n=-\infty}^{\infty}\delta(t - nT_s)\right] e^{-in\frac{2\pi}{T_s}t}\mathrm{d}t = \frac{1}{T_s}e^0 = \frac{1}{T_s}\,. \tag{5.4}$$

So

$$s(t;T_s) = \frac{1}{T_s}\sum_{n=-\infty}^{\infty} e^{i2\pi\frac{n}{T_s}t}\,. \tag{5.5}$$

Now, from previously calculated δ function Fourier Transform pairs, we get

$$S(f) = \frac{1}{T_s}\sum_{n=-\infty}^{\infty}\delta\left(f - \frac{n}{T_s}\right)\,. \tag{5.6}$$

Therefore, since $x_s(t) = x(t)s(t;T_s)$, by convolution we get

$$\mathcal{F}\left[x_s\right] = X_s(f) = X(f) * S(f)\,, \tag{5.7}$$

that is

$$X_s(f) = \frac{1}{T_s}\sum_{n=-\infty}^{\infty}\int_{-\infty}^{\infty} X(s)\delta\left(f - \frac{n}{T_s} - s\right)\mathrm{d}s\,, \tag{5.8}$$

or

$$X_s(f) = \frac{1}{T_s}\sum_{n=-\infty}^{\infty} X\left(f - \frac{n}{T_s}\right)\,. \tag{5.9}$$

That is, sampling results in an ∞ number of periodically spaced copies of the true $X(f)$. Now, we see that if $f_s < 2f_{BW}$, we will get spectral overlap or *aliasing*: high frequencies are "folded" onto lower ones. This leads to the *Nyquist sampling criterion*: $f_s \geq 2f_{BW}$ for accurate frequency domain estimates with sampled data (see Fig. 5.3). Shannon [83] was the one who credited Nyquist when he stated his theorem:

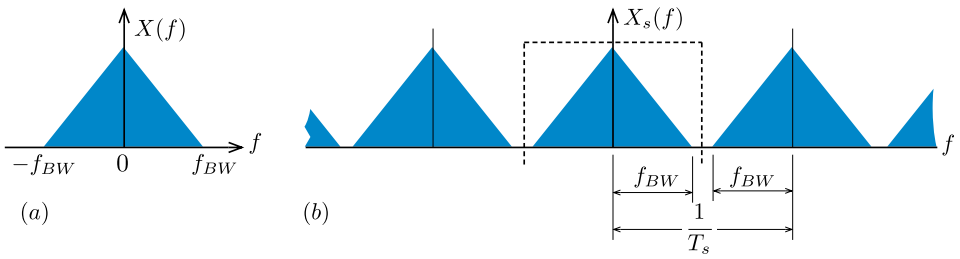

Figure 5.3: (a) true Fourier transform, and (b) Fourier transform of sampled data

Shannon's Sampling Theorem: If a function $x(t)$ contains no frequencies higher than f_{BW} hertz, it is completely determined by giving its ordinates at a series of points spaced $T_s = \frac{1}{2f_{BW}}$ seconds apart.

This means that the time series can be completely reconstructed from its Fourier transform if the Nyquist Criteria is satisfied.

Remark 1: In practice, we may need $f_s = 3F_{BW}$ or even $4f_{BW}$ to avoid aliasing.

Remark 2: Also, we usually use an *anti-aliasing* (low-pass) analog filter, such as shown in Fig. 5.4, which is meant as an example only. If $f_c \approx f_{BW}$, sample at $4f_c = 4f_{BW}$ and discard bottom one-half of the Fourier transform data.

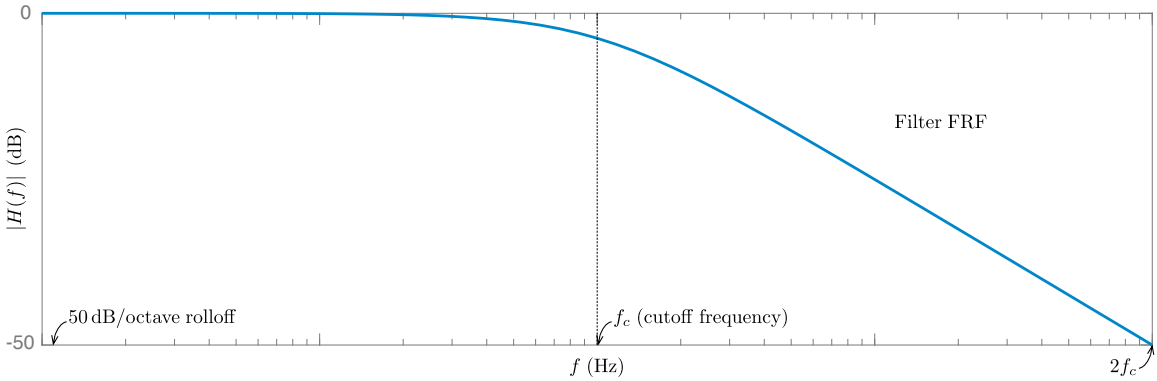

Figure 5.4: Example of analog low-pass filter frequency response function (FRF) magnitude versus frequency

5.2 Windowing

Another artifact of data collection is that we take only a *finite time record* of our data (see Fig. 5.5). Mathematically we can express this as we are sampling the following signal $x_w(t) = x(t)w(t)$, where $w(t)$ is a data collection window function. If our data collection window $w(t)$ is a rectangular window function, we get what is shown in Fig. 5.6.

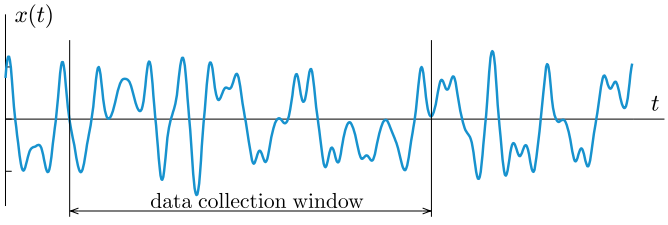

Figure 5.5: We only take a finite time record of data during the data collection window

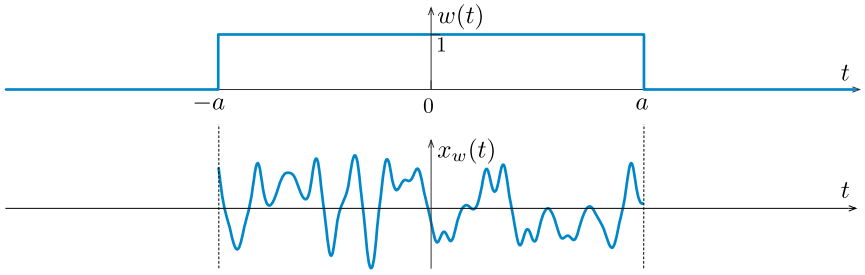

Figure 5.6: Rectangular window function $w(t)$ and the resultant observed signal

Now, we know how to find the Fourier transform of a rectangular window:

$$\begin{aligned} W(f) &= \int_{-\infty}^{\infty} w(t) e^{-2\pi i f t} \mathrm{d}t = \int_{-a}^{a} e^{-2\pi i f t} \mathrm{d}t \\ &= \frac{1}{2\pi i f} \left(e^{2\pi i f a} - e^{-2\pi i f a} \right) = \frac{\sin 2\pi f a}{\pi f} = 2a \,\mathrm{sinc}(2\pi f a)\,, \end{aligned} \tag{5.10}$$

where the sinc function has the form shown in Fig. 5.7.

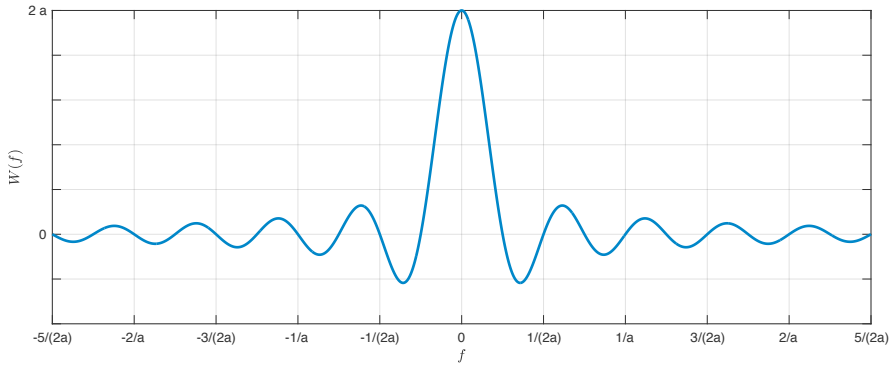

Figure 5.7: Fourier transform of $w(t)$ in Fig. 5.6

As an example of the effect of the window, let $x(t) = \cos 2\pi f_0 t$, then using convolution property we have the following:

$$\begin{aligned} X_w(f) &= X(f) * W(f) \\ &= \int_{-\infty}^{\infty} \frac{1}{2} \left[\delta\left(f + f_0 - s\right) + \delta\left(f - f_0 - s\right) \right] \frac{\sin 2\pi s a}{\pi s} \mathrm{d}s \\ &= \frac{1}{2} \left[\frac{\sin\left(2\pi (f + f_0) a\right)}{\pi (f + f_0)} + \frac{\sin\left(2\pi (f - f_0) a\right)}{\pi (f - f_0)} \right] \end{aligned} \tag{5.11}$$

The results of this transform are shown in Fig. 5.8, which shows that windowing causes *spectral leackage* into neighboring frequencies.

The solution to spectral leakage is to use different window functions. Some of the widely used window functions are shown in Fig. 5.9.

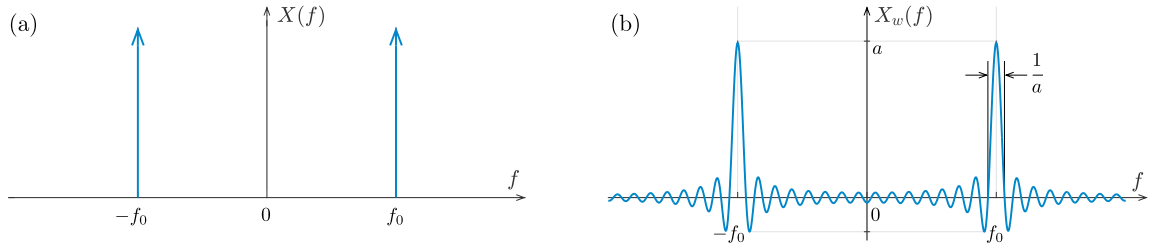

Figure 5.8: True Fourier transforms $X(f) = \mathcal{F}(\cos 2\pi f_0 t)$ (a) and windowed Fourier transform $X_w(f) = \mathcal{F}(w(t)\cos 2\pi f_0 t)$ (b) showing *spectral leakage*

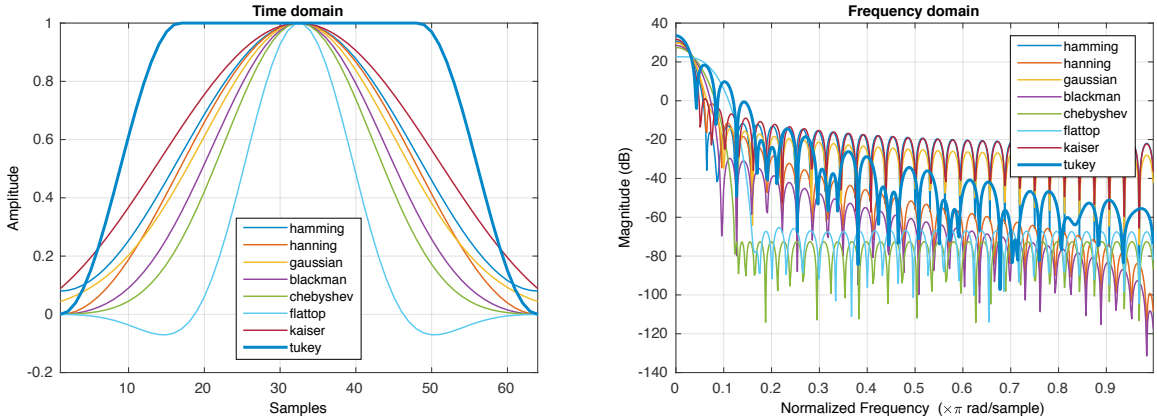

Figure 5.9: Some of the windows from the `window` design tool of MATLAB

.

5.3 Discrete Fourier Transfrom

Continuous Fourier transfrom was defined in Eq. (4.7):

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-i2\pi ft}\, \mathrm{d}t\,. \tag{5.12}$$

However, we only measure over a *finite* time T, with equally-spaced samples with $\Delta t \equiv \Delta = \frac{T}{N}$, where N is the number of samples (see Fig. 5.10). Therefore,

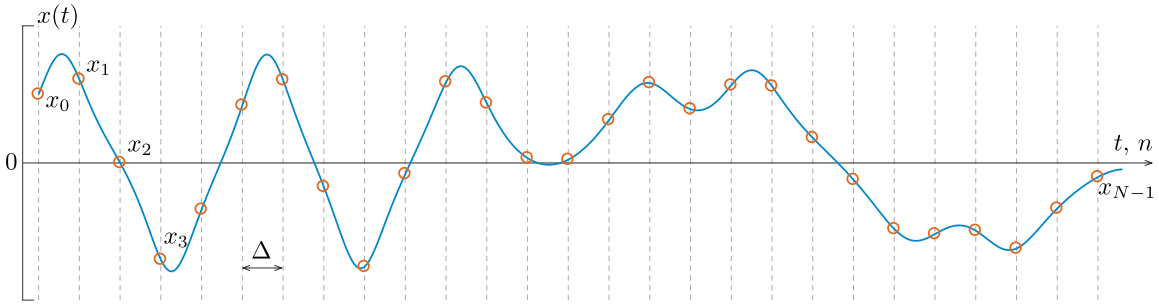

Figure 5.10: We observe only sampled data of a continuous signal every sampling period of T_s

$$X(f) \approx \int_0^T x(t)\, e^{-i2\pi ft}\, \mathrm{d}t \approx \sum_{n=0}^{N-1} x_n e^{-i2\pi f(n\Delta)} \Delta\,, \tag{5.13}$$

where we ignore the fact that $X(f)$ is no longer "true" $\mathcal{F}(x(t))$, and $x_n \equiv x(t_n) = x(n\Delta)$. Now, let $f = \frac{m}{T} \equiv f_m$ and $X(f_m) = \Delta X_m$ (Δ scaling is needed to have correct units), then we get

$$X_m = \frac{1}{\Delta} \sum_{n=0}^{N-1} x_n\, e^{-i2\pi \frac{m}{T}(n\Delta)} \Delta = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{\Delta}{T}(mn)}\,. \tag{5.14}$$

However, $\Delta = \frac{T}{N}$, so $\frac{\Delta}{T} = \frac{1}{N}$. Then,

$$X_m = \sum_{n=0}^{N-1} x_n\, e^{-i2\pi \frac{mn}{N}}\,, \tag{5.15}$$

which is *discrete Fourier transform* or DFT.

We cannot possibly get $x(t)$ or $X(f)$ for t or f between the $t_n = n\Delta$ (i.e., $n\Delta < t < (n+1)\Delta$) or $m\Delta f < f < (m+1)\Delta f$ ($\Delta f = \frac{m+1}{T} - \frac{m}{T} = \frac{1}{T}$), respectively. However, given $\{x_n\}_{n=0}^{N-1}$ we get $\{X_m\}_{m=0}^{N-1}$, and

$$x_n = \frac{1}{N} \sum_{m=0}^{N-1} X_m\, e^{\frac{i2\pi nm}{N}}\,, \tag{5.16}$$

which is called *inverse discrete Fourier transfor* (IDFT) and *discrete* values are transformed *exactly*.

We can prove the above as follows. Given

$$X_m = \sum_{p=0}^{N-1} x_p\, e^{-i2\pi \frac{mp}{N}}\,. \tag{5.17}$$

we can write:

$$
\begin{aligned}
x_n &= \frac{1}{N}\sum_{m=0}^{N-1}\left(\sum_{p=0}^{N-1} x_p\, e^{-i2\pi\frac{mp}{N}}\right) e^{i2\pi\frac{nm}{N}}\,, \\
&= \frac{1}{N}\sum_{p=0}^{N-1} x_p\left(\sum_{m=0}^{N-1} e^{i2\pi\frac{m(n-p)}{N}}\right)\,, \qquad \left(\sum_{m=0}^{N-1} e^{i2\pi\frac{m(n-p)}{N}} = \begin{cases} 0 & \text{for } n\neq p\,, \\ N & \text{for } n=p\,. \end{cases}\right) \qquad (5.18)\\
&= \frac{1}{N}x_n N = x_n\,. \qquad \square
\end{aligned}
$$

In conclusion, the *discrete and inverse discrete Fourier transforms are exact transforms.*

5.3.1 Notes on Fast Fourier Transforms

Fast Fourier Transform (FFT) is a fast algorithm for DFT (see, e.q., Numerical Recipes). It has the following requirements and parameters:

- Time records of N points, Δt between points must satisfy $\frac{1}{\Delta t} \geq 2f_{BW}$ of signal.
- Simplest and fastest implementation is when $N = 2^m$, so data records (buffer) are usually 2^m (the DP 420 has $4K$ or 4096 point buffer).
- We get $\frac{N}{2}$ lines in the frequency domain. Heuristically, the N real data points give $\frac{N}{2}$ complex (amplitude and phase) data points in the frequency domain.
- Δf in the FFT is $\frac{1}{T}$ of data record. Thus, $f_{\max} = \frac{N}{2}\Delta f = \frac{N}{2T}$. However, $T = N\Delta t$ and, thus, $f_{\max} = \frac{1}{2\Delta t} = \frac{f_s}{2}$.

Problems

Problem 5.1

Consider the driven, two-well Duffing's oscillator

$$\ddot{x} + \epsilon\gamma\dot{x} - x + x^3 = \epsilon F \cos(\omega t)$$

which can be written in state variable form as

$$\dot{x} = v$$
$$\dot{v} = x - x^3 + \epsilon(-\gamma v + F cos(\omega t)).$$

In the above expressions, γ is the linear (viscous) damping coefficient, F is the forcing amplitude, and ω is the forcing frequency.[1]

Develop a MATLAB code to simulate this system. You need to be able to generate uniformly sampled time series of any length, with any sampling time t_s (recall that the sampling frequency is $f_s = 1/t_s$, and be very careful when choosing your sampling time).

Experiment with the simulations, and by varying the system parameters and examples of:

1. a periodic, approximately simple harmonic response;
2. a periodic response that is not simple harmonic; and
3. a chaotic (i.e. aperiodic) response.

Plot the *steady state* results in the time domain and in the x–v phase plane. To help you get started, reasonable values for the parameters to use in the simulations are $\epsilon = 0.1$, $\gamma \in [0.7; 2.0]$, $\omega \in [0.9; 1, 1]$, $F \in [1.5; 2.8]$.

Problem 5.2

Using the time series $\{x_i\}$ generated in Exercise 1, make an observable via $\{y_i\} = \{x_i + n_i\}$, where the n_i are independent identically-distributed (IID) mean zero Gaussian random variable with variance σ^2 (be careful when choosing your parameters—we want to approximate real-life situations). Find the digital Power Spectral Density (PSD) for the three types of response found in Exercise 1, using the MATLAB functions `pwelch` or `periodogram` (e.g., at MATLAB prompt type "≫ `help pwelch`"). By experimentation, examine the effect of σ, t_s, the number of points

[1] The parameter ϵ is included here only for convenience when comparing our results to certain theoretical treatments. In particular, the damping and forcing are scaled by ϵ to indicate that the oscillator can be thought of as a perturbation of of a conservative system, with the conservative system corresponding to the case $\epsilon = 0$. See, for example, Guckenheimer and Holmes (1983).

used in the FFT, and windowing (including no window, which is called `rectwin` in MATLAB). A good window to use is the Hanning, however you can experiment to see the differences. Type `help window` at the command line for more information. Plot your results using a semilog or Decibel (dB) scale. Discuss your results: what do you see?

References

John Guckenheimer and Philip Holmes (1983) *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, New York. See in particular Ch. 4.5, Eq. (4.5.18).

Chapter 6

Bifurcations and Poincaré Maps

In this chapter, purely nonlinear phenomenon of *bifurcations* will be considered. Mathematically, bifurcation theory studies the changes in the qualitative or topological structure of the solutions of a family of differential equations. We can say that bifurcation occurs in a dynamical system when a small smooth change of a (bifurcation) parameter causes a sudden 'qualitative' or topological change in the dynamical system's behavior.

6.1 Stability of Equilibrium Points

In nonlinear systems, the transition from stable to unstable behavior is of practical concern, since it can happen due to *perturbations* (global change), or due to the smooth drift in parameters (local bifurcations). During bifurcation one or both of these are possible: (1) one set of solutions can loose stability, while others can become stable; and (2) some solutions can appear or disappear suddenly. To understand bifurcations, we need to understand the stability of fixed points and steady states. We first start by considering stability of the static equilibrium points. Consider a two-well Duffing equation:

$$\ddot{x} + \dot{x} - x + x^3 = 0\,. \tag{6.1}$$

In static equilibrium both velocity and the acceleration are set to zero ($\ddot{x} = \dot{x} = 0$), thus

$$-x + x^3 = 0\,, \tag{6.2}$$

which has three solutions $x = 0$, and $x = \pm 1$.

To examine the stability of these equilibrium points, we study the behavior of small perturbations ($|\delta| \ll 1$) about the equilibrium point:

$$x = x_e + \delta\,, \quad \Rightarrow \quad \dot{x} = \dot{\delta} \quad \text{and} \quad \ddot{x} = \ddot{\delta}\,. \tag{6.3}$$

Then, our equation can be rewritten in terms of δ as:

$$\ddot{\delta} + \dot{\delta} - \delta + \delta^3 + 3x_e^2\delta + 3x_e\delta^2 = 0\,. \tag{6.4}$$

Now ignoring the terms of $\mathcal{O}(\delta^2)$ or higher, we get:

$$\ddot{\delta} + \dot{\delta} + (3x_e^2 - 1)\delta = 0\,. \tag{6.5}$$

Please note, this is equivalent to the Taylor series expansion of Eq. (6.1) about the equilibrium point.

Now we can see that for $x_e = 0$, we have a negative stiffness and thus the solutions to Eq. (6.5) would experience exponentially growing (or unstable) motion. In contrast, the solutions for $x_e = \pm 1$ would experience decaying (or stable) motion. Therefore, we have observed how this nonlinear system behaves locally in the vicinity of equilibrium points. In general, we want to use this local behavior to gain insight into full nonlinear behavior.

6.1.1 Classification of Linear Two-Dimensional Flows

Generically, single-degree-of-freedom oscillator can be linearized about the equilibrium point by the following two-dimensional flow:

$$\begin{aligned} \dot{x} &= ax + bv\,, \\ \dot{v} &= cx + dv\,. \end{aligned} \tag{6.6}$$

For an invariant solution to Eq. (6.6) we are looking for straight line trajectories on a plane:

$$\mathbf{x}(t) = e^{\lambda t}\boldsymbol{\xi}\,, \tag{6.7}$$

where $\mathbf{x}(t) = [x(t),\, v(t)]^T$ is our state vector, $\boldsymbol{\xi}$ is some fixed vector (determining the direction of the invariant trajectory or eigenvector), and λ is the corresponding growth rate or eigenvalue. To find these variables, we substitute Eq. (6.7) into Eq. (6.6) (i.e., $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$), where

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\,, \tag{6.8}$$

which results into $\lambda e^{\lambda t}\boldsymbol{\xi} = e^{\lambda t}\mathbf{A}\boldsymbol{\xi}$. Now, cancelling $e^{\lambda t}$ on both sides results in the following *eigenvalue problem*:

$$\mathbf{A}\boldsymbol{\xi} = \lambda\boldsymbol{\xi}\,, \tag{6.9}$$

or

$$(\mathbf{A} - \lambda\mathbf{I})\boldsymbol{\xi} = 0\,. \tag{6.10}$$

To satisfy the above equation for a nontrivial $\boldsymbol{\xi}$, we need $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$, or

$$\det\begin{pmatrix} a-\lambda & b \\ c & d-\lambda \end{pmatrix} = 0\,, \tag{6.11}$$

which gives us the *characteristic equation*:

$$\lambda^2 - \tau\lambda + \Delta = 0\,, \tag{6.12}$$

where $\tau = \text{trace}(\mathbf{A}) = a + d$ and $\Delta = \det(\mathbf{A}) = ad - bc$. Therefore,

$$\lambda_{1,2} = \frac{\tau \pm \sqrt{\tau^2 - 4\Delta}}{2}\,. \tag{6.13}$$

Thus, the eigenvalues, also called *characteristic exponents* or CEs, depend only on the trace and determinant of $\mathbf{A}$.

For $\lambda_1 \neq \lambda_2$, we can find linearly independent $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ that span the whole phase (state) plane. Thus, any initial condition can be written as:

$$\mathbf{x}_0 = c_1\boldsymbol{\xi}_1 + c_2\boldsymbol{\xi}_2\,, \tag{6.14}$$

which can be used to write the solution corresponding to that initial condition as a linear combination of solutions to $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$:

$$\mathbf{x}(t) = c_1 e^{\lambda_1 t}\boldsymbol{\xi}_1 + c_2 e^{\lambda_2 t}\boldsymbol{\xi}_2\,. \tag{6.15}$$

Therefore, by the existence and uniqueness property this is the *only* solution.

The type of the solution depends on the values of τ and $\tau^2 - 4\Delta$, and we have the following possibilities:

1. $\Delta < 0$, results in two real eigenvalues, with one positive and one negative. This characterizes a *saddle point* equilibrium that has one unstable eigendirection for $\lambda > 0$ and one stable eigendirection for $\lambda < 0$.

2. $\Delta > 0$ and $\tau^2 < 4\Delta$, results in two complex conjugate eigenvalues, for which we could have three cases:

   (a) If $\tau < 0$ we have a *stable spiral* solution;

   (b) If $\tau > 0$ we have a *unstable spiral* solution; and

   (c) If $\tau = 0$ we have a *center* solution.

3. $\Delta > 0$ and $\tau^2 > 4\Delta$, results in two real eigenvalues:

   (a) For $\tau < 0$, both are negative and we have a *stable node*; and

   (b) For $\tau > 0$, both are positive and we have a *unstable node*.

4. $\Delta = 0$, we have $\lambda_1 = 0$ and $\lambda_2 = \tau$, which gives a *non-isolated fixed point* or an equilibrium line span by $\boldsymbol{\xi}_1$ is either attracting ($\tau < 0$) or repelling ($\tau > 0$).

5. $\tau^2 = 4\Delta$, we have repeated eigenvalues that give *degenerate nodes*. In special cases, when the matrix $\mathbf{A}$ is diagonal (i.e., $a = d$ and $b = c = 0$), we can also get *stars*.

For the illustration of all possible phase portraits, please see the Fig. 6.1.

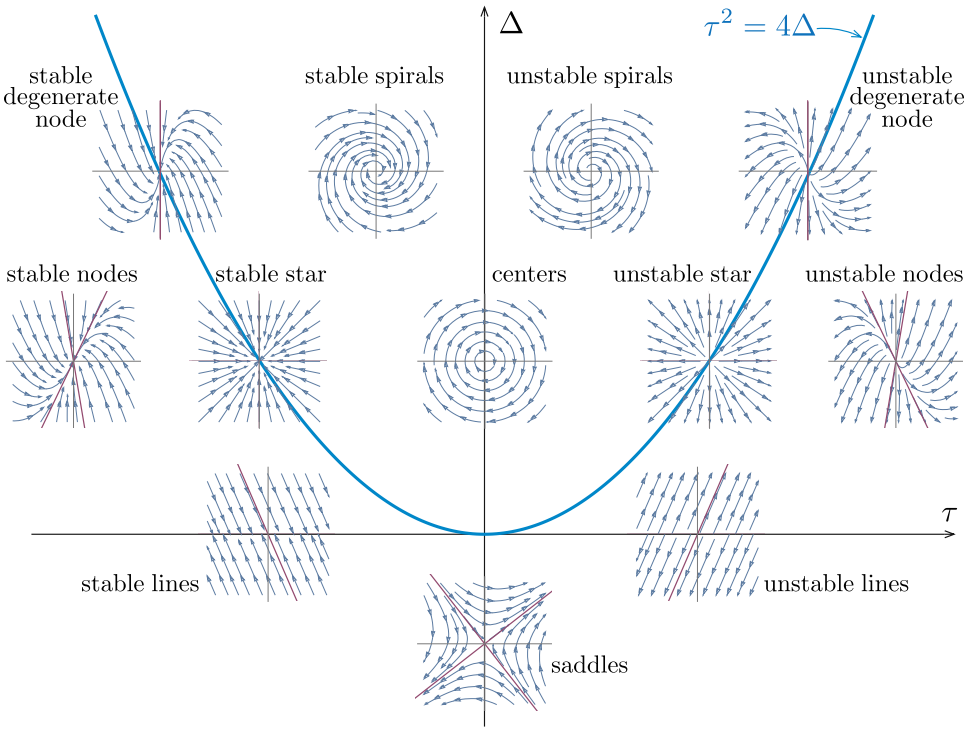

Figure 6.1: Types of two-dimensional static equilibria

6.1.2 Connection to Potential Energy

For one-degree-of-freedom system with a potential function $V = V(x)$, equilibrium points are located at the extrema of the potential determined by solving the following

$$\frac{\mathrm{d}V}{\mathrm{d}x} = 0\,. \tag{6.16}$$

In general, we can expand the potential into Taylor series near the equilibrium point $x = x_e + \delta$:

$$\begin{aligned} V(\delta) &= V(x_e) + \underbrace{\left.\frac{\mathrm{d}V}{\mathrm{d}x}\right|_{x=x_0}}_{0}\delta + \left.\frac{1}{2!}\frac{\mathrm{d}^2V}{\mathrm{d}x^2}\right|_{x=x_0}\delta^2 + \mathcal{O}(\delta^3) \\ &= V(x_e) + \left.\frac{1}{2}\frac{\mathrm{d}^2V}{\mathrm{d}x^2}\right|_{x=x_0}\delta^2 + \mathcal{O}(\delta^3)\,, \end{aligned} \tag{6.17}$$

where $V(x_e)$ is an arbitrary construct and can be set to zero. Then, we can write:

$$V(\delta) = \left.\frac{1}{2}\frac{\mathrm{d}^2V}{\mathrm{d}x^2}\right|_{x=x_0}\delta^2\,. \tag{6.18}$$

Thus, if $\left.\frac{\mathrm{d}^2V}{\mathrm{d}x^2}\right|_{x=x_0} > 0$ we have a local minimum, of $\left.\frac{\mathrm{d}^2V}{\mathrm{d}x^2}\right|_{x=x_0} < 0$ local maximum, and if $\left.\frac{\mathrm{d}^2V}{\mathrm{d}x^2}\right|_{x=x_0} = 0$ we need to go to higher-order approximation (e.g, we may have a purely cubic potential near the equilibrium which would correspond to a semi-stable cusp fixed point).

Now, we can use the following theorem to specify when the equilibrium is stable.

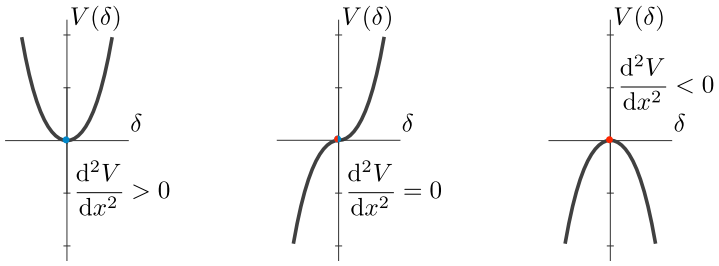

Figure 6.2: Potential energy extrema at an equilibrium points

Lagrange's Theorem: An equilibrium state at which the total potential energy is an isolated minimum (maximum) is necessarily stable (unstable).

In summary, referring to Fig. 6.2 we can say: if a potential energy has a local minimum at an equilibrium point, it is stable; if the potential has a local maximum at an equilibrium, it is unstable; and if the potential has a cusp point, we have a semi-stable equilibrium.

6.2 Bifurcations

To illustrate what are bifurcations, let us consider an example of Duffing Oscillator:

$$\ddot{x} + \gamma\dot{x} - \beta x + x^3 = F\cos\omega t\,. \tag{6.19}$$

Generally, we start the analysis by determining the *static equilibria* of the unforced system ($F = 0$). In static equilibrium both velocity and the acceleration are set to zero ($\ddot{x} = \dot{x} = 0$), thus

$$-\beta x + x^3 = 0\,, \tag{6.20}$$

which has three possible solutions $x = 0$, and $x = \pm\sqrt{\beta}$ if $\beta \geq 0$. As we vary β from $-\infty$ to ∞, our system goes from one real solution ($x = 0$ for $\beta < 0$), to three equilibrium solutions as β passes through 0. This type of change in the possible solutions is called a "pitchfork bifurcation." Please note that $x = 0$ stable spiral solution becomes unstable spiral as β passes through 0 from negative β to positive β values. Therefore, bifurcations are also associated with changes in the stability. The illustration of this bifurcation is shown in Fig. 6.3.

6.2.1 Classification of Instability

The nature of the loss of stability relates to how the eigenvalues (i.e., CEs) exit the negative real half-plane as a single bifurcation (or control) parameter is changed, which also determines the bifurcation type. For example, if a real eigenvalue changes its sign (i.e., $\Delta < 0 \to \Delta > 0$) we go from saddle type fixed point to a node type of fixed point. This type of bifurcation is called *saddle-node*, *fold*, or divergence bifurcation, see Fig. 6.4(a). In mechanical or engineered systems this corresponds

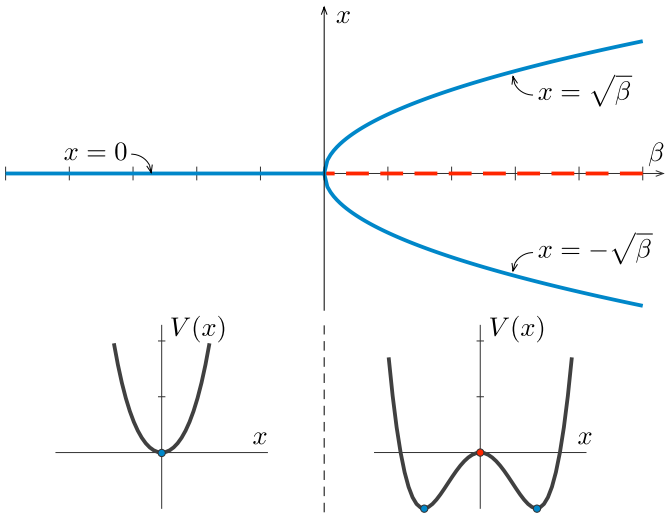

Figure 6.3: Pitchfork bifurcation. Blue curves indicate stable equilibrium solutions and the red dashed line indicates the unstable equilibrium solution.

to buckling (stiffness goes to zero and then becomes negative). In contrast, if the real part of the complex conjugate eigenvalues change sign (i.e., $\tau < 0 \rightarrow \tau > 0$) we effectively change the sign of damping. This type of bifurcation is called *Hopf*, see Fig. 6.4(b), and common mechanical systems examples are flatter or vortex induced vibrations.

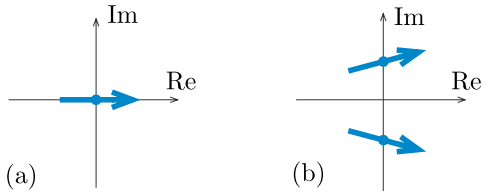

Figure 6.4: CE behavior during the loss of stability due to saddle-node (a) and Hopf (b) bifurcations

6.3 Stability and Bifurcation of Cycles

We start by considering the Logistic map equation:

$$x_{n+1} = \mu x_n (1 - x_n)\,. \tag{6.21}$$

Equilibria or "fixed points" for the map are actually *periodic orbits* since $x_{n+1} = x_n$ defines the fixed point. To find fixed points of the map, set $x_n \equiv x$ for "period 1" or P-1 fixed point to get:

$$x = \mu x (1 - x)\,, \tag{6.22}$$

or

$$x(1 - \mu) = -\mu x^2\,. \tag{6.23}$$

So $x = 0$, and $x = \frac{\mu-1}{\mu}$ if $\mu \neq 0$, are the P-1 orbits or fixed points.

To evaluate the stability near $x = \frac{\mu-1}{\mu}$, let $x_n = \frac{\mu-1}{\mu} + u_n$, where $u_n \ll 1$ is a small perturbation from the fixed point. Thus, the Logistic map can be written as:

$$\frac{\mu-1}{\mu} + u_{n+1} = \mu\left(\frac{\mu-1}{\mu} + u_n\right)\left(1 - \frac{\mu-1}{\mu} - u_n\right), \tag{6.24}$$

or

$$u_{n+1} = (2-\mu)u_n - \mu u_n^2 . \tag{6.25}$$

Remembering that we have only *small* perturbations, we get

$$u_{n+1} \cong (2-\mu)u_n . \tag{6.26}$$

For stability we want $|u_n|$ bounded $\forall\, n$, and for *asymptotic* stability, $|u_n| \to 0$ as $n \to \infty$. Therefore, we require $|2-\mu| \leq 1$, or $-1 \leq 2-\mu \leq 1$. Thus, for stability we need:

$$1 \leq \mu \leq 3 . \tag{6.27}$$

Please note, that when $\mu = 1$, $x = \frac{\mu-1}{\mu}$ "collides" with $x = 0$ solution. Also, $2-\mu$ switches sign at $\mu = 2$ as shown in Fig. 6.5(a). Then the approach to the steady state P-1 orbit will have either monotonic as in Fig. 6.5(b) or oscillatory as in Fig. 6.5(c) character.

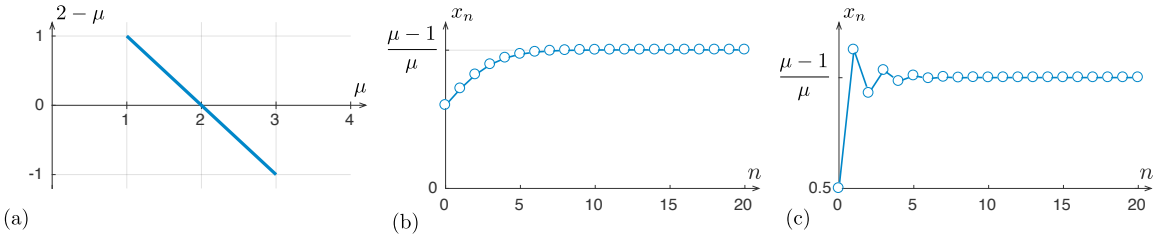

Figure 6.5: In the stability region for $x = \frac{\mu-1}{\mu}$ equlibrium, $2-\mu$ switches sign at $\mu = 2$ (a), for $1 < \mu < 2$ we have a monotonic approach to equilibrium (b), and for $2 < \mu < 3$ we have an oscillatory approach to equilibrium (b).

Now the question is what happens when $\mu = 3$? To answer this, consider a "period 2" (P-2) fixed point $x_{n+2} = x_n$:

$$x_{n+2} = \mu x_{n+1}(1 - x_{n+1}) = \mu^2 x_n(1-x_n)\left(1 - \mu x_n(1-x_n)\right) . \tag{6.28}$$

We seek the solution for $x_{n+2} = x_n \equiv x$:

$$x = \mu^2 x(1-x)\left(1 - \mu x(1-x)\right) . \tag{6.29}$$

The roots to the above equation are the roots for the P-1 fixed points (as P-1 orbits are also a P-2 fixed points) plus

$$x_{a,b} = \frac{1}{2} + \frac{1}{2\mu} \pm \frac{1}{2\mu}\sqrt{\mu^2 - 2\mu - 3} . \tag{6.30}$$

For roots $x_{a,b}$ to exist, we need to satisfy the following condition

$$\mu^2 - 2\mu - 3 \geq 0\,,$$

which is only satisfied for $\mu \leq -1$ and $\mu \geq 3$. Therefore, at $\mu = 3$, the P-1 fixed point $x = \frac{\mu-1}{\mu}$ becomes unstable and $x_{a,b}$ become stable (which we have not shown here). Bifurcation points like this (see Fig. 6.6, when stable P-1 fixed point looses stability and P-2 orbits become stable, are called *period-doubling bifurcation points*.

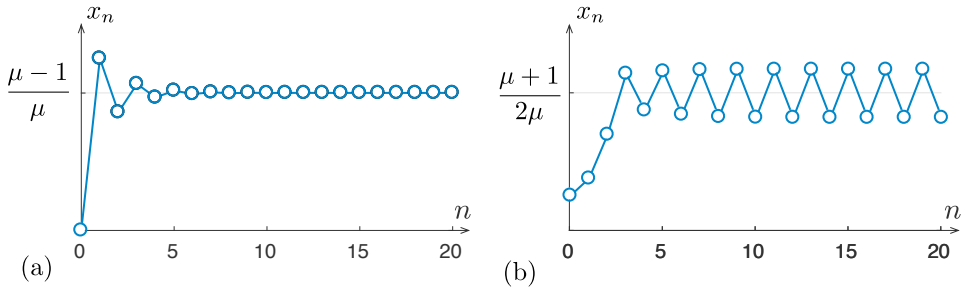

Figure 6.6: Sample P-1 orbit for $2 < \mu < 3$ (a), and P-2 orbit for $\mu = 3.1$ (b).

Doing all this, we are trying to make a point that with maps we can study bifurcations of periodic motions (and even nonperiodic or chaotic motions) since they (i.e., periodic orbits) "look like" static equilibria.

6.4 Dynamic Bifurcations in Continuous Systems

Study of stability of continuous dynamical systems can be converted into study of stability of maps using idea of *Poincaré Map*. We start exploring this idea by considering an example of dynamical system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})\,, \tag{6.31}$$

where $\mathbf{x} \in \mathbb{R}^3$. A solution starting from some initial condition $\mathbf{x}(0) = x_0$ can be written in the following form:

$$\mathbf{x}(t) = \mathbf{X}(x_0, t) = \mathbf{X}_t(x_0)\,. \tag{6.32}$$

For a fixed time $t = \tau$, $\mathbf{X}_\tau(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}^3$. This can be used to develop a corresponding map equation:

$$\mathbf{x}_{n+1} = \mathbf{X}_\tau(\mathbf{x}_n)\,, \tag{6.33}$$

where $x_{n+1} = x(n\tau)$. Then, if the original continuous system has a P-1 solution with a period of $\tau = T$ (i.e., $\mathbf{X}(x_0, t) = \mathbf{X}(x_0, t + T)$), we can write:

$$x_{n+1} = \mathbf{X}_T(x_n) = x_n\,. \tag{6.34}$$

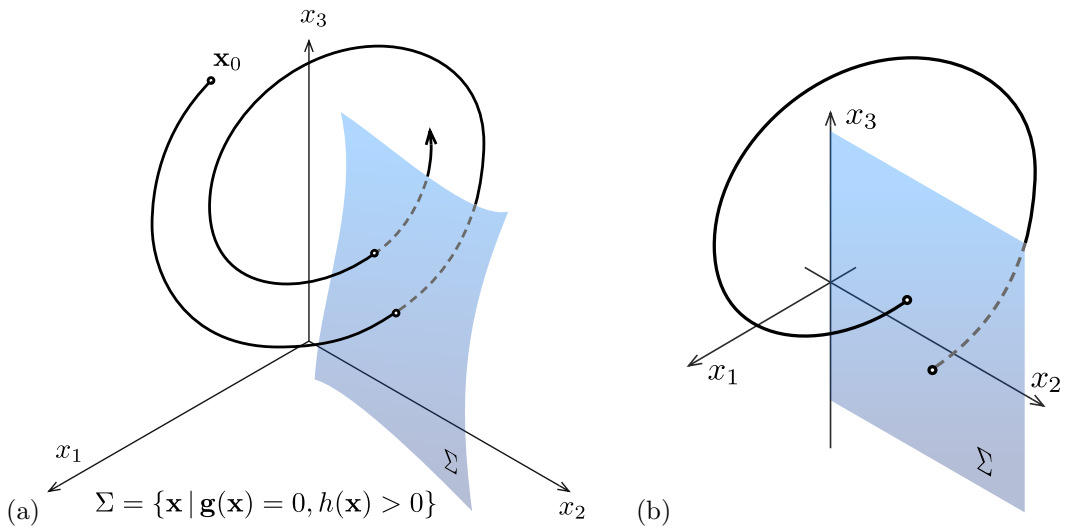

Figure 6.7: Poincaré section or surface of section (a), and its simple example with $g(\mathbf{x}) = x_1 = 0$ and $h(\mathbf{x}) = x_2 > 0$ (b).

Alternatively we can define a surface or manifold in $\mathbb{R}^3$ using:

$$\Sigma = \{\mathbf{x} \,|\, \mathbf{g}(\mathbf{x}) = \mathbf{0}, h(\mathbf{x}) > 0\} \,, \tag{6.35}$$

where we require that $\mathbf{f}(\mathbf{x}) \perp \Sigma$, $\forall \mathbf{x} \in \Sigma$, and $\mathbf{f} \cdot \nabla \mathbf{g} > 0$, $\forall \mathbf{x} \in \Sigma$. The first condition requires that trajectories pierce Σ manifold perpendicular to its surface, and the second condition requires to count only points that are generated by trajectories going through Σ surface from only one side. Simplest case for $\mathbf{g}$ and h functions is:

$$g(\mathbf{x}) = x_1 = 0 \quad \text{and} \quad h(\mathbf{x}) = x_2 > 0 \,. \tag{6.36}$$

Therefore, the Poincar section Σ is a half-plane $x_1 = 0$, $x_2 > 0$. The point is that Σ is transverse to the flow $\mathbf{X}_T$.

Given Σ, *uniqueness* of solution to ODE defines a *map* form:

$$\mathbf{x}_{n+1} = \mathbf{X}_{\tau_n}(\mathbf{x}_n) \triangleq \mathbf{P}(\mathbf{x}_n) \,, \tag{6.37}$$

where $\mathbf{x}_n \in \Sigma$ and τ_n is defined by first t such that $\mathbf{x}_{n+1} \in \Sigma$. Therefore, $\mathbf{P} : \Sigma \to \Sigma$ is the Poincaré map as shown in Fig. 14.2(a). Now, we can ask a question: what does a fixed point of $\mathbf{P}$, $\mathbf{x}^* = \mathbf{P}(\mathbf{x}^*)$, mean? Let us look at Fig. 14.2(b), where we can see that if $\mathbf{x}^* = \mathbf{P}(\mathbf{x}^*)$, $\mathbf{x}^*$ is the intersection of a P-1 orbit with Σ. Thus, each fixed point of $\mathbf{P}$ corresponds to a *unique periodic orbit.*

Remark: In equation

$$\mathbf{x}_{n+1} = \mathbf{P}(\mathbf{x}_n) = \mathbf{X}_{\tau_n}(\mathbf{x}_n) \,, \tag{6.38}$$

the τ_n, which are the times between visits to Σ need not be equal. In other words, the discrete-time map defined by $\mathbf{P}$ can be *asynchronous.*

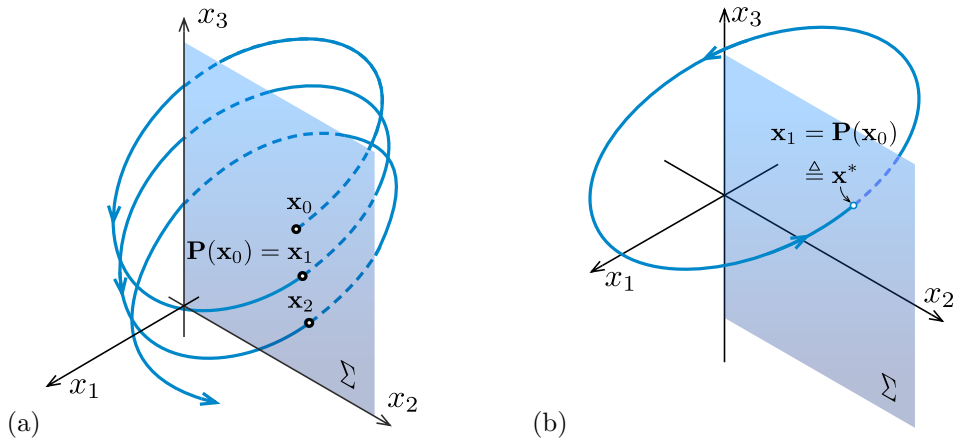

Figure 6.8: Poincaré map $\mathbf{P} : \Sigma \to \Sigma$ (a), and its P-1 fixed point $\mathbf{x}^* = \mathbf{P}(\mathbf{x}^*)$ or P-1 orbit (b).

6.4.1 Periodic Vector Fields

Consider a driven oscillator:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)\,, \tag{6.39}$$

where $\mathbf{f}(\mathbf{x}, t + T) = \mathbf{f}(\mathbf{x}, t)$ and $T = 2\pi/\omega$ (i.e., ω is the forcing frequency). If $\mathbf{x} \in \mathbb{R}^2$, then phase space $\mathcal{S} = \mathbb{R}^2 \times \mathbb{S}^1$ (i.e., direct product of a plane and a circle). For this particular case, a commonly used definition of Poincaré section is:

$$\Sigma = \left\{ (\mathbf{x}, t) \;\middle|\; t \mod \frac{2\pi}{\omega} = \varphi \right\}. \tag{6.40}$$

This is called a *stroboscopic* Poincaré section. Figs. 6.9 and 6.10 show how data is sampled every $T = 2\pi/\omega$ time period at a constant forcing phase φ.

Remark: In numerical simulations, the easiest way to do this is to set *time step*:

$$\Delta t = \frac{2\pi}{\omega n}\,, \tag{6.41}$$

for $n \in \mathbb{Z}$. Then every n steps gives $\mathbf{x} \in \Sigma$. Furthermore, in this way you get n different Σ's, which could be useful for animating the Poincaré section dynamics.

6.4.2 Stability and Bifurcation of Periodic Orbits

For linear systems, we have a Floquet Theory to study local stability based on *Monodromy Matrix Eigenvalues* of the global solution for linear ordinary differential equations with periodically time-varying coefficients. With Poincaré section we can develop similar procedure for evaluating stability of close orbits for nonlinear dynamical systems. For a P-k periodic orbit we have $\mathbf{P}^k(\mathbf{x}^*) = \mathbf{x}^*$. The corresponding solution to our original dynamical system Eq. (6.39) can be expressed as:

$$\mathbf{x}(t, \mathbf{x}^*) = \mathbf{X}_t(\mathbf{x}^*)\,. \tag{6.42}$$

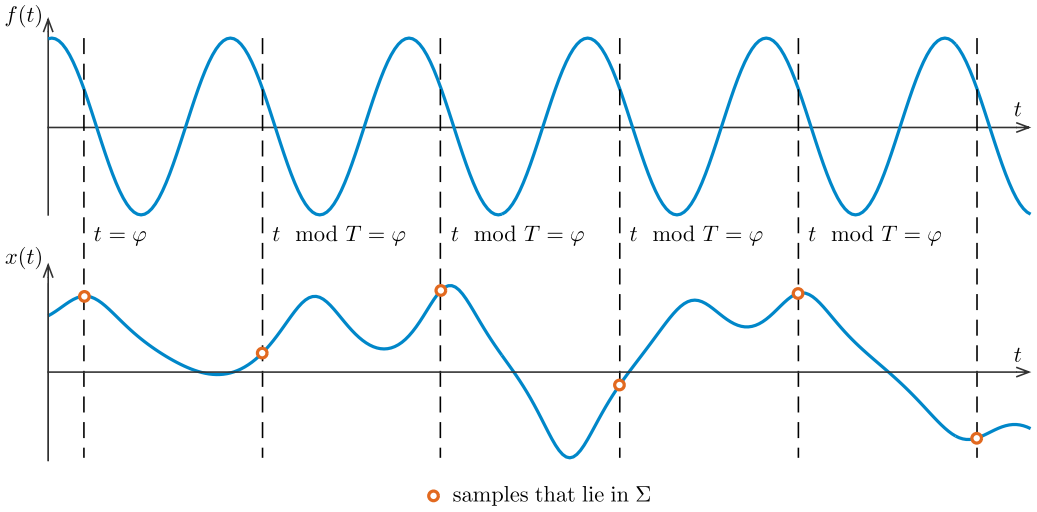

Figure 6.9: Sampling for a stroboscopic Poincaré Section, where $f(t) = f\cos\frac{2\pi}{T}t$, and $x(t)$ is one of the coordinates of the response.

Now we introduce a small perturbation to the initial condition $\mathbf{x}_0 = \mathbf{x}^* + \boldsymbol{\eta}_0$ ($\|\boldsymbol{\eta}_0\| \ll 1$), then we can use $\mathbf{x}_n = \mathbf{x}^* + \boldsymbol{\eta}_n$ in $\mathbf{x}_{n+1} = \mathbf{P}(\mathbf{x}_n)$ to get:

$$\mathbf{x}^* + \boldsymbol{\eta}_{n+1} = \mathbf{P}(\mathbf{x}^* + \boldsymbol{\eta}_n) = \mathbf{P}(\mathbf{x}^*) + D\mathbf{P}(\mathbf{x}^*)\boldsymbol{\eta}_n + \mathcal{O}(\|\boldsymbol{\eta}_n\|^2)\,, \tag{6.43}$$

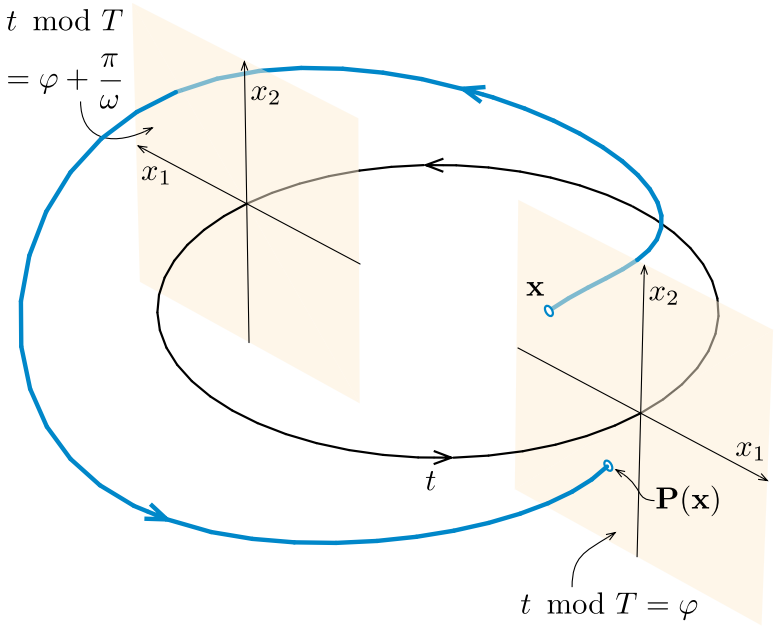

Figure 6.10: Stroboscopic sampling in the phase space $\mathbb{R}^2 \times \mathbb{S}^1$, where the black circle represents time as a forcing phase and blue curve is the trajectory.

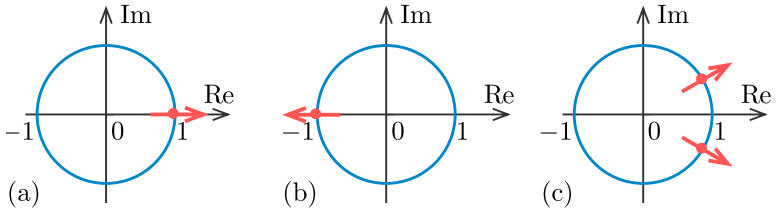

Figure 6.11: Instabilities arise as CMs leave the unit circle during saddle-node (a), period-doubling (b), and Neimark (c) bifurcations.

where $D\mathbf{P}(\mathbf{x}^*)$ is a Jacobian of the Poincaré map evaluate at $\mathbf{x}^*$. Then,

$$\boldsymbol{\eta}_{n+1} \approx D\mathbf{P}(\mathbf{x}^*)\boldsymbol{\eta}_n \,. \tag{6.44}$$

Now the fixed point $\mathbf{x}^*$ is stable if $\boldsymbol{\eta}_n \to \mathbf{0}$ as $n \to \infty$, which is equivalent to requiring that the magnitude of all the eigenvalues—called *characteristic multipliers* (CMs)—of the Jacobian are less than unity.

For a two-dimensional Poincaré section, the Poincaré map near the $\mathbf{x}^*$ Eq. (6.44) can be linearized as:

$$\begin{aligned} \eta_{n+1} &= a\eta_n + b\mu_n \\ \mu_{n+1} &= c\eta_n + d\mu_n \end{aligned} \tag{6.45}$$

or

$$\boldsymbol{\eta}_{n+1} = \mathbf{A}\boldsymbol{\eta}_n \,. \tag{6.46}$$

where $\mathbf{A} = D\mathbf{P}(\mathbf{x}^*)$, and its eigenvalues are also CMs. If one of CMs is on the unit circle fixed point is termed *hyperbolic*. CMs are related to characteristic exponents (CEs) from the ODE solution as $\text{CM} = e^{\text{CE}}$. Bifurcations of cycles happen when the CMs cross the unit circle. Some of the examples of bifurcations are shown in Fig. 6.11: (a) saddle-node or jump bifurcation where cycle looses stability and settles to another periodic solution; (b) period-doubling bifurcation when P-k becomes P-$2k$ and eventually (as $k \to \infty$) leads to chaos; and (c) Neimark bifurcation which is not common in mechanical systems.

Problems

Problem 6.1

[Bifurcation Diagram for Logistics Map] Consider a logistics map equation:

$$x_{n+1} = Ax_n(1 - x_n)\,. \tag{6.47}$$

Run the simulations with the initial condition $X_0 = 0.1$, for different values of A starting from $A = 2.9$ and going up to $A = 4$. Make increments in A small enough, e.g., $\Delta A < 0.01$ (e.g., $\Delta A = 0.001$). For each magnitude of A record long time series and retain only the last 1,000 (or more, e.g., 4,000) points representing its *steady state* response. Now plot the bifurcation diagram, by plotting all the response points versus the corresponding value of A. What can you see in the diagram?

Problem 6.2

Numerically integrate the Lorenz system described in Problem 3.2 and perform a Poincaré section. Namely, record (x, z) every time the y-coordinate equals zero and its derivative is negative. You can accomplish by providing `options` to `ode45` MATLAB program by first setting it through `options = odeset('Events',@events);`, where `events` is the function that specifies the when to sample the flow (i.e., you need to specify $y = 0$ and $\dot{y} < 0$ conditions). I would suggest reading ODE Event Location on Mathworks site and following their example. Once you have the points, plot the Poincaré section by plotting z vs x as points.

Problem 6.3

Numerically integrate the Lorenz system described in Problem 3.2 and collect all the maxima of y variable. Again, you can use `events` to accomplish this, or use `findpeaks` command in MATLAB on y time series that is sampled sufficiently. Plot the series of the maxima Y_n versus Y_{n+1} and compare it to the attractor of Problem 6.2.

Chapter 7

Delay Coordinate Embedding

Up to this point, we have known our state space explicitly. But what if we do not know it? How can we then study the dynamics in *phase space*? A typical case is when our data is a *scalar time series* (e.g., temperature measured at uniform time intervals in a specific geographic point). Let us consider a continuous time systems with *true* state $\mathbf{u} \in \mathcal{S}$, where $\mathcal{S}$ is our phase space.[1] Then our measured signal is

$$x_n \triangleq x\left(\mathbf{u}(t_n)\right) + \eta_n \,, \tag{7.1}$$

where η_n is measurement noise. If we have uniform time sampling Δt (which is a generic case), $t_n = n\,\Delta t$. Thus, we usually have a *time series*

$$\{x_n\}_{n=1}^{N} = \{x_1, x_2, x_3, \ldots, x_N\} \,. \tag{7.2}$$

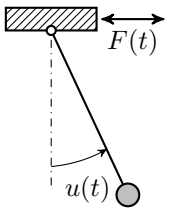

Figure 7.1: Driven pendulum

Example 1: Driven Pendulum

We consider a driven pendulum in Fig. 7.1, described by the following equation:

$$\ddot{u} + \gamma\dot{u} + \sin u = F\cos\omega t \,, \tag{7.3}$$

[1] In general, $\mathcal{S}$ is not even known precisely.

or

$$\begin{aligned} \dot{u} &= v\,, \\ \dot{v} &= -\gamma v - \sin u + F\cos\theta\,, \\ \dot{\theta} &= \omega\,, \end{aligned} \tag{7.4}$$

or

$$\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u})\,, \tag{7.5}$$

where $\mathbf{u} = [u, v, \theta]^T \in \mathbb{S}^1 \times \mathbb{R}^1 \times \mathbb{S}^1$. Now let us assume that we measure only the first coordinate u as

$$x = h\left(\Pi_1(\mathbf{u})\right) = cu\,, \tag{7.6}$$

where $h : \mathbb{S} \to \mathbb{S}$ is the angular position sensor function, $\Pi_1 : \mathbb{S} \times \mathbb{R} \times \mathbb{S} \to \mathbb{S}$ is the projection onto the first coordinate $u \in \mathbb{S}$, and c is the linear sensitivity ($c = h'(\mathbf{u}^*)$).

In this example we know $\mathcal{S}$, and the question is if we can reconstruct the dynamics in the phase space and estimate all the relevant system parameters by just having time series of x.

Example 2: Kicked Elastica

As an another example we consider an experimental system [25] shown in Fig. 7.2. Here, we have

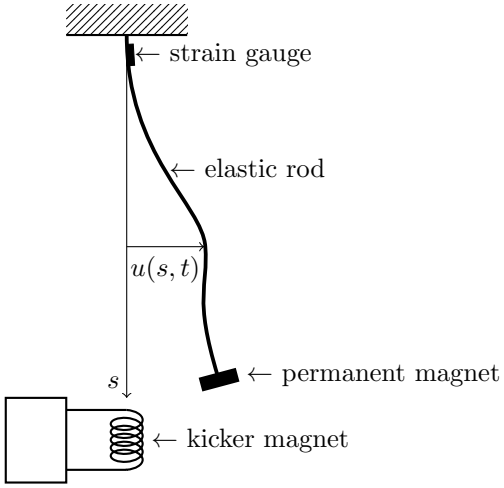

Figure 7.2: Kicked elastica

a cantilevered elastic rod with a permanent magnet mounted at its free end. The kicker magnet located near the free end of the rod applies transverse load to the rod tip as it passes over it. The rod displacement can be expressed in terms of its normal modes as

$$u(s,t) = \sum_{n=1}^{\infty} q_n(t)\varphi(s)\,, \tag{7.7}$$

where $\varphi_n : [0, L] \to \mathbb{R}$ are normal modes of the system satisfying the boundary conditions, and the $q_n(t)$ are the corresponding time coordinates. The corresponding velocity is $v = \dot{u}$. Therefore,

$\mathbf{u} = [u, v]^T \in \mathcal{S}$ is the ∞-dimensional function space. We measure strain $x(t)$ at $s = s^*$. The strain $x \propto \kappa(s^*) \approx cu''(s^*)$, where κ is the curvature of the rod and the approximation is valid for small u and u'. The question here would be can we use measure strain time series $\{x\}_{n=1}^{N}$ to reconstruct the observed dynamics of the elastic rod. Namely, can we determine the number of the active modes and their contribution to the response at various value of the supply voltage to the kicker magnet? Can we develop a bifurcation diagram while varying supply voltage to the kicker magnet? etc.

Example 3: Walking Human

We have seen many examples of the studies on humans kinematics [80, 85] shown in Fig. 7.3, where their gait dynamics are recorded using goniometers (for joint angles) or other instruments (such as video based motion capture). Here, the state of human locomotion $\mathbf{u} \in \mathcal{S}$ is unknown. We do not even know what is our phase space $\mathcal{S}$, which might be collection of joint angles, their velocities, plus many unknown variables. We would usually measure some joint angle $x = c\,\theta$, where θ is the true joint angle, or track motion using video based system, etc.

Figure 7.3: Examples of studies relating to tracking and identifying human kinematics: elite cyclist on the left [85] and soldier walking on the right [80].

A pertinent question, when we do not know $\mathcal{S}$: is it reasonable to believe that one even exists? Additionally, can we use the measured kinematic variable to track and predict other physiological processes like muscle fatigue or oxygen consumption?

7.1 Delay Reconstruction

Given the scalar measurement $\{x_n\}_{n=1}^N$, we can attempt to form a multi-dimensional observable using *delay reconstruction*. That is, given $\{x_1, x_2, \ldots\}$, we define:

$$\mathbf{x}_n = [x_n, x_{n-\tau}, \ldots, x_{n-(d-2)\tau}, x_{n-(d-1)\tau}]^T \in \mathbb{R}^{d\times 1}, \tag{7.8}$$

where τ is the *lag* or *delay time* ($\tau\Delta t$) and d is the *embedding dimension*. A simple illustration of three-dimensional reconstruction is shown in Fig. 7.4.

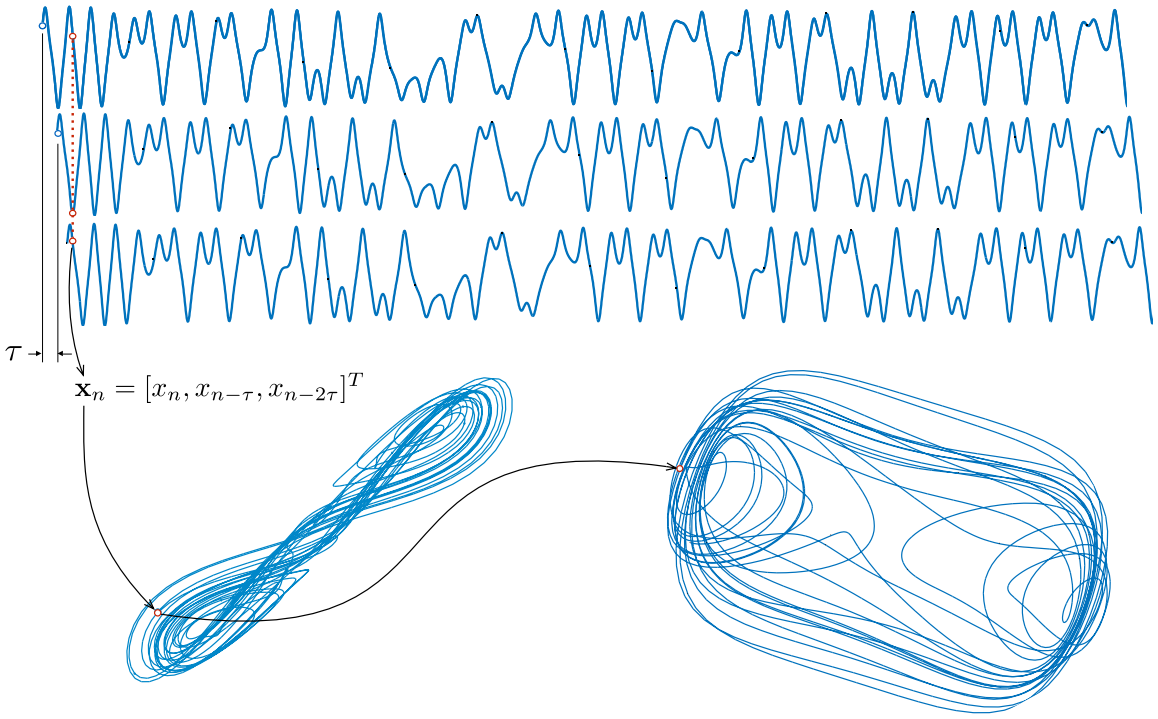

Figure 7.4: Illustration of delay reconstruction of scalar time series into a three-dimensional phase space trajectory using τ delay time

Why do we think that the delay reconstruction might work? Consider

$$\begin{aligned}\mathbf{x}_n &= [x_n, x_{n-\tau}, \ldots, x_{n-(d-2)\tau}, x_{n-(d-1)\tau}]^T \in \mathbb{R}^{d\times 1},\\ &= [x(\mathbf{u}(t_n)), x(\mathbf{u}(t_{n-\tau})), \ldots, x(\mathbf{u}(t_{n-(d-2)\tau})), x(\mathbf{u}(t_{n-(d-1)\tau}))]^T \in \mathbb{R}^{d\times 1}\,.\end{aligned} \tag{7.9}$$

If our time series are from a deterministic system then $\mathbf{u}(t_n - \tau) = \mathbf{F}(\mathbf{u}(t_n))$.[2] Thus,

$$\begin{aligned}\mathbf{u}(t_n - 2\tau) = \mathbf{F}(\mathbf{u}(t_n - \tau)) = \mathbf{F}(\mathbf{F}(\mathbf{u}(t_n))) = \mathbf{F}^2(\mathbf{u}(t_n))\,,\\ \mathbf{u}(t_n - (d-1)\tau) = \mathbf{F}^d(\mathbf{u}(t_n))\,,\end{aligned} \tag{7.10}$$

and

$$\mathbf{x}_n = \mathbf{G}(\mathbf{u}(t_n)) \equiv \mathbf{G}(\mathbf{u}_n)\,. \tag{7.11}$$

[2] Think of $\mathbf{u}(t_n)$ as an initial condition and integrate backwards.

Or, at least, we might hope to be able to do so. To have any hope of this working, we need to get the dimension d and delay τ right.

In summary, as shown in Fig. 7.5, our deterministic trajectory $\varphi_t(\mathbf{u}_0) : \mathcal{S} \to \mathcal{S}$ evolves on a manifold $\mathcal{A} \subset \mathcal{S}$, with dim $\mathcal{A} = k$. We are looking for a delay reconstruction $\Phi : \mathcal{S} \to \mathbb{R}^d$, which is a composition of the measurement $x : \mathcal{S} \to \mathbb{R}$ and the embedding procedure $\mathbf{e} : \mathbb{R} \to \mathbb{R}^d$. The delay embedding will map $\mathbf{u}_0$ through $\mathbf{x}_0 = \Phi(\mathbf{u}_0)$ to $\mathbf{x}_0$, and the embedding will have a trajectory $\hat{\varphi}_t(\mathbf{x}_0)$.

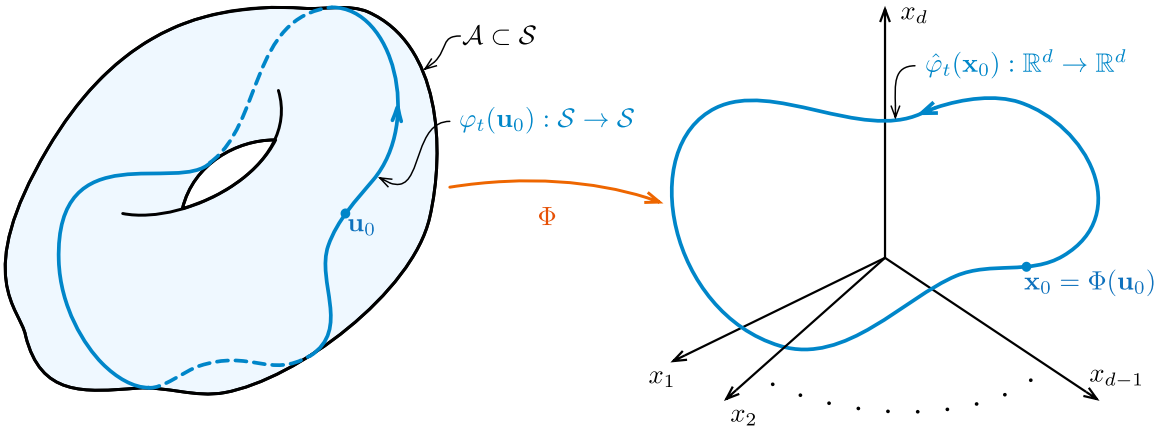

Figure 7.5: Illustration of delay coordinate reconstruction embedding procedure

The big question is: when will this process work properly? We would like all the *topological properties* to be preserved (i.e., dimension and type of orbit), as well as key *dynamical* properties like *stability*, and more generally we would like $\hat{\varphi}_t$ to be "like" φ_t.

The big answer developed by Whitney, Takes, Sauer, *at al.* is:

Embedding Theorem: Consider an invariant manifold $\mathcal{A} \subset \mathcal{S}$ with capacity (i.e., box counting) dimension D_F. Then *almost every*[a] smooth[b] $x : \mathcal{S} \to \mathbb{R}$ and delay time τ, the delay map $\Phi(x, \varphi, \tau) : \mathcal{S} \to \mathbb{R}^d$ with $d > 2D_F$ is an *embedding* of $\mathcal{A}$ into $\mathbb{R}^d$ if:

1. There are *no* periodic orbits with period at τ or 2τ (in $\mathcal{A}$).
2. There are only *finite* number of periodic orbits with period $n\tau$ $(2 \le n \le d)$.
3. There are a finite number of equilibria.

[a] i.e., with probability 1.
[b] That is, $x \in C^1$

This is the *fractal delay embedding theorem* by Sauer, Yorke, and Casdagli (1991) [11]. To understand this theorem, we need to clarify some terms.

Definition: An *embedding* is a map $\mathbf{F}$ of $\mathcal{A}$ manifold that is a C^1 *diffeomorphism* (one-to-one, onto, and invertible) for which the Jacobian $D\mathbf{F}$ has full rank everywhere in $\mathcal{A}$.

Schematic of an embedding is shown in Fig. 7.6. Please note that manifold $\mathcal{A}$ can be distorted by the embedding $\mathbf{F}$.

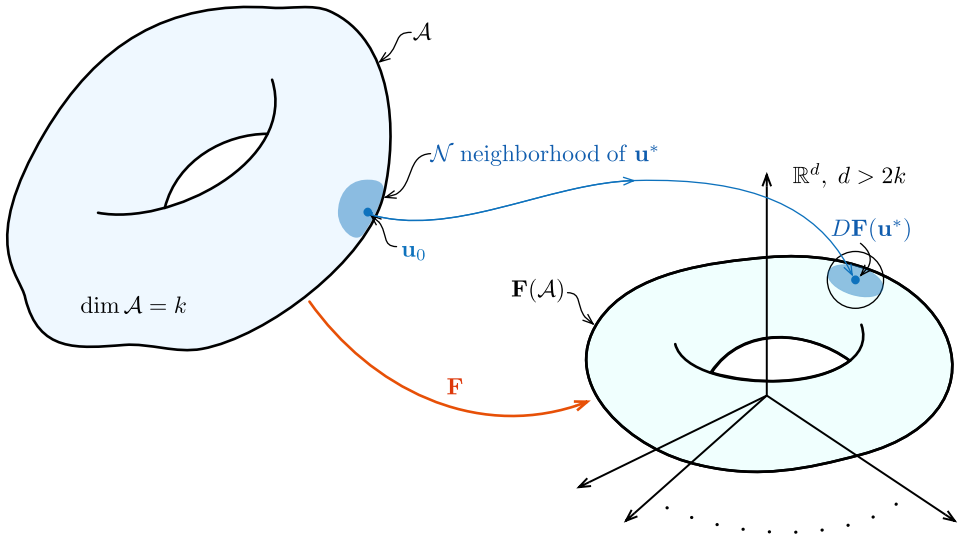

Figure 7.6: Schematic of an embedding

We still need to understand what the *capacity dimension* D_F is, which we will address in *Dimension Theory* chapter. However, right now we can consider a case when $D_F = D$ a "normal" integer dimension. Then the requirement that $d > 2D$ (or $d \geq 2D + 1$) can be understood with the cartoons shown in Fig. 7.7. In Fig. 7.7(a), when $m < D$, we clearly do not have large enough room for the embedding. In Fig. 7.7(b), when $m = D$, we see that the mapping cannot be one-to-one globally (though locally we are ok). In Fig. 7.7(c), when $m = 2D$, we might have an embedding $\mathbf{F}$ if we are lucky (top case), but generically we cannot guarantee one-to-one mapping due to expected intersections caused by projection (bottom case). In addition, there are infinite number of "nearby" $\mathbf{F}$ that fail too. Finally, in Fig. 7.7(d), when $d = 2D + 1$, we have an embedding that is geometrically ok and preserves the original trajectories dynamical characteristics. Cartoons in Fig. 7.7 are actually describing the *Whitney Embedding Theorem* [27], which the previously discussed *Embedding Theorem* generalizes to fractal dimensions.

7.1.1 Some Remarks

- In practice, we do not know D_F—that is one thing we may be trying to find. So the question is how to estimate d?
- The Delay-Embedding theorem says τ not important (except to avoid periodicities). In practice, this is not true: τ *too small* will collapse $\mathcal{A}$ onto hyper-diagonal of the embedding space and loose "attractor" in noise. With chaos, if τ is *too large*, successive $\mathbf{x}_n$ are approximately uncorrelated and the deterministic structure of the attractor may be obscured.

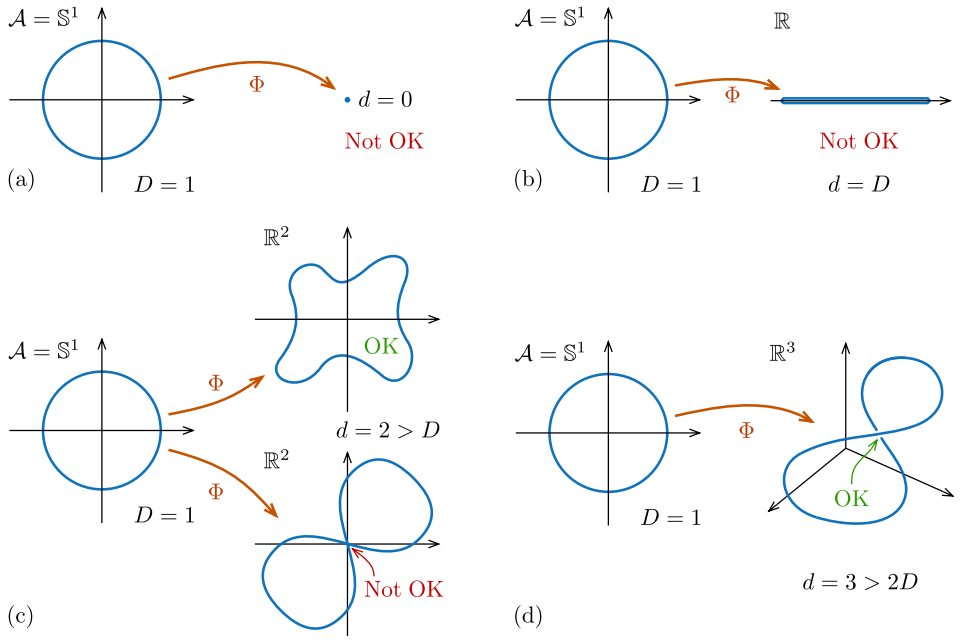

Figure 7.7: Cartoon illustration of the *Whitney Embedding Theorem*

- Embedding Theorem is only the *sufficient* condition ($d \geq 2D_F + 1$), but may not be necessary and we may achieve embeddings at lower dimensions.

In summary, if the conditions of the Embedding Theorem are satisfied, we have the commutator diagram shown in Fig. 7.8, where x is the measurement function and $\mathbf{e}$ is the delay-reconstruction procedure. Thus, the true dynamics can be expressed as:

$$\begin{aligned} \mathbf{u}(t) &= \varphi_t(\mathbf{u}_0) \\ &= \Phi^{-1} \circ \hat{\varphi}_t \circ \Phi(\mathbf{u}_0)\,, \end{aligned} \tag{7.12}$$

that is

$$\varphi_t = \Phi^{-1} \circ \hat{\varphi}_t \circ \Phi\,. \tag{7.13}$$

Therefore, in this sense, the dynamics in $\mathbb{R}^d$ and $\mathcal{A}$ are *equivalent*. Further, if we study, estimate, model $\hat{\varphi}_t$ on $\mathbb{R}^d$, we know *important properties* of φ_t on $\mathcal{A}$.

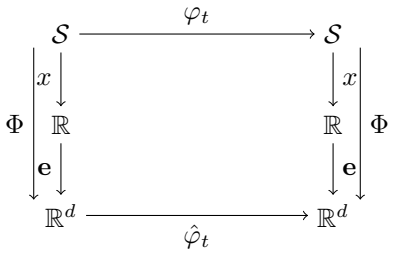

Figure 7.8: Commutator Diagram for the Delay Embedding

7.2 Phase Space Reconstruction

Two basic parameters need to be determined for delay coordinate embedding: delay time τ and the embedding dimension d. The delay time should be large enough that each coordinate of the reconstructed phase point is distinct and not redundant. Taking a *too small delay* results in delay vectors close to the hyper-diagonal of the reconstructed phase space and any variations transverse to it are not well defined. At the same time, for a chaotic response, the delay should *small enough* that coordinates are not statistically independent of each other and the attractor geometry is not complicated or obscured. For example, the ideal delay for a harmonic signal would be a quarter of its period. This suggests that the first zero crossing of the auto-covariance is a good choice for the delay (e.g., for a harmonic signal this happens at quarter of the period as shown in Fig. 7.9), but for chaotic time series this would generally overestimate the needed delay since it cannot account for nonlinear correlations.

Average mutual information (AMI), being a measure of nonlinear correlations, in the data provides another alternative. Its first minimum, when identifiable is usually a good estimate of optimal delay as it identifies a delay at which a new coordinate provides maximal new information compared to its delayed version while still being nonlinearly correlated to it. However, AMI estimates are very sensitive to noise, which usually makes it impossible to identify the first minimum in AMI data even if it exists. Another alternative is the use of *characteristic length spectrum* (CLS), which is highly robust to noise. The simplest of all methods is to just look at the reconstructed phase portrait and, when possible, identify the delay that provide a good cover of space without causing complex geometry.

The selection of the embedding dimension can be guided by the embedding theorem, which stipulated that any dimension higher than twice the capacity dimension of the attractor provides a good embedding. However, we do not usually know system's capacity dimension *a priori*, and need to

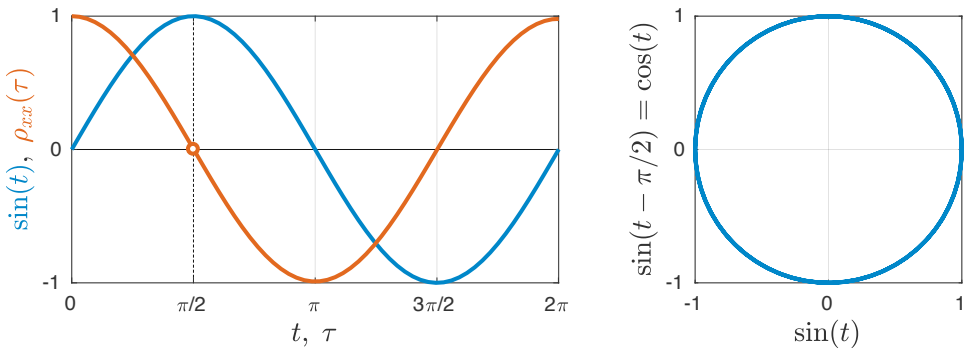

Figure 7.9: Sinusoid and its autocorrelation show that optimal delay should be $T/4$ since it provides for $\pi/2$ shift in phase.

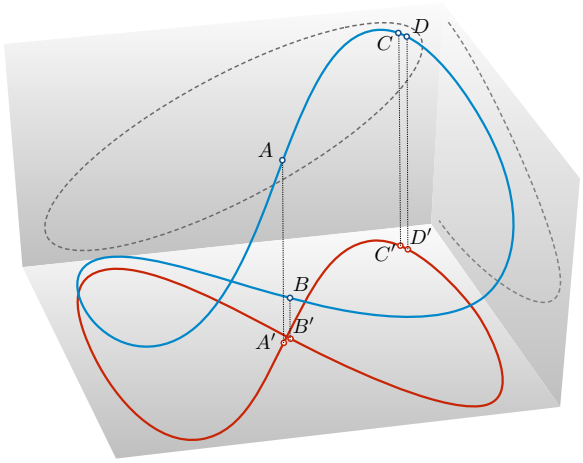

Figure 7.10: Basic idea of false nearest neighbors is illustrated by a three-dimensional closed curve (blue line) and its two-dimensional projection (red line). The thin dotted lines show two-dimensional projections onto other planes. The points that are false neighbors in the projection (A' and B') are far apart in higher dimensions (A and B), while the true neighbors (C' and D') remain close neighbors (C and D).

estimate it using the measured data and the corresponding reconstructed trajectory. In addition, this is only a *sufficient* condition and there may as well be perfectly good lower-dimensional embeddings. The *necessary* condition on the minimally needed embedding dimension is usually provided by the *false nearest neighbor* (FNN) method.

The idea behind the FNN method is simple (see Fig. 7.10): if two points in d dimensions are *true* nearest neighbors to each other, then adding $(d+1)$-th coordinate should not separate them from each other. If, however, they were nearest neighbors only due to the projection from a higher dimensional structure they will separate from each other in $(d+1)$-dimensional space and thus would constitute *false neighbors* in d dimensions. The minimum embedding dimension is identified by by the smallest dimension having zero fraction of the FNNs.

Problems

Problem 7.1

Consider the Hénon map [45],

$$x_{n+1} = a - x_n^2 + by_n \,, \quad y_{n+1} = x_n \,,$$

which yields chaotic solutions for $a = 1.4$ and $b = 0.3$.

1. Using a typical sequence of x_n, create different two dimensional phase portraits using delay times $\tau = 1, 2, \ldots$.
2. Which picture gives the clearest information about the original system? Why?
3. Rewrite the map in delay coordinates with unit delay and interpret the results.

Chapter 8

Selecting Delay Time for Phase Space Reconstruction

The estimation of optimal delay time can be approached from a purely geometrical perspective [56] or by considering linear/nonlinear (auto)correlations in the time series [31, 32]. As discussed in the previous section, there are primarily three methods for estimating the delay time using: (1) visual inspection of the reconstructed phase portrait (see Fig. 8.1 for the illustrations); (2) the first zero crossing of the autocorrelation function or correlation time τ_{corr} (refer to Fig. 7.9); and (3) the first minimum of the average mutual information. In some cases, selection of the optimal delay time and embedding dimension cannot be decoupled (e.g., when mutual information changes with the embedding dimension [38]). Some studies have advocated focusing on the so-called *embedding window* ($\tau_e = (d-1)\tau$) that describes the total time span covered by all coordinates in each of the reconstructed phase-space points [7, 35, 63, 78]. Others use model-based parameter estima-

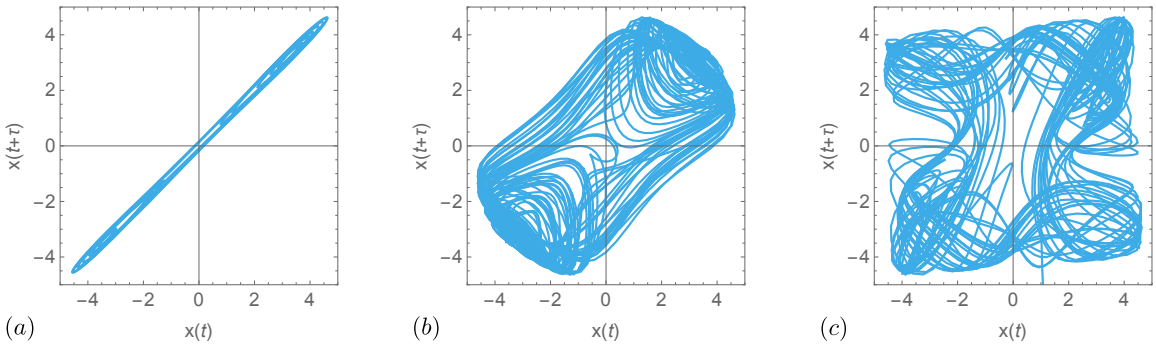

Figure 8.1: Determining delay "by eye:" if points are bunched long the diagonal, then τ is too small (a), if we see the dynamical structure and phase portrait looks "nice," then τ is about right (b); and if dynamical structure is obscured or "too complex," then τ is too long (c).

tion [9, 49, 60, 63, 84] to estimate both optimal delay and embedding dimension simultaneously. More recently, the optimal short-term prediction metric (based on artificial neural network model) was used for estimating both appropriate delays and embedding dimension [69]. A similar idea, in conjunction with average mutual information for the delay estimation, was used in estimating embedding dimension in Ref. [14]. Some of these methods are applicable in certain scenarios, others have some deficiencies, as discussed in Refs. [2, 38, 53]. Others advocate choosing the lag conservatively and then applying a spatial decorrelation transform [57] to allow for the even unfolding.

Before we discuss the nonlinear methodologies used in selecting delay time. We need to briefly review concept of *information entropy* addressed in *information theory*. Entropy as viewed from the *information theory* perspective is an average measure of information coming from a probabilistic or stochastic data source. This view is connected to but is not the same as thermodynamic entropy (a measure of disorder in a systems) in statistical mechanics.

8.1 Shannon Entropy

A measure of "disorder" or "uncertainty" in a system can also be called *entropy* from information perspective. Given a set $\mathcal{A}$ with partition of "boxes" $\{\mathcal{B}_i\}_{i=1}^N$, as shown in Fig. 8.2 by partitioning the x axis into equal-sized bins for one-dimensional data or as in Fig. 8.3 for a two-dimensional set, *Shannon Entropy* is defined as:

$$H(\mathcal{A}) = -\langle \log p_i \rangle = -\sum_{i=1}^{N} p_i \log p_i \,, \tag{8.1}$$

where $p_i = \text{Prob}\{x \in \mathcal{B}_i\}$, $\forall x \in \mathcal{A}$.[1]

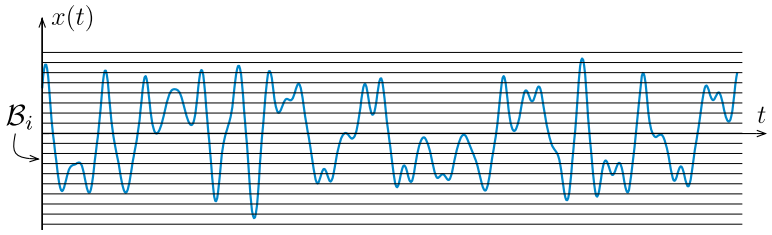

Figure 8.2: A time series can be partitioned by segmenting x axis into equal-sized bins $\mathcal{B}_i$

In a special case of "deterministic" dynamics on a partition, where $p_k = 1$ and, thus, $p_i = 0$ for all $i \neq k$ (remember $\sum_i p_i = 1$). In addition we also have (see Fig. 8.5)[2]

$$\lim_{p\to 0} p \ln p = \lim_{p\to 0} \frac{\ln p}{1/p} = \lim_{p\to 0} \frac{1/p}{-1/p^2} = \lim_{p\to 0} -p = \lim_{p\to 1} p \ln p = 0 \,. \tag{8.2}$$

[1] In terms of Shannon Information the partition is an "alphabet" and each box is a "letter" or "symbol."

[2] This result is obtained using l'Hospital's rule

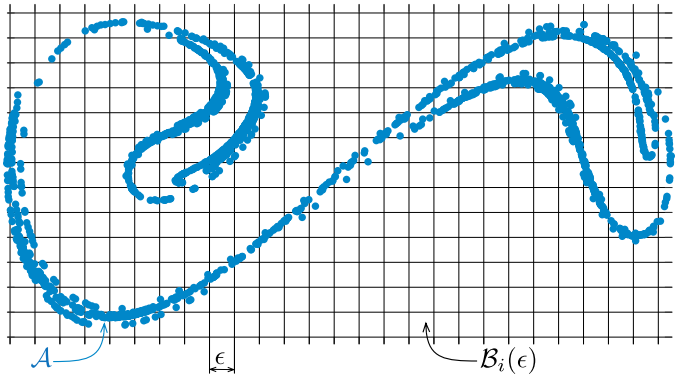

Figure 8.3: A set of points $\mathcal{A}$ in a Poincaré section covered by the boords of size ϵ.

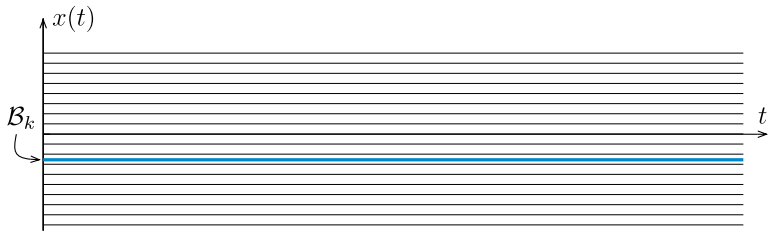

Figure 8.4: Deterministic dynamics on a partition $\mathcal{B}_i$, where $p_k = 1$

Therefore, for deterministic dynamics on a partition, $H(\mathcal{A}) = 0$. In other words, if the uncertainty in *state* is zero (only one box has $p = 1$), entropy is also zero. On the other hand, $\mathcal{H}$ achieves its maximum when $p_i = p_j = 1/N$ (please refer to Fig. 8.6 for a binary set example, when we only have two boxes). Thus, maximum entropy is associated with maximum uncertainty in state (any box is equally likely to be populated).

8.2 Mutual Information

The change in *entropy* of some process is usually called *information*. Thus, information of some process can be thought as $I = \Delta H$. Given a partition $\{\mathcal{B}_i\}_{i=1}^m$ on $\mathcal{A}$ of process x we denote the probability of each partition as:

$$p_i^{(x)} = \text{Prob}\{x \in \mathcal{B}_i\} = p_i^{(x)}(x) \tag{8.3}$$

and if we have another process y with partition $\{\mathcal{C}_i\}_{i=1}^n$,

$$p_i^{(y)} = \text{Prob}\{y \in \mathcal{C}_i\} = p_i^{(y)}(y) \tag{8.4}$$

Now we also define a *joint probability density*:

$$p_{ij}^{(xy)} = \text{Prob}\{x \in \mathcal{B}_i, y \in \mathcal{C}_j\} \triangleq p_{ij}^{(xy)}(x, y) \tag{8.5}$$

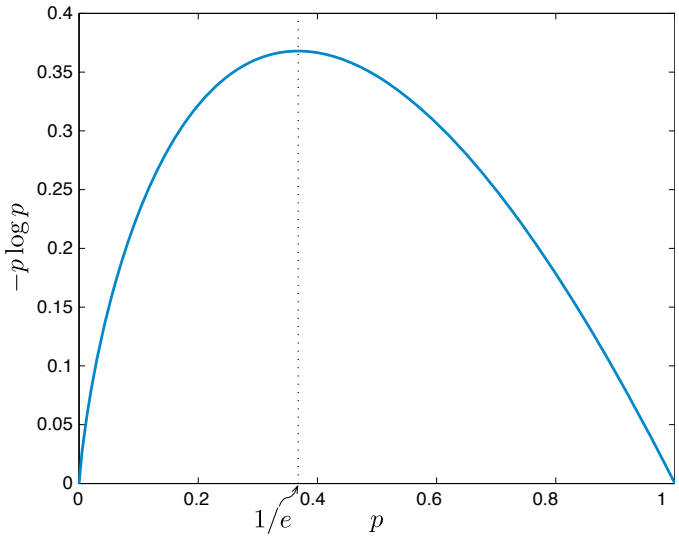

Figure 8.5: Plot of $-p\log p$ versus p

and the corresponding *joint entropy* is

$$H_{xy} = -\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{(xy)} \log p_{ij}^{(xy)} \tag{8.6}$$

If the two processes are independent, i.e.

$$p_{ij}^{(xy)} = p_i^{(x)} p_j^{(y)} \tag{8.7}$$

then

$$\begin{aligned} H_{xy} &= -\sum_{i=1}^{m}\sum_{j=1}^{n} \left(p_{ij}^{(xy)} \log p_i^{(x)} + p_{ij}^{(xy)} \log p_j^{(y)} \right) \\ &= -\sum_{i=1}^{m} p_i^{(x)} \log p_i^{(x)} - \sum_{j=1}^{n} p_j^{(y)} \log p_j^{(y)} \\ &= H_x + H_y \end{aligned} \tag{8.8}$$

where we have used the fact that $\sum_j p_{ij}^{(xy)} = p_i^{(x)}$. Therefore, Shannon entropy is additive for x and y independent. Now, we define the *mutual information* between x and y processes as

$$\begin{aligned} I_{xy} &= H_x + H_y - H_{xy} \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} \left(p_{ij}^{(xy)} \log p_{ij}^{(xy)} - p_{ij}^{(xy)} \log p_i^{(x)} - p_{ij}^{(xy)} \log p_j^{(y)} \right) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{(xy)} \left(\log p_{ij}^{(xy)} - \log p_i^{(x)} - \log p_j^{(y)} \right) , \end{aligned} \tag{8.9}$$

and

$$I_{xy} = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{(xy)} \log \frac{p_{ij}^{(xy)}}{p_i^{(x)} p_j^{(y)}} . \tag{8.10}$$

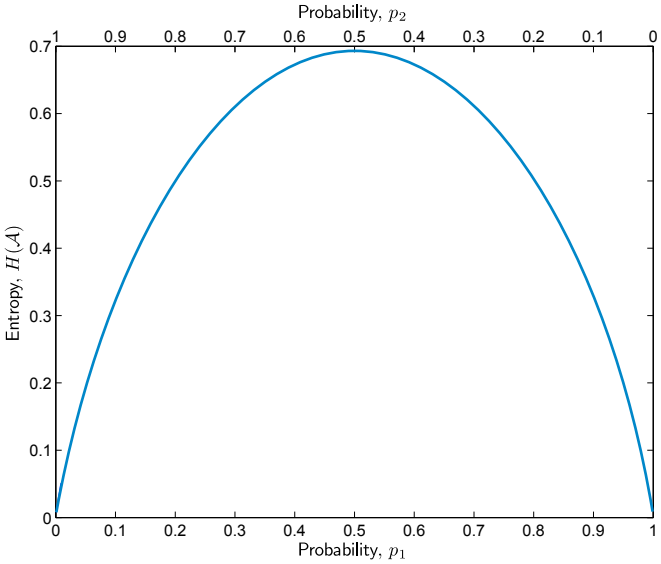

Figure 8.6: Plot of $H(\mathcal{A}) = -p_1 \log p_1 - p_2 \log p_2$ versus probabilities $(p_1 + p_2 = 1)$

Therefore, if $p_{ij}^{(xy)} = p_i^{(x)} p_j^{(y)}$, then $I_{xy} = 0$. That is, mutual information $I_{xy} = 0$ if x process is completely *independent* of y, which is a *much stronger than* x *and* y *being just* uncorrelated.

Now consider a case when $x = x(t)$ and $y = x(t+\tau)$. Then, for a stationary signal, $p_i^{(x)} = p_i^{(y)} \equiv p_i$ and $p_{ij}^{(xy)} = p_{ij}^{(x(t)x(t+\tau))} \equiv p_{ij}(\tau)$, so

$$I(\tau) = \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij}(\tau) \log \frac{p_{ij}(\tau)}{p_i p_j} \,. \tag{8.11}$$

When we use $\log_2$ in I equation, then it is given in *bits*. In addition, we can simplify further using the log property:

$$\begin{aligned} I(\tau) &= \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij}(\tau) \left(\log p_{ij}(\tau) - \log p_i - \log p_j \right) \,, \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij}(\tau) \log p_{ij}(\tau) - \sum_{i=1}^{N} p_i \log p_i - \sum_{j=1}^{N} p_j \log p_j \,, \end{aligned} \tag{8.12}$$

or

$$I(\tau) = \sum_{i=1}^{N} \sum_{j=1}^{N} p_{ij}(\tau) \log p_{ij}(\tau) - 2 \sum_{i=1}^{N} p_i \log p_i \,. \tag{8.13}$$

This is the form used in Ref. [53] for *average mutual information*.

Remarks

1. Ref. [53] claims that τ at which the first $\frac{\mathrm{d}I}{\mathrm{d}\tau} = 0$ is a "good" delay. For $I \geq 0$, $I = 0$ only if $x(t)$ and $x(t+\tau)$ are perfectly independent . Therefore, $x(t+\tau)$ adds maximal "information" since it is totally "new" data. Thus, when $I = 0$ we have zero redundancy in $x(t)$ and $x(t+\tau)$

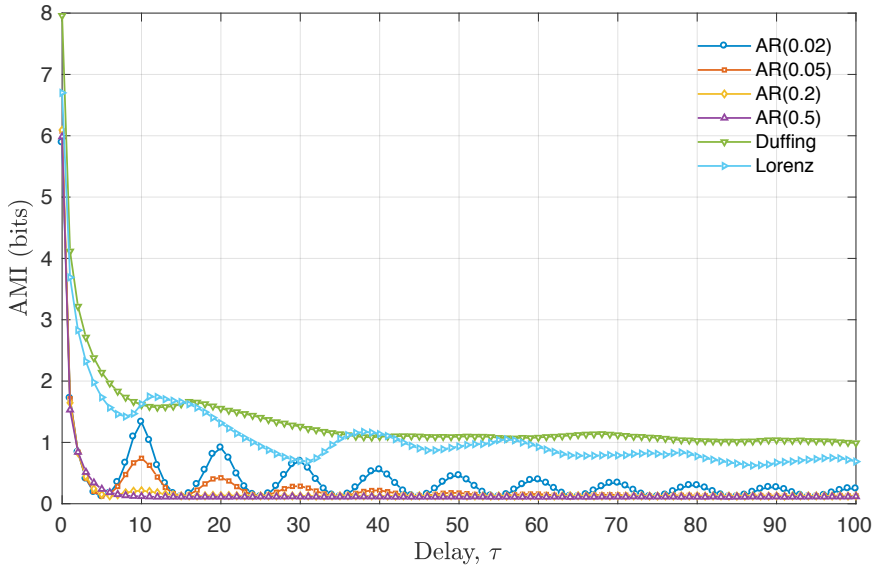

Figure 8.7: Average mutual information $I_{xx}(\tau)$ versus delay τ for a colored random processes described by $x_{n+1} = (2 - (\pi/10)^2 - \rho)x_n + (\rho - 1)x_{n-1} + \eta_n$, where η_n is white noise and $\rho = [0.02, 0.05, 0.2, 0.5]$. Each case is labeled as AR(ρ). In addition, autocorrelations of x time series from Duffing ($\ddot{x} + 0.2\,\dot{x} - x + x^3 = 0.33\cos t$) and y time series from Lorenz ($\dot{x} = -\frac{8}{3}x + yz$, $\dot{y} = -10\,(y - z)$, and $\dot{z} = -xy + 28y - z$) oscillators are plotted.

variables—they are mutually independent. So, when $\frac{\mathrm{d}I}{\mathrm{d}\tau} = 0$, information added is at a *local maximum* and, thus, redundancy is at a *local minimum*. For illustration purposes we have plotted in Fig. 8.7 average mutual information estimates for the time series shown in Fig. 3.2.

2. The minimum ($\frac{\mathrm{d}I}{\mathrm{d}\tau} = 0$) may not exist, or may be hard to determine due to additive noise or luck of sufficient data.

3. Remember that you cannot blindly trust the delay that the average mutual information gives. Always look at you data and remember that the *optimal* delay will depend on the application. For example, we can ask: what τ minimizes the variability in the estimate of some nonlinear characteristics such as Lyapunov exponents and fractal dimensions?

4. On the other hand, I_{xy} can provide evidence of nonlinear coupling between two process x and y that seem linearly uncorrelated. $I(\tau)$ could also indicate *possible* "dynamical structure" in a time series.

Some Properties of I_{xy}

1. $I_{xy} = H_x + H_y - H_{xy} = I_{yx}$, or x and y give equal information about each other.

2. Using the joint probability property $p(x,y) = p(x|y)p(y)$, we get

$$\begin{aligned} I_{xy} &= \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{(xy)} \log \frac{p_{ij}^{(xy)}}{p_i^{(x)} p_j^{(y)}} \,, \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{(xy)} \log \frac{p_{ij}^{(x|y)}}{p_i^{(x)}} = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{(xy)} \log \frac{p_{ij}^{(y|x)}}{p_i^{(y)}} \,, \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{(xy)} \log p_{ij}^{(x|y)} - \sum_{i=1}^{m} p_i^{(x)} \log p_i^{(x)} \,. \end{aligned} \tag{8.14}$$

Now, considering that

$$H_{x|y} = p_{ij}^{(xy)} \log p_{ij}^{(x|y)} \,, \tag{8.15}$$

we get that

$$I_{xy} = H_x - H_{x|y} = H_y - H_{y|x} \,, \tag{8.16}$$

where the first equality reflects the uncertainty in x and the second the uncertainty in y.

Problems

Problem 8.1

Consider the y time series from the Lorenz oscillator:

$$\dot{x} = -\frac{8}{3}x + yz, \quad \dot{y} = -10\,(y - z)\,, \quad \text{and} \quad \dot{z} = -xy + 28y - z\,,$$

staring from $x(0) = 20$, $y(0) = 5$, and $z(0) = -5$ initial condition and with 0.02 sampling time. Estimate and plot the corresponding average mutual information and autocorrelation coefficient versus the delay time. Estimate what should be the optimal delay time using both metrics. Now plot the phase portraits of using various delays and identify the one that most resembles the original phase portrait x versus y. Is there any difference between this reconstruction and the optimal delay reconstructions? What happens if we increase the delay further? Is the first zero crossing of the autocorrelation a good estimate for the delay?

Chapter 9

Selecting Embedding Dimension for Phase Space Reconstruction

Here, we focus on one of the popular methods used for estimating the minimally nessesary embedding dimension called *false nearest neighbors* (FNN) [58]. There are other methods that utilize convergence in the estimates of some nonlinear measures or metrics [8], but they are either more complex or cumbersome in the applications. The idea behind the FNN [58] method is based on the uniqueness property of a dynamical system's phase-space trajectory as discussed at the beginning of this chapter: the *nearest neighbor* (NN) of a point in a d-dimensional embedding is false if the pair of points are close only due to the projection from a higher-dimensional (e.g., $(d+1)$-dimensional) phase space. Thus, the FNN will separate if the data is embedded into a $(d+1)$-dimensional space, while the *true* NNs will remain close. If one is able to detect all the FNN, then the minimally nessesary embedding dimension can be identified as the least dimension needed to achieve zero fraction of the FNN.

There are three distinct variations of the FNN algorithm. We label them as: the *original* or *Kennel* [58], *improved* [43], and *reliable* [57] algorithms. Variations of them are frequently used in very diverse fields such as stock market price forecasting [55], astrophysics [42], engineering [94], and biology [41]. The effect of additive noise in deterministic time series on the performance of these algorithms will also be discussed.

It has become customary, in the nonlinear time series analysis, to compare the results one gets from applying an algorithm to the data to the results obtained using *surrogate data*, which possesses the same linear characteristics (i.e., power spectrum and probability density) as the original data. There are many different methods to obtain a surrogate data [82], the simplest are only preserving either probability density (by randomly reshuffling data points time sequence) or power spectrum (by randomizing the phase of the Fourier transform and then converting it back to the time domain).

The idea is that one cannot trust the results if they cannot be differentiated from the ones obtained from the surrogate data. The estimating of minimally necessary embedding dimension requires the surrogate analysis to make we are not using algorithm that would yield low-dimensional indication for the noise. Instead of going through an exercise of frequency and time domain randomization and normalization procedures [82], we will follow [57], where they used a much simpler strategy. For each d-dimensional NN pair, the $(d+1)$-dimensional coordinate distance is estimated by randomly selecting one of the coordinates from all the data. This is justifiable, if all the linear correlations are absent from the components of the reconstructed phase space (the needed decorrelation coordinate transformation is also discussed in Section 9.3.1), where the noise would appear to be effectively white. The algorithms described in the following sections will be contrasted and compared to each other using the normally and uniformly distributed white noise time series as well as the ones used in Figs. 3.2 and 8.7.

9.1 The Original False Nearest Neighbor Algorithm

The original FNN method was proposed by Kennel *et al.* in 1992 [58]. Working in each d-dimensional reconstructed phase space, for each point $\mathbf{x}_i$ its NN $\mathbf{x}_{j(i)}$ is *identified as false* if:

$$\left|x_{i+d\tau} - x_{j(i)+d\tau}\right| > r \left\|\mathbf{x}_i - \mathbf{x}_{j(i)}\right\| , \tag{9.1}$$

where r is an *a priori* fixed threshold value (typically chosen to be 10 in Ref. [58]).

To simplify notation and streamline discussion, we define the following variables:

$$\delta_i \triangleq \left|x_{i+d\tau} - x_{j(i)+d\tau}\right| , \quad \text{and} \quad \epsilon_i \triangleq \left\|\mathbf{x}_i - \mathbf{x}_{j(i)}\right\| . \tag{9.2}$$

In what follows, we will also imply the functional dependence of index j on the index i (i.e., $\mathbf{x}_j$ is always the NN to $\mathbf{x}_i$ using the appropriate NN metric). Then, the probability that any point $\mathbf{x}_i$ and its NN $\mathbf{x}_j$ are FNN can be written as:

$$f_{nn}(d; r) \triangleq \text{Prob}\left\{\delta_i > r\, \epsilon_i\right\} , \tag{9.3}$$

and we refer to it as the *original fraction*. Then, the idea is that the smallest dimension d for which $f_{nn}(d; r)$ reaches zero is the minimally *sufficient* embedding dimension. The original fraction provides misleading results for the white random and correlated stochastic time series for which it erroneously shows low embedding dimensions and the surrogate analysis does not provide the differentiation [57]. In fact, results for uncorrelated random time series and for large amplitude autoregresive processes are virtually identical and indicate the existence of a low-dimensional attractor. This is a problem for noisy deterministic time series, since we cannot definitively attribute zero FNN to the actual existence of a low-dimensional attractor.

Hegger and Kantz [43] were first to clearly identify the shortcomings of this definition when noisy deterministic data were considered. For uniformly distributed white noise, the original fraction $f_{nn}(d;r) = 1 - 2\langle\epsilon_i\rangle\, r + \langle\epsilon_i^2\rangle\, r^2$, where $\epsilon_i = \epsilon_i(d)$ is the distance between the NNs in d dimensions. For uncorrelated noise time series of size N, $\langle\epsilon_i\rangle \simeq N^{-1/d}$. Therefore, this FNN fraction approaches zero when $\langle\epsilon_i\rangle \simeq r^{-1}$ or when $N \simeq r^d$. To always differentiate noise (which is inherently infinite-dimensional) from deterministic data (finite-dimensional), one always needs $N \geq r^d$ to get the nonzero original fraction for uncorrelated noise time series. Unfortunately, this is not always feasible for experimental data.

To address this problem, in [58], the NNs were also counted as false if the distance in $(d+1)$-dimensions was greater than the a times σ_x of the data (e.g., $a = 2$ in [58]). Thus, the FNN condition changed to:

$$\hat{f}_{nn}(d;r) \triangleq \text{Prob}\left\{(\delta_i > r\,\epsilon_i) \;\vee\; \left(\delta_i^2 + \epsilon_i^2 > a^2\sigma_x^2\right)\right\}, \tag{9.4}$$

which we call the *Kennel fraction*.

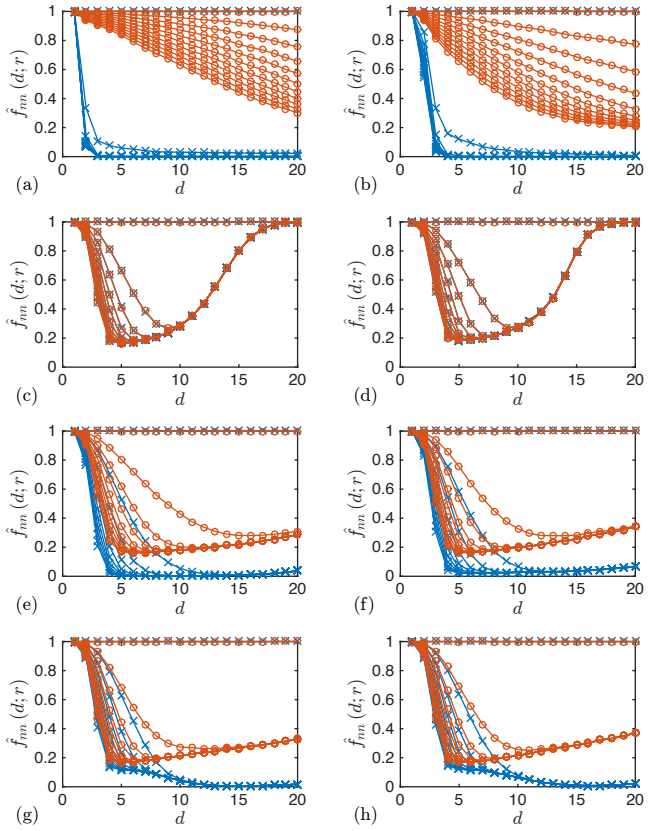

Figure 9.1: Kennel fraction of FNN using Eq. (9.4) for Lorenz (a), Duffing (b), Normal (c), Uniform (d), AR(0.02) (e), AR(0.05) (f), AR(0.2) (g), and AR(0.5) (h). The gradual decrease in the FNN fraction is precipitated by incrementing r from zero to 20 by two. Lines with circles reflect the corresponding surrogate data results.

Figure 9.1 shows the results of this algorithm applied to our data sets. This solution seems to improve performance for the white random time series by increasing the FNN fraction for higher embedding dimensions. However, it exhibits a dip in the FNN fraction at lower dimensions (i.e., it underestimates the FNN fraction for large r and low d). This dip is also present for the correlated stochastic time series especially for ones with the larger stochastic components. However, the surrogates also show a similar dip and diverge from actual data after the minimum is reached. Even with this divergence the correlated stochastic data is hard to interpret and this solution is inadequate [57]. This will also be problematic for noisy deterministic time series, since this dip cannot be uniquely attributed to the existence of a low-dimensional attractor. In addition, [58] points out that this solution may introduce false neighbors in large d for an insufficient amount of deterministic data.

9.2 The Improved Algorithm

In [43], it is supposed that if the distance between the NNs is larger than σ_x, these points cannot be considered true NNs and should be discarded in computation. In this approach, the NNs for which the d-dimensional distances are greater or equal to $1/r$ times σ_x are disregarded and the original fraction is used for FNN. Unfortunately, for large values of d and r, this causes the number of available points to decrease drastically to zero for both white random and correlated stochastic data sets. The current algorithm for false NNs, as described in TISEAN package [44, 53] has the following form:

$$\tilde{f}_{nn}(d;r) \triangleq \frac{\sum_{i=1}^{N-(d-1)\tau} \Theta\left(\delta_i - r\epsilon_i\right) \Theta\left(\sigma_x - r\epsilon_i\right)}{\sum_{i=1}^{N-(d-1)\tau} \Theta\left(\sigma_x - r\epsilon_i\right)}, \tag{9.5}$$

where Θ is the Heavyside function and $\tilde{f}_{nn}$ is referred to as the *TISEAN fraction*. They claim that this solution for white random data leads to 50–60 % FNN for large N, independent of choice of d and r. However, in our white random and dominantly stochastic test cases, the number of points suitable for averaging drops to zero precipitously for large d values. In addition, it indicates low-dimensional embedding for stochastic or random data and does not provide any differentiation with the surrogate analysis.

9.3 The Reliable Algorithm

In [57], some of the ideas in [43] were further expanded and a new FNN algorithm was proposed in conjunction with new and interesting *false nearest strands* (FNS) idea. These new FNN and FNS methods were shown to be usable for noisy or stochastic time series. In [57] several important improvements to the FNN algorithm were proposed: (1) only NNs that are *temporarily uncorrelated* are considered to focus exclusively on spatial proximity; (2) instead of FNN, FNS were advocated to

make sure only true NNs were considered in the calculations even for noisy data; and (3) a spatial decorrelation transform was introduced to deal with linear correlation bias due to small delays.

9.3.1 Spatial Decorrelation

The spatial decorrelation step is aimed at negating the effect of the too-small delay used in the reconstruction, which makes the FNN fraction small irrespective of the actual dynamics. In addition, this step makes the method insensitive to the appropriate choice of the time delay, unless it is chosen to be unreasonably large.

The basic idea of the decorrelation is to counteract the bunching of all reconstructed points near a thin tube along the hyperdiagonal of a phase space. Given the original $(d+1)$-dimensional reconstructed phase-space trajectory $Y \in \mathbb{R}^{(d+1)\times[N-\tau(d+1)]}$, where the i-th column is given by the i-th delay vector $\mathbf{x}_i \in \mathbb{R}^{d+1}$, Y is first transformed into its proper orthogonal space $\tilde{Y}$ using *singular value decomposition* [36]:

$$Y = UCV^T, \quad \Rightarrow \quad \tilde{Y} = U^T Y\,, \tag{9.6}$$

Then $\tilde{Y}$ is rotated in the phase space so that the first d-coordinates are *only* a function of the first d coordinates of the original phase space Y. To accomplish this we take the last column of $\tilde{Y}$, $\mathbf{u} \in \mathbb{R}^{d+1}$, and then the needed orthogonal transformation is given by the following Householder matrix [36]:

$$Q = H\left(\mathbf{u} - |\mathbf{u}|\,\mathbf{e}_{d+1}\right)\,, \tag{9.7}$$

where $\mathbf{e}_{d+1}$ is the unit vector in the $(d+1)$ direction. Then the decorrelated phase space is given by:

$$\bar{Y} = Q\tilde{Y}\,. \tag{9.8}$$

In our test samples, the choice of delay time is close to optimal and the use of decorrelation does not provide any significant benefits, but in practice it can be a differentiating factor. Thus, we have included this step in the results presented from now on.

9.3.2 Temporal Decorrelation

As opposed to the TISEAN fraction, Ref. [57] advocates to look at just the distance between the $(d+1)$-th coordinates of the NNs. If we only consider the FNN- and *not* the FNS-based metric we will get:

$$\bar{f}_{nn}(d;s) \triangleq \text{Prob}\left\{\delta_i > s\,\sigma_x\right\}\,, \tag{9.9}$$

where, to identify true NNs for each point $\mathbf{x}_i$, its NN $\mathbf{x}_j$ is determined such that:

$$j = \underset{j}{\text{argmin}}\,\|\mathbf{x}_i - \mathbf{x}_j\|\,, \text{ subject to } |i-j| > w\,, \tag{9.10}$$

where w is a Theiler window [88, 89] used to remove temporal correlations. We call the metric defined by Eq. (9.9) the *reliable fraction*. Ref. [57] discusses the appropriate choice of the Theiler window and advocates to use two or three times the first minimum of average mutual information as a good rule of thumb. Another alternative is to match the w time scale to the point at which the average mutual information reaches its asymptotic zero value.

The results for Eqs. (9.9), and (9.10) using a Theiler window of four times the first minimum of the average mutual information are shown in Fig. 9.2 (left plots). For the deterministic time series, we observe an initial underestimation of the FNN fraction with the increase in s value. The fractions reach zero for an appropriate embedding dimension and then show gradual increase for small s values with the increase in d values. This gradual increase is more pronounced for the Duffing data and is absent for larger s values. For the white random data sets the FNN fraction behaves as expected, showing linear decrease as s is increased, while remaining constant for all values of d. For the correlated stochastic data this metric is more interesting and shows trends intermediate between the trends for the purely white random and deterministic data. Specifically, it can clearly be used with large s values to differentiate small stochastic component from the larger deterministic data—plots (e,f). The results for the surrogates are clearly separated from the actual data, especially for larger s values, and remain constant for all d.

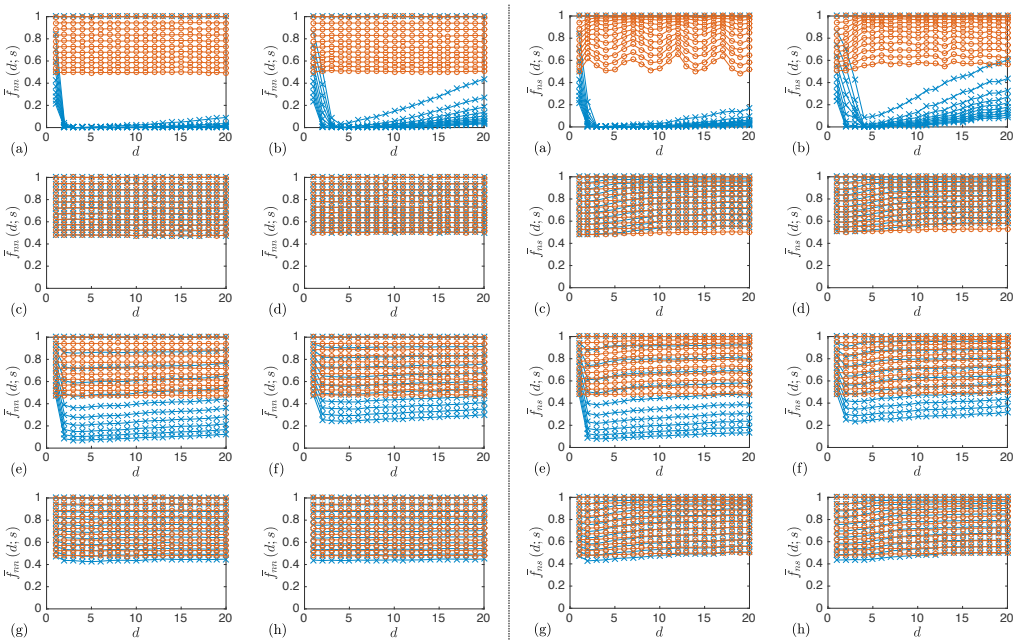

Figure 9.2: Reliable FNN (left plots) and reliable FNS (right plots) fraction of Eq. (9.9) applied to Lorenz (a), Duffing (b), Normal (c), Uniform (d), AR(0.02) (e), AR(0.05) (f), AR(0.2) (g), and AR(0.5) (h). The gradual decrease in the FNN fraction is precipitated by incrementing s from zero to two by 0.1. Lines with circles reflect the corresponding surrogate data results.

9.3.3 False Nearest Strands

In the same paper [57], the shortcomings of the previous FNN methods for the deterministic data contaminated by small random noise are also identified. In this case, the NNs identified through the analysis may be proximal not due to the deterministic dynamics or the projections, but due to noise. If the proportion of the misidentified NNs is large, then the algorithm will overestimate the corresponding FNN fraction. To identify only true (in terms of deterministic dynamics and projection only) NNs, they propose to consider *NN strands* (NNS). The following procedure is used to identify NNS:

1. Identify NN $\mathbf{x}_{j(i)}$ for each point $\mathbf{x}_i$ given by Eq. (9.10).

2. If there is any pair of points (for a minimal $k = k^\star$) from a set of preceding NNs $\{\mathbf{x}_{i-k}, \mathbf{x}_{j(i-k)}\}_{k=1}^{w}$ such that $j(i) = k^\star + j(i-k^\star)$, then $[\mathbf{x}_i, \mathbf{x}_{j(i)}]$ are assigned to the strand containing $[\mathbf{x}_{i-k^\star}, \mathbf{x}_{j(i-k^\star)}]$. Otherwise, $[\mathbf{x}_i, \mathbf{x}_{j(i)}]$ are assigned to a new pair of NN strands.

3. After examining all the data, N_s number of NNS are obtained, each containing a set of NN points S_k $(k = 1, \ldots, N_s)$. Then the k-th strand is designated false if $\langle \delta_i \rangle_k > s\,\sigma_x$, where

$$\langle \delta_i \rangle_k \triangleq \langle \delta_i \rangle_{i \in S_k} = \frac{1}{\|S_k\|} \sum_{i \in S_k} \delta_i \,. \tag{9.11}$$

   Alternatively we can define *Reliable FNS Fraction* as:

$$\bar{f}_{ns}(d; s) \triangleq \text{Prob}\left\{ \langle \delta_i \rangle_k > s\,\sigma_x \right\} \,. \tag{9.12}$$

The results of the FNS calculations for our test time series are shown in Fig. 9.2 (right plots). The decrease in the FNS fraction with the increase in s is still present as reliable FNN. For the deterministic data, some gradual increase in this fraction with the increase in d for small values of s is also present. In addition, and in contrast with the reliable FNN fraction, this metric does not converge to zero as fast and indicates larger minimal embedding dimension for the deterministic data. The results for the white random data are still indistinguishable from the ones for the surrogate data. However, the white random data trends show a gradual increase in the FNS fraction not observed reliable FNN. For the correlated stochastic data sets, FNS trends are clearly delineated from the surrogates for the low level of stochasticity, and show results similar to the white random data for the larger stochastic components. To accurately estimate embedding dimension in deterministic cases, one needs to set some threshold value for the FNS fraction since it does not reach zero as fast as in the reliable FNN fraction. This task is made easier by the clear separation from the surrogates FNS fraction, which stays relatively level with the increase in d. For the larger s values, the Duffing FNS fraction also indicates lower than expected embedding dimension.

To see the advantage of these new metrics when dealing with noisy data, we apply them to Duffing data with additive Gaussian noise as shown in Fig. 9.3. For the reliable FNN algorithm, the

new fraction floor rises with the noise level, but it is still usable for small values of s and low 5 % noise level Fig. 9.3(a). At the same time, the surrogate's FNN fraction stays constant with respect

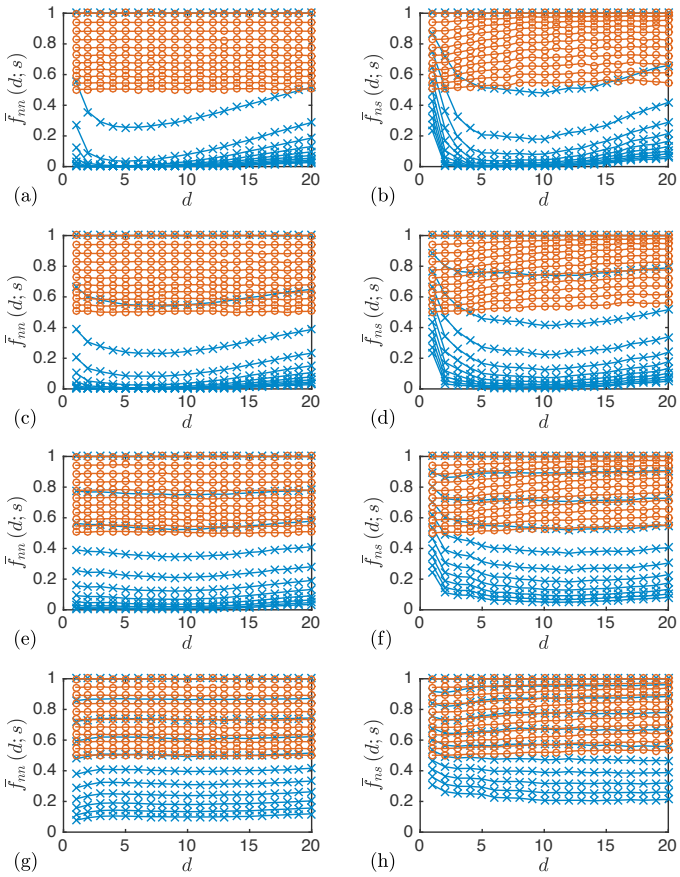

Figure 9.3: Reliable FNN (first column: a,c,e,g) and FNS (second column: b,d,f,h) algorithm applied to Duffing data with 5 % (a,b), 10 % (c,d), 20 % (e,f), and 40 % (g,h) noise. The gradual decrease in the fractions is precipitated by incrementing s from zero to two by 0.1. Lines with circles reflect the corresponding surrogate data results.

to d and asymptotes down with the increase in s to approximately 50 % fraction. The FNS data looks better for the low noise levels Fig. 9.3(b,d) and exhibits similar surrogate behavior. Therefore, it provides better separation between the surrogate and actual data FNS fractions compared to the FNN fractions. One might be able to estimate embedding dimension for the low noise levels in Fig. 9.3 if an appropriate threshold value is set. However, both reliable fractions cannot identify the embedding dimension for the larger noise levels.

In what follows, we want to further expand on some ideas in [57] and propose an FNN algorithm that is usable for noisy or stochastic time series with deterministic components. We begin with exploring the nearest-neighbor statistics for deterministic, white random, and correlated stochastic time series.

9.4 Nearest-Neighbor Statistics

To investigate the problems outlined in the previous sections, we investigate NN statistics for our test time series. First, consider the expected average nearest-neighbor distance $\langle \epsilon_i(d) \rangle$ for white random time series. Given a d-dimensional reconstruction of normally distributed random time series, the distance between any two points will be distributed according to a χ_d (d-dimensional chi distribution [39]). We also make the proposition that the expected value of the distance between the NN points will be proportional to the mean of the corresponding χ_d distribution and to the average density of points in d dimensions $(\sqrt{d}/N)^{1/d}$. This will result in the following approximation:

$$\langle \epsilon_i(d) \rangle \simeq \sqrt{2} \frac{\Gamma\left((d+1)/2\right)}{\Gamma\left(d/2\right)} \left(\frac{\sqrt{d}}{N} \right)^{1/d} , \tag{9.13}$$

where Γ is the gamma function. Figure 9.4(a) shows the expected values of nearest-neighbor distances versus the embedding dimension for all eight test time series, without accounting for temporarily correlated NNs. The increase between the mean nearest-neighbor distance is more drastic for the white random data than the deterministic time series, and Eq. (9.13) follows this increase for the white random time series very closely.

The deterministic time series show more power-law type increase after the needed embedding dimension is reached ($d \sim 3$–4, in both case), and it has a considerably lower rate than for the random data. The nearest-neighbor distance for deterministic data is expected to grow according to local divergence rate $e^{\lambda_{\max} d \tau t_s}$, where $\lambda_{\max}$ is maximal Lyapunov exponent, after the minimum embedding dimension is reached. This can be expressed as:

$$\langle \epsilon_i \rangle \approx \varepsilon \left(e^{\lambda_{\max} d \tau t_s} - 1 \right) , \quad \forall d \geq m , \tag{9.14}$$

where ε is a constant reflecting the point density in the phase space. The parameters that follow the observed curves closely are: $\varepsilon = 0.0404 \pm 0.0008$ and $\lambda_{\max} \tau t_s = 0.0689 \pm 0.0008$ for the Lorenz

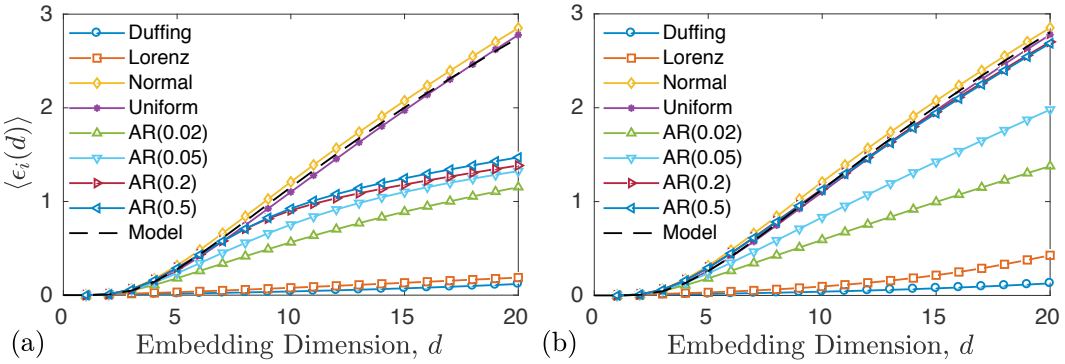

Figure 9.4: Expected value of nearest-neighbor distances in d dimensions without (a) and with (b) temporarily decorrelated NNs

data, and $\varepsilon = 0.5110 \pm 0.0742$ and $\lambda_{\max}\tau t_s = 0.0167 \pm 0.0020$ for the Duffing data. The exponential growth in Eq. (9.14) is only expected until it reaches the attractor size at which point it will saturate. Even if it is allowed to grow unimpeded it will only cross Eq. (9.13) somewhere $d \sim 60$. Thus, when dealing with low-dimensional deterministic systems (e.g., $d < 40$) their expected nearest-neighbor distance will stay well below the distance expected for the white random data in the same dimension.

The correlated stochastic time series show behavior somewhere in the middle of the white random and deterministic nearest-neighbor distance trends. They start close to the white random at low dimensions and then diverge to smaller distances for the larger embedding dimensions.

The same plot was regenerated for the distance between temporarily decorrelated NNs as shown in Fig. 9.4(b). The main observed differences are that the large stochastic component data now closely follows the white random distances, while the small stochastic component data shows the intermediate behavior. In addition, the deterministic data have more pronounced trends clearly showing differences in the divergence rates. Fitting the Eq. (9.14) to these curves gives the following parameters: $\varepsilon = 0.0333 \pm 0.0007$ and $\lambda_{\max}\tau t_s = 0.0790 \pm 0.0008$ for the Lorenz data, and $\varepsilon = 0.0425 \pm 0.0020$ and $\lambda_{\max}\tau t_s = 0.1212 \pm 0.0021$ for the Duffing data. Thus, we observe: (1) temporal correlations considerably stint the divergence rates, especially for the Duffing data; and (2) divergence exponents after removing temporal correlations for Lorenz ($\lambda_{\max} \approx 0.5$) and Duffing ($\lambda_{\max} \approx 0.1$), while not being suitable for estimating maximal Lyapunov exponents, are close to their expected values [86] (experimentally estimated largest Lyapunov exponents from data are $\lambda_1 \approx 0.825$ for Lorenz and $\lambda_1 \approx 0.134$ for Duffing).

Now, let us consider the $(d+1)$-th coordinates of the NNs in d dimensions, which are still uncorrelated random numbers for the white random time series. Thus, the distance between these $(d+1)$-th coordinates for normally distributed variables ($x_i \sim \mathcal{N}[0, \sigma_x]$) follows a chi distribution [39][1], and the corresponding expected value can be expressed as:

$$\langle \delta_i \rangle = \mathbb{E}[\delta_i] = \frac{2\,\Gamma(1)}{\Gamma(1/2)}\sigma_x \approx 1.1284\,\sigma_x\,. \tag{9.15}$$

Now, it is easy to show that for the uniformly distributed variables (i.e., $x_i \sim \mathcal{U}[-a, b]$), this expected value becomes:

$$\langle \delta_i \rangle = \frac{2}{\sqrt{3}}\sigma_x \approx 1.1547\,\sigma_x\,. \tag{9.16}$$

We conclude that for some general uncorrelated random time series the average distance between the $(d+1)$-th coordinates is expected to be slightly greater than the corresponding σ_x (~ 1.12–$1.16\,\sigma_x$).

A similar relationship is expected for the colored-noise time series if the delay used in the reconstruction is greater than the correlation time. However, for a deterministic time series we expect this $(d+1)$ component's distance to be small for true NNs. The estimation results using our synthetic

[1] If $x_i \sim \mathcal{N}[0, \sigma_x]$, then $x_i - x_j \sim \mathcal{N}[0, 2\,\sigma_x]$, and $\frac{|x_i - x_j|}{2\,\sigma_x} \sim \chi_1$.

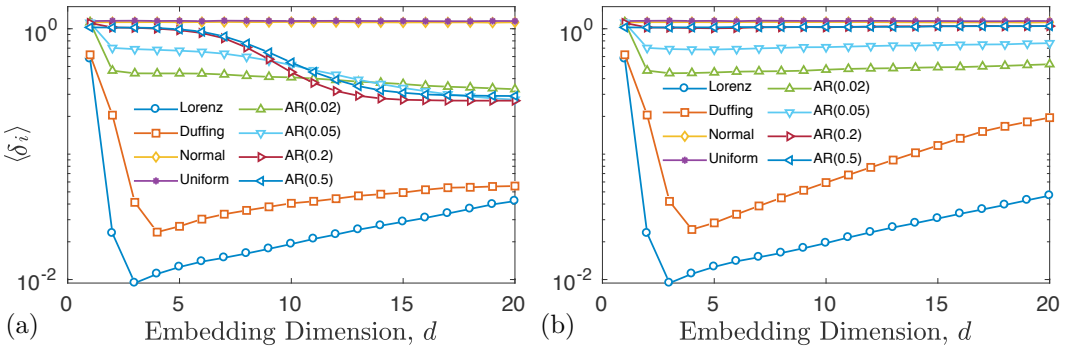

Figure 9.5: Expected value of distances between the $(d+1)$-th coordinates of the NNs in d dimensions without (a) and with (b) temporarily decorrelated NNs

time series without removing temporal correlations are shown in Fig. 9.5(a), where we plot the mean value of this distance at each embedding dimension. As we can see, the analytical estimates are very close to the numerical results for the white random data sets, which are clearly larger than the ones for the deterministic time series.

For the correlated stochastic data we have an initial rapid drop in the mean $(d+1)$ components' distance followed by a slower gradual decrease. The initial drop is due to the deterministic correlations in the data caused by small delay and is more pronounced for weakly stochastic sets. The consequent slow decrease is due to the longer temporal correlations in the data, and it is more drastic for the strongly stochastic sets. All the correlated stochastic trends seem to level our at about 30 % of σ_x. The $(d+1)$-th distances for the deterministic data start near $0.5 \sim 0.6$ FNN fraction and drop down rapidly to near zero at the true minimal embedding dimension and then have slow gradual increase Fig. 9.5(a). This gradual increase is caused by the maximal local divergence rate of the nearby trajectories, and is expected to follow $e^{\lambda_{\max} d\tau t_s}$ similar to Eq. (9.14).

The same plots were regenerated for the temporarily decorrelated nearest-neighbor statistics as shown in Fig. 9.5(b). The decorrelation had big effect on the correlated stochastic data: they were pushed up to the white random trends for the correlated data with larger stochastic components. In contrast, the AR(0.02) and AR(0.05) cases show trends more similar to the deterministic data, by exhibiting initial drops followed by the very slow increase. All the stochastic data stays above $0.45\sigma_x$ level and can be clearly differentiated from the deterministic data. The deterministic trends did not change during the initial drop to zero. However, the following gradual increase has a more pronounced character, especially for the Duffing data. This increase is more in line of expected exponential divergence caused by the maximal Lyapunov exponent, and was not as pronounced before due to temporarily correlated components staying close irrespective of embedding dimension.

The statistical analysis described in this section can be summarized as:

1. Temporal correlations significantly alter the observed results and need to be accounted for

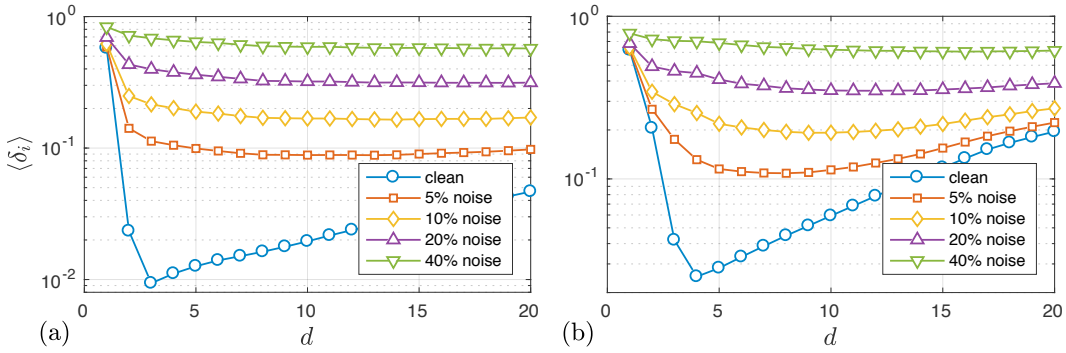

Figure 9.6: Expected NN distance $\langle\delta_i\rangle$ versus the embedding dimension d for noisy Lorenz (a) and Duffing (b) time-series

in accurate FNN analysis (removal of temporarily correlated points from the FNN analysis increases the differentiating power of the NN statistics).

2. Of the considered temporarily decorrelated metrics, $(d+1)$-th coordinate distance (δ_i)—as shown in shown in Fig. 9.5(b)—provides the best indication of the minimal embedding dimension and superior separation between the deterministic and random/stochastic trends, which remain constant with the increase in d. Thus, this metric can be used directly for estimating the embedding dimension for the noise free deterministic data. However, for noise contaminated deterministic data, it loses its ability to clearly identify the minimal embedding dimension as shown in Fig. 9.6.

3. Both $\langle\epsilon_i\rangle$ and $\langle\delta_i\rangle$ for deterministic data show exponential growth after minimal embedding dimension is reached. Thus, the original FNN fraction shown in Fig. 9.7 for both without (a) and with (b) temporarily decorrelated NNs is a good differentiator of deterministic data for low embedding dimensions. Setting a threshold value near $5 \sim 10$, this ratio with tem-

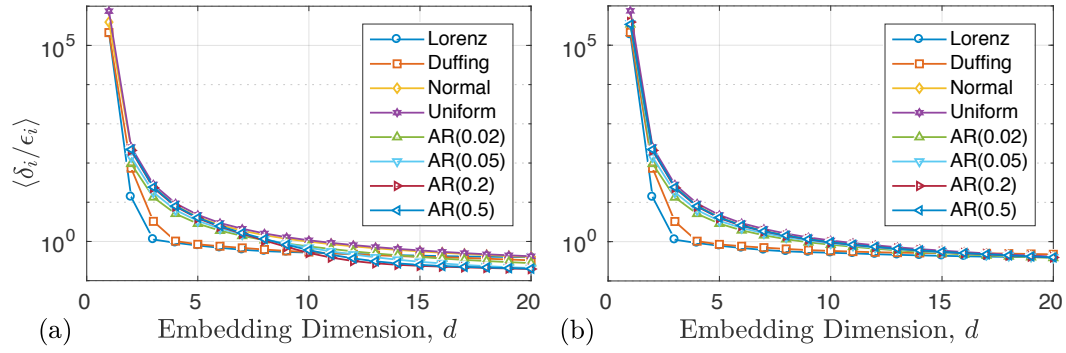

Figure 9.7: Expected value of the Kennel ratio in d dimensions without (a) and with (b) temporarily decorrelated NNs

poral decorrelation provides accurate estimates of FNN for $d < 10$, and clear separation of deterministic trends.

9.4.1 New Composite-Fraction-Based Metrics

Using the results of the statistical analysis described in Section 4, and combining them with the original definition of FNN, the following FNN *composite fraction* can be advocated:

$$R_{nn}(d; r, s) \triangleq \text{Prob}\left\{ (\delta_i > r\,\epsilon_i) \ \vee \ (\delta_i > s\,\sigma_x) \right\} . \tag{9.17}$$

This new ratio has two parameters: r is the same as for the original fraction, while s is the same as for the reliable fraction. The first part is expected to serve as good indication of FNN for low d, while the second provides good indication of minimal embedding dimension and best separation of deterministic data from random/stochastic for larger d.

This composite metric still does not address the problem of NNs being close to each other due to the presence of noise instead of either dynamics or projection. Therefore, we propose a procedure similar to the FNS, where instead of looking up just one NN (NN) for each point $\mathbf{x}_i$, we look up k NNs. Then, the true, temporarily uncorrelated NN (i.e. the one close only due to dynamics or projection) is identified by evaluating the preceding and future l states of all k NNs. In particular, we evaluate the average distances between the base strand and all k NN strands of length $2l + 1$. The strand with the smallest average distance from the base strand will contain the needed true NN.

9.4.2 New Composite-Fraction-FNN Algorithm

The composite FNN procedure can be summarized as follows:

1. For each point $\mathbf{x}_i$, identify its temporarily uncorrelated k NNs

$$\{\mathbf{x}_{j(m)}\}_{m=1}^{k} = \left\{\mathbf{x}_{j(m)} : \ |i - j(m)| > w, \ m = 1, \ldots, k\right\}$$

just like in Eq. (9.10) to eliminate temporarily correlated NNs.

2. Then the true NN $\mathbf{x}_j$ is identified as:

$$j(m) = \arg\min_{m} \frac{\sum_{q=-l}^{l} \left\|\mathbf{x}_{j(m)+q} - \mathbf{x}_{i+q}\right\|}{(2l+1)\left\|\mathbf{x}_{j(m)} - \mathbf{x}_i\right\|} , \tag{9.18}$$

and used in Eq. (9.17) to determine FNN fraction.

Equation (9.18) indicates that the composite fraction will also be a function of l and k parameters:

$$R_{nn} = R_{nn}(d; r, s, l, k) . \tag{9.19}$$

It is reasonable to set the value of $l \leq w/2$ in Eq. (9.18) since we are focusing on temporal correlations. The choice of k can be influenced by many factors, such as noise level, density of points in the embedding, and sampling time. Here we only present results obtained with just one NN point ($k = 1$ and $l = 0$) for noise free data. In addition, we use $k = 10$ and $l = \tau$ for data with additive noise. However, we explicitly set these values in what follows and do not study their influence on the quality of the results, which is left for future work. Therefore, $R_{nn} = R_{nn}(d; r, s, \tau, k) \triangleq R_{nn}(d; r, s)$ in what follows.

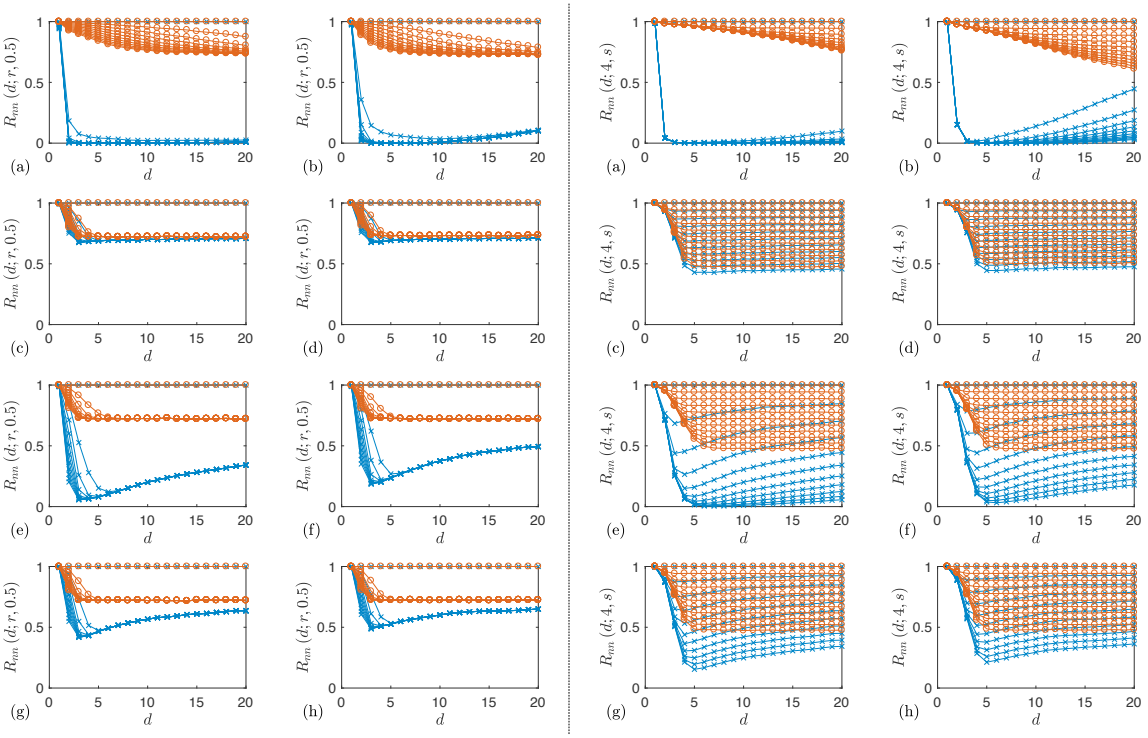

Figure 9.8: Eq. (9.17) based composite fraction FNN algorithm with $s = 0.5$ (left plots) and with $r = 4$ (right plots) for Lorenz (a), Duffing (b), Normal (c), Uniform (d), AR(0.02) (e), AR(0.05) (f), AR(0.2) (g), and AR(0.5) (h). The gradual decrease in the fractions is precipitated by incrementing r from zero to 20 by two and s from zero to two by 0.1. Lines with circles reflect the corresponding surrogate data results.

The results using Eq. (9.17) are shown in Fig. 9.8 for fixed $s = 0.5$ (left plots) and for fixed $r = 4$ (right plots) for all the test data. For the deterministic data, the algorithm works well for $r \geq 4$ and for all $s > 0.2$ values, while for the same r values it provides correct indication of high-dimensional data for the white and correlated stochastic time series. In particular, the actual results are indistinguishable from the surrogate analysis for the random data and the composite fraction never drops below 50 %. For correlated stochastic data with a smaller stochastic component, we can clearly see that there is some deterministic component to the data, while for a large stochastic

component, the results are very similar to the ones for the white noise. However, there still is the small but clearly identifiable separation between the surrogate and correlated stochastic data. This is especially evident for $s = 0.5$ plots, where for larger values of d there is a clear convergence in the FNN composite fraction to the same constant value for the surrogates, and to different values for all the correlated stochastic data sets. The separation of this surrogate FNN fraction from the actual FNN fraction seems to scale with d and the relative magnitude of the deterministic component. Therefore, we can use this separation between the actual and surrogate data composite fractions as an indication of some deterministic trends in the stochastic data. However, this needs further investigation for practical applicability in the analysis, with the purpose of relating the observed differences with the surrogates to the relative strength of the deterministic components.

One may expect the removal of temporal correlations and the new definition of NN points to positively alter the Kennel fraction results. However, the observed improvements were marginal: (1) marginally better indication of minimal embedding dimension for the deterministic data; (2) removal of artificial separation between the surrogates and actual data for the stochastic time series; and (3) accentuation exponential divergence for the deterministic data for larger embedding dimensions. Thus, the main deficiencies that were identified previously still remain.

9.4.3 New-Composite-Fraction-FNS Algorithm

We have also modified the FNS algorithm to use a new metric specified by Eq. (9.17). We use exactly the same procedure as before, except instead of Eq. (9.12) we use:

$$R_{ns}(d; r, s) \triangleq \text{Prob}\left\{ \langle \delta_i \rangle_k > r \, \langle \epsilon_i \rangle_k \ \vee \ \langle \delta_i \rangle_k > s \, \sigma_x \right\} , \tag{9.20}$$

where

$$\langle \epsilon_i \rangle_k \triangleq \langle \epsilon_i \rangle_{i \in S_k} = \frac{1}{\|S_k\|} \sum_{i \in S_k} \epsilon_i \, . \tag{9.21}$$

The results of calculation using Eq. (9.20) are shown in Fig. 9.9 for fixed $s = 0.5$ (left plots) and for fixed $r = 10$ (right plots) for all the test data.

For all the simulations we have used $w = \tau$. The differences with the new FNN are similar to what was observed previously in the reliable algorithm. The decrease of the FNS composite fraction to zero is slower and more gradual than for the FNN composite fraction for the deterministic data. However, the FNS composite fraction shows a much clearer indication for $d = 3$ for the Lorenz and $d = 4$ for the Duffing data as opposed to FNN, where we have to plot data on a log scale to clearly see the minima at the same values. This algorithm can be clearly used to differentiate the deterministic versus stochastic or random data using surrogate analysis. In addition, for the smaller stochastic component data, the surrogate analysis also provides the needed indication of the deterministic component. However, this indication is absent for the larger stochastic components for which they are indistinguishable from the random data sets.

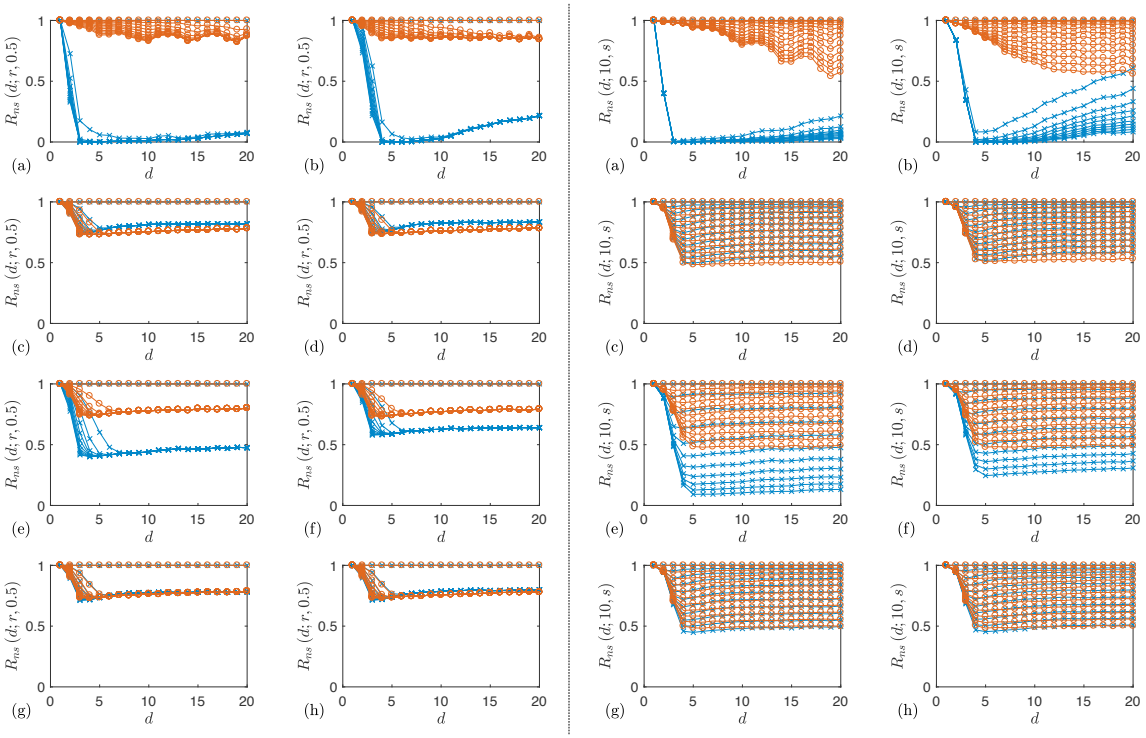

Figure 9.9: Eq. (9.20) based composite fraction FNS algorithm with $s = 0.5$ (left plots) and with $r = 10$ (right plots) applied to Lorenz (a), Duffing (b), Normal (c), Uniform (d), AR(0.02) (e), AR(0.05) (f), AR(0.2) (g), and AR(0.5) (h). The gradual decrease in the fractions is precipitated by incrementing s from zero to two by 0.1 and r from zero to 20 by two. Lines with circles reflect the corresponding surrogate data results.

9.4.4 Noise Robustness of Composite Fraction Algorithms

To test the applicability of these new algorithms for the noisy deterministic time series we have tested them on both Lorenz and Duffing data contaminated by additive noise as described before. The composite FNN fraction results for noisy Lorenz and Duffing are shown in Fig. 9.10 for $k = 1$ NN (left plots) and for $k = 10$ NNs (right plots), respectively. For both cases, we set $r = 10$ and $s = 1$. Plots demonstrate that using ten NNs in the algorithm considerably improves on results using just one NN. Composite FNN fractions with ten NNs have a clear inflection point, or elbow, in their trends at the appropriate minimal embedding dimension for low noise levels. Even for high noise levels the estimates for Duffing are good and for Lorenz only overestimate the embedding dimension by one.

New composite FNS plots for noisy Lorenz and Duffing data are shown in Fig. 9.11 left and right plots, respectively. Compared to the composite FNN fraction plots, these do not show as clear of an indication when the FNS composite fraction levels out or has a clear inflection point at the

appropriate minimal embedding dimensions. Instead, it has a more gradual transition requiring some careful thresholding to identify the appropriate minimal embedding dimensions. It should be noted that these results are similar to the ones obtained for the stochastic time series with small stochastic components. Therefore, we have to be careful when making any judgments about the nature of additive noise versus systematic stochasticity. However, we can still clearly use these new methods in both cases to identify the deterministic components in the data and their dimensionality.

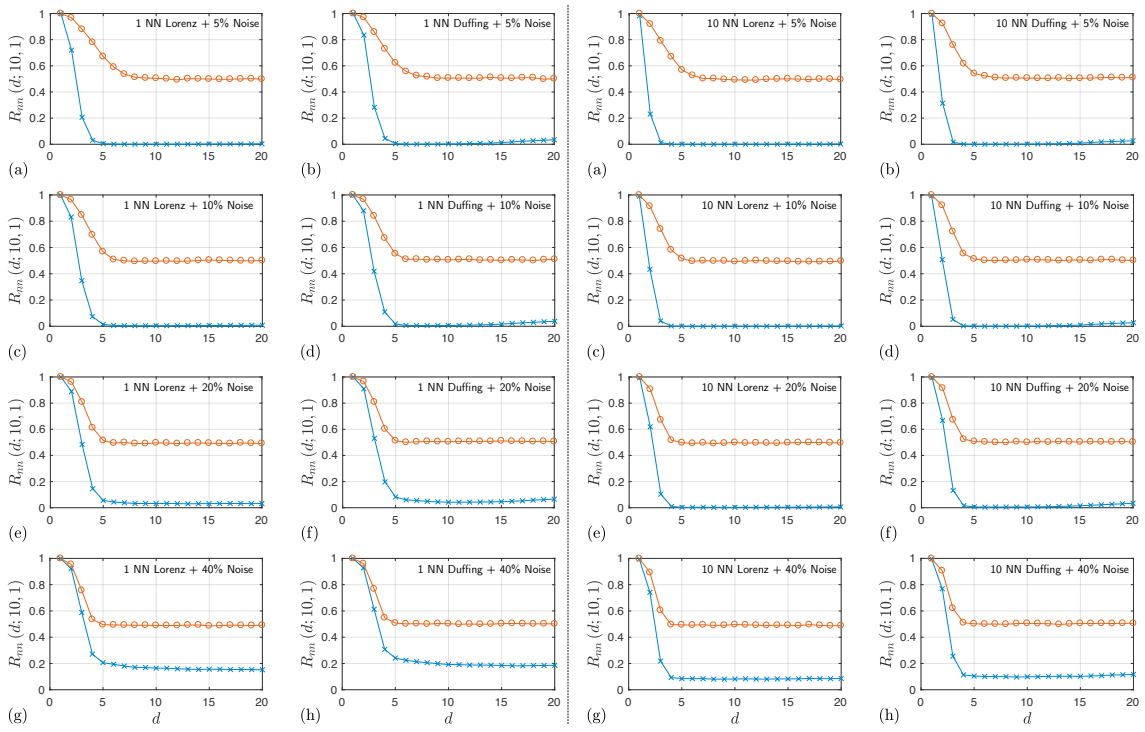

Figure 9.10: Eq. (9.19) based FNN algorithms with $k = 1$ (left plots) and $k = 10$ (right plots) for Lorenz (first and third columns) and Duffing (second and fourth columns) and the surrogate data with 5 % (a,b), 10 % (c,d), 20 % (e,f), 40 % (g,h) additive noise levels. Lines with circles reflect the corresponding surrogate data results.

9.4.5 Concluding Remarks on False Nearest Neighbors

1. The expected $(d+1)$-dimensional distance between the d-dimensional nearest neighbors is a good indicator of minimal embedding dimension for the clean deterministic time series, but it loses its differentiation when even small additive noise is introduced into the deterministic data.

2. New algorithms based on the *composite fraction* overcome the deficiencies of the earlier meth-

ods, and are applicable even for noisy deterministic data.

3. While both composite FNN and FNS concepts provide results that can be used for estimating the minimal embedding dimension even in a noisy data set, the composite FNS algorithm provides a clearer indication of the minimal embedding dimension for the noise free data compared to the composite FNN.

4. However, the composite FNN algorithm has a more differentiating power and provides less ambiguous results for the deterministic data contaminated with additive noise.

5. These FNN/FNS composite fractions are also suitable in evaluating the presence and relative magnitude of a deterministic component in a stochastic or noisy data set.

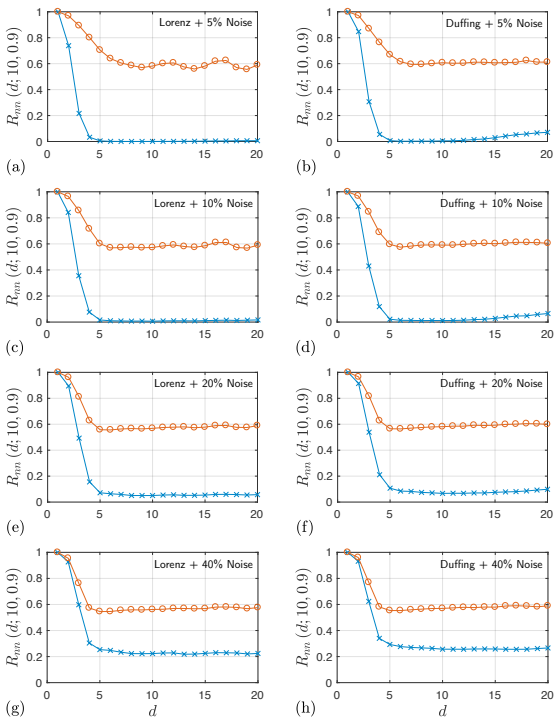

Figure 9.11: Eq. (9.20) based FNS algorithms for Lorenz (left column) and Duffing (right column) and the surrogate data with 5 % (a,b), 10 % (c,d), 20 % (e,f), 40 % (g,h) additive noise levels. Lines with circles reflect the corresponding surrogate data results.

Problems

Problem 9.1

Create an artificial time series by integrating the Chua's circuit [67, 68] by Matsumoto *et al.* (1985):

$$\dot{x} = \alpha \left(y - x + bx + \frac{1}{2}(a-b)\left(|x+1| - |x-1|\right) \right) \tag{9.22}$$

$$\dot{y} = x - y + z \tag{9.23}$$

$$\dot{z} = -\beta y \tag{9.24}$$

using these parameters: $\alpha = 9$, $\beta = 100/7$, $a = 8/7$, and $b = 5/7$. Start the simulation using the following initial conditions: $x(0) = 0$, $y(0) = 0$, and $z(0) = 0.6$. In particular, using the x and y time series:

1. Make sure you are using an appropriate sampling frequency or sampling time period to generate the trajectory (check using power spectrum and auto-correlation), and record at least 60,000 evenly sampled points.

2. Determine the appropriate delay for the delay coordinate reconstruction and verify it visually.

3. Use false nearest neighbors method to determine the minimally necessary embedding dimension using composite FNN fraction algorithms.

4. Now add some Gaussian noise to the data and do the same calculations for different signal-to-noise ratios.

5. Interpret your results.

Some comments on using the supplied programs to plot the FNN results.

1. `false_neighbors_kd_n.m` outputs only the statistics on the nearest neighbors in the requested dimensions in `nn` structure.

2. `plot_false_neighbors.m` processes the output of the `false_neighbors_kd_n.m` and provides the actual false nearest neighbor metrics inside the `nn` structure.

3. you can ask for ranges for the input thresholds `r` (usually between 0 to 20, say `r = 1:2:20;`), and `s` (usually between 0 to 1, say `s = 0.1:0.1:1;`)

4. Most of the FNN metrics inside `nn` are matrices (which can be plotted using `mesh` or `surf` commands), however two most important ones `nn.P` and `nn.S` are three-dimensional arrays. They need special care in plotting.

5. The first coordinate in `nn.P` and `nn.S` corresponds to the embedding dimensions considered, the second coordinate is threshold for the Kennel ratio (`r`), and the third threshold for the $(d+1)$-th distance (`s`).

6. To do a 3D plot of FNNs versus `d` embedding dimension and `r` you can just use `mesh(nn.P(:,:,k));`, where `k` is the index of the `s` for which you want to make a plot.

7. To do a 3D plot of FNNs versus `d` embedding dimension and `s`, first you need to permute the three-dimensional array as `nnP = permute(nn.P,[1,3,2]);` and then you can use `mesh(nnP(:,:,k));` to plot, where `k` now is the index of the `r` for which you want to make a plot.

Chapter 10

Dimension Theory

We are used to the notion of *dimension* from *vector spaces*: dimension of a vector space V is the minimum number of independent bases vectors needed to span V. Therefore, a *point* has dim $= 0$; a *line* has dim $= 1$; a plane has dim $= 2$, and etc. In other words, *dimension* is the number of independent coordinates needed to specify a point in a "space." However, there is a problem: Cantor showed that there is a one-to-one correspondence between $\mathbb{R}^1$ and $\mathbb{R}^2$ (i.e., they have the same *cardinality*), and Peano actually constructed a continuous map from $\mathbb{R}^1$ to $\mathbb{R}^2$ (Peano curve). Then, the question is do $\mathbb{R}^1$ and $\mathbb{R}^2$ have same dimension?

This suggests the need for a *topological definition of dimension*:

Topological Dimension: A set $\mathcal{X}$ has Lebesgue covering dimension or topological dimension D if every open cover $\mathcal{C}$ of $\mathcal{X}$ has a refinement $\mathcal{C}^1$ such that every point $x \in \mathcal{X}$ is contained in *at most* $D+1$ sets in $\mathcal{C}$, and D is the *minimum* such integer.

Remarks

1. We can think of the sets $\mathcal{C}$ as open balls centered at x with radius r:

$$\mathcal{B}(x,r) = \{y \in \mathcal{X} \,|\, d(x,y) < r\} \ . \tag{10.1}$$

   However, in actual topology a distance function is not used to define open sets.

2. The topological dimension corresponds to our "usual" notion of dimension. In particular, for $\mathbb{R}^m$, $D = m$ (see Figs. 10.1 and 10.2).

3. This notion of dimension (also called *Brovwer dimension*) has *nothing* to do with dynamics. In particular, our set $\mathcal{A}$ is being generated by an orbit that visits the various portions of $\mathcal{A}$.

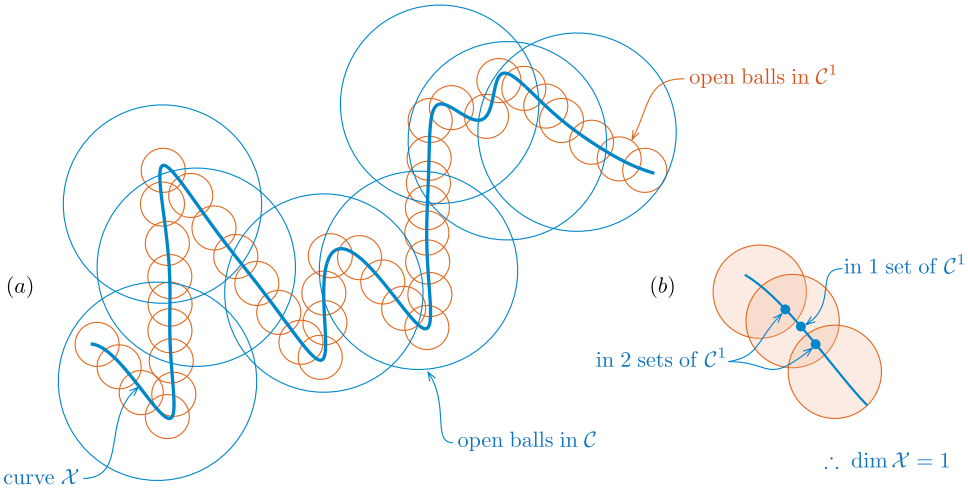

Figure 10.1: For a curve $\mathcal{X}$, refinement cover $\mathcal{C}^1$ is such that every point $x \in \mathcal{X}$ is contained in *at most* 2 sets in $\mathcal{C}^1$, therefore $\dim \mathcal{X} = 1$.

10.1 Invariant Measure of an Attractor

Recall that we are discussing *invariant sets* $\mathcal{A} \in \mathcal{S}$. This means that for every $\mathbf{u}_0 \in \mathcal{A}$, we have $\varphi_t(\mathbf{u}_0) \in \mathcal{A}\ \forall t$ (see Fig. 10.3). As the dynamics travels over $\mathcal{A}$, the probability of visiting different parts of $\mathcal{A}$ will be given by a *probability density* p defined by

$$\lim_{T\to\infty} \int_0^T f(\varphi_t(\mathbf{u}_0)\mathrm{d}t = \int_{\mathcal{A}} f(\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u}\,, \tag{10.2}$$

where by definition $\int_{\mathcal{A}} p(\mathbf{u})\mathrm{d}\mathbf{u} = 1$.

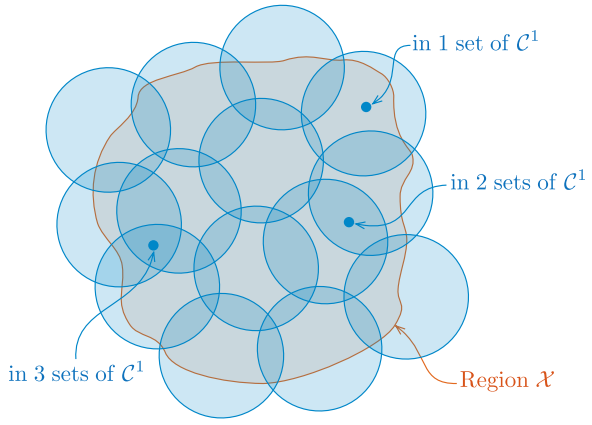

Figure 10.2: For a region $\mathcal{X}$, refinement cover $\mathcal{C}^1$ is such that every point $x \in \mathcal{X}$ is contained in *at most* 3 sets in $\mathcal{C}^1$, therefore $\dim \mathcal{X} = 2$.

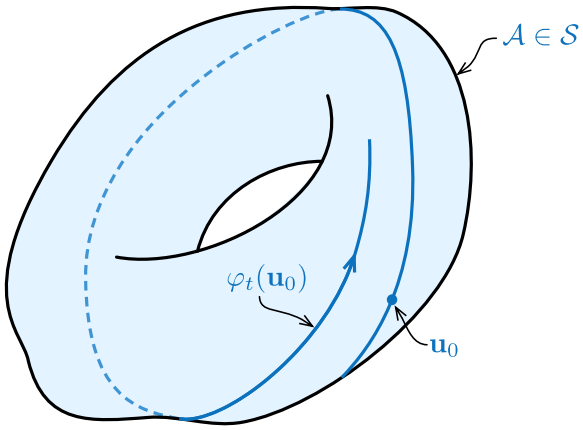

Figure 10.3: For every $\mathbf{u}_0 \in \mathcal{A}$, we have $\varphi_t(\mathbf{u}_0) \in \mathcal{A}\ \forall t$.

Remarks

1. If $p(\mathbf{u})$ is the same for almost all[1] $\mathbf{u}_0$ the basin of attraction of a given attractor, p is a *natural density*.

2. This density is *invariant* if we have *conservation of probability*:

$$\int_{\mathcal{B}} p(\mathbf{u})\mathrm{d}\mathbf{u} = \int_{\varphi_t(\mathcal{B})} p(\mathbf{u})\mathrm{d}\mathbf{u}\,, \tag{10.3}$$

   for $\mathcal{B} \in \mathcal{A}$ (see Fig. 10.5). In other words, the probability that the orbit visits the set $\mathcal{B}$ is equal to the probability that it visits the set $\varphi_t(\mathcal{B})$ for all t.

3. Note that by definition, the invariant density is *ergodic* with respect to the dynamics.

[1]Here, "almost all" means except on a set $\mathcal{C}$ with $\int_{\mathcal{C}} p(\mathbf{u})\mathrm{d}\mathbf{u} = 0$ (set of measure zero).

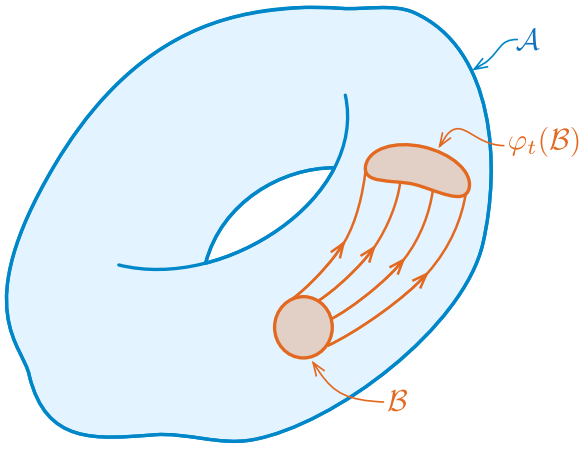

Figure 10.4: For every $\mathbf{u}_0 \in \mathcal{A}$, we have $\varphi_t(\mathbf{u}_0) \in \mathcal{A}\ \forall t$.

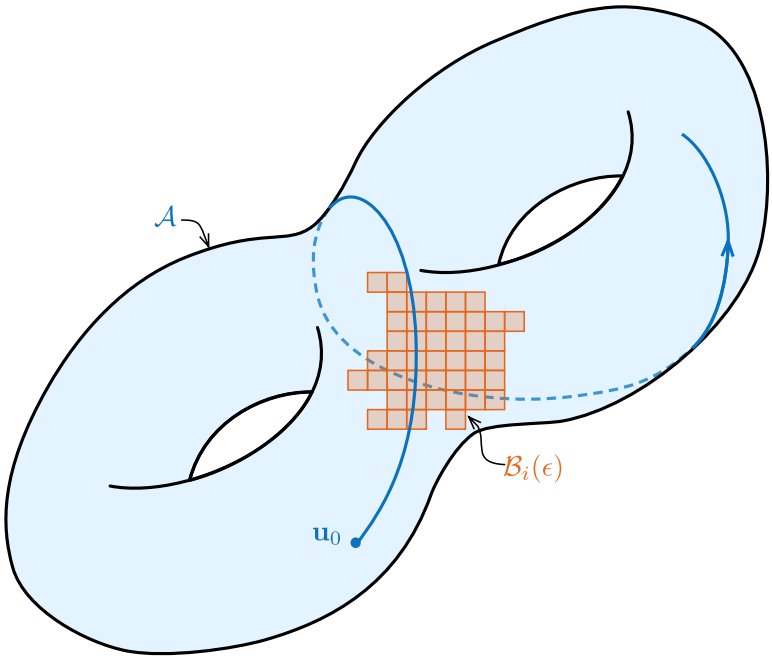

Figure 10.5: $\mathcal{A}$ is covered by uniform boxes $\mathcal{B}_i(\epsilon)$ of size ϵ.

4. Practically speaking, then, the *natural invariant density* is that generated by all orbits observed to lie on the *same attractor*.

5. The fact is that $p(\mathbf{u})$ can be very *nonsmooth* or *spikey*. Indeed, it may be difficult to define $p(\mathbf{u})$, whereas one can always define the *invariant probability measure*, $\mu(\mathbf{u})$ (or cumulative probability distribution), so that we use $\int_{\mathcal{B}} \mathrm{d}\mu$ instead.

6. When $p(\mathbf{u})$ *does* exist, then $\mathrm{d}\mu(\mathbf{u}) = p(\mathbf{u})\mathrm{d}\mathbf{u}$ and we say μ is absolutely continuous with respect to *Lebesque measure* (so $p = \frac{\mathrm{d}\mu}{\mathrm{d}\mathbf{u}}$).

10.2 Dimension Spectrum D_q

Viewing from the perspective of topological dimension, consider an attractor $\mathcal{A}$ covered by uniform boxes $\mathcal{B}_i(\epsilon)$ of size ϵ (see Fig. 10.5). Then

$$p_i = \int_{\mathcal{B}_i} \mathrm{d}\mu = \lim_{T\to\infty} \eta\left(\mathcal{B}_i, \mathbf{u}_0, T\right) , \tag{10.4}$$

where $\eta\left(\mathcal{B}_i, \mathbf{u}_0, T\right)$ is a function of time (i.e., relative number of data points) the orbit spends in box $\mathcal{B}_i$. If μ is the invariant measure, this limit will be the same for almost all $\mathbf{u}_0$. Then define

$$I_q(\epsilon) = \sum_{i=1}^{N(\epsilon)} p_i^q \tag{10.5}$$

for $q \geq 0$. Idea is that for $q > 0$, larger μ_i have more effect on I.

Observe that

$$I_q(\epsilon) = \sum_{i=1}^{N(\epsilon)} p_i^{q-1} p_i = \langle p_i^{q-1} \rangle \,, \tag{10.6}$$

where $\langle p_i^{q-1} \rangle$ is average of p_i^{q-1} on $\mathcal{A}$. Therefore,

$$\sum_{i=1}^{N(\epsilon)} p_i^{q-1} p_i \approx \int_{\mathcal{A}} p_\epsilon(\mathbf{u})^{q-1} p \mathrm{d}\mathbf{u} = \int_{\mathcal{A}} p_\epsilon(\mathbf{u})^{q-1} \mathrm{d}\mu \,, \tag{10.7}$$

where $p_\epsilon(\mathbf{u})$ is the probability in a ball of radius ϵ centered at $\mathbf{u}$. But from definition of p (or μ), this is

$$C_q(\epsilon) \triangleq \int_{\mathcal{A}} p_\epsilon(\mathbf{u})^{q-1} \mathrm{d}\mu = \lim_{T\to\infty} \int_0^T p_\epsilon \left(\varphi_t(\mathbf{u}_0)\right)^{q-1} \mathrm{d}t \,, \tag{10.8}$$

where $C_q(\epsilon)$ is the *Generalized Correlation Integral*, and we have

$$I_q(\epsilon) \sim C_q(\epsilon) \,, \tag{10.9}$$

or $I_q(\epsilon)$ is asymptotic to $C_q(\epsilon)$ as $\epsilon \to 0$.

If $\mathcal{A}$ attractor is *self similar*

$$C_q(\epsilon) \sim \left(\epsilon^{q-1}\right)^{D_q} = \epsilon^{(q-1)D_q} \,. \tag{10.10}$$

For some exponent D_q.

Remark: While historically motivated by thinking of the right hand side of Eq. (10.10) as a "generalized volume," this is essentially a definition of *self similarity*.

Thus

$$D_q \sim \frac{1}{q-1} \frac{\ln C_q(\epsilon)}{\ln \epsilon} \,, \tag{10.11}$$

or

$$D_q = \lim_{\epsilon\to 0} \frac{1}{q-1} \frac{\ln C_q(\epsilon)}{\ln \epsilon} \,, \tag{10.12}$$

which is called the *generalized dimension* of order q, when it exists. Since, $I_q \sim C_q$, we can also write

$$D_q = \lim_{\epsilon\to 0} \frac{1}{q-1} \frac{\ln I_q(\epsilon)}{\ln \epsilon} \,, \tag{10.13}$$

Remarks

1. Definition Eq. (10.52), in terms of "boxes" is in some way easier to understand; Eq. (10.12) is better for actual data-based estimates.

2. If $q = 0$, $I_0(\epsilon) = \sum_{i=1}^{N(\epsilon)} p_i^0 = N(\epsilon)$, so

$$D_0 \sim \frac{\ln N(\epsilon)}{\ln(1/\epsilon)} \,, \tag{10.14}$$

or it reflects how does number of "boxes" needed to cover $\mathcal{A}$ scale with ϵ. This is usually the definition of the *box counting* or *capacity* dimension. In other words, $D_0 = D_F$ in the embedding theorem.

3. When $q = 1$ we need to use l'Hospital's rule to get:

$$\begin{aligned} D_1 &\sim \lim_{q\to 1} \frac{\ln \sum p_i^q}{(q-1)\ln\epsilon} \\ &= \lim_{q\to 1} \frac{\frac{1}{\sum p_i^q}\sum \frac{\mathrm{d}}{\mathrm{d}q} p_i^q}{\ln\epsilon} \\ &= \lim_{q\to 1} \frac{\frac{1}{\sum p_i^q}\sum \frac{\mathrm{d}}{\mathrm{d}q} e^{q\ln p_i}}{\ln\epsilon} \\ &= \lim_{q\to 1} \frac{\frac{1}{\sum p_i^q}\sum p_i^q \ln p_i}{\ln\epsilon}\,. \end{aligned} \tag{10.15}$$

   Since $\lim_{q\to 1}\sum p_i = 1$, we get

$$D_1 \sim \frac{\sum p_i \ln p_i}{\ln\epsilon}\,. \tag{10.16}$$

   D_1 is called the *information dimension* since $-\sum p_i \ln p_i = -\langle \ln p_i\rangle$, is the Shannon information—the average number of "digits" (or "bits" if we use $\log_2$) to specify the state to an accuracy of ϵ.

4. In special case when $p_i = p_j \equiv p$, then $p = N(\epsilon)^{-1}$ and

$$\begin{aligned} D_q &\sim \frac{1}{q-1}\frac{\ln I_q(\epsilon)}{\ln\epsilon} = \frac{1}{q-1}\frac{\ln\sum N^{-q}}{\ln\epsilon} \\ &= \frac{1}{q-1}\frac{\ln N\left(N^{-q}\right)}{\ln\epsilon} = \frac{1-q}{q-1}\frac{\ln N(\epsilon)}{\ln\epsilon} \quad = \frac{\ln N(\epsilon)}{\ln\frac{1}{\epsilon}} = D_0 \equiv D_F \end{aligned} \tag{10.17}$$

   independently of q.

5. In general, $D_{q_1} \le D_{q_2}$ for $q_1 > q_2$.

6. $D_q = D$ for "normal" sets like points, lines, planes, surfaces, etc. If $D_f \neq D$, we say that the object is "fractal." If $D_q \neq D$ for all q the object is a "multifractal."

7. D_2 is easiest to measure and it is called the *correlation dimension*. Also, when $q = 2$:

$$C_2(\epsilon) = \int_{\mathcal{A}} p_\epsilon(\mathbf{u})\mathrm{d}\mu \tag{10.18}$$

   is the *correlation integral*.

8. Since $D_F = D_0$, $D_2 \le D_0$, to choose embedding dimension d, we require $d > 2D_F \approx 2D_2$ (typically, D_F is "close" to D_2). Now if d is a minimal good value for embedding dimension, $d/2 > D_F$, or $d/2 > D_0 \ge D_1 \ge D_2 \ge \cdots$.

10.3 Some Illustrative Examples

10.3.1 A Line Segment

Let us consider a line segment shown in Fig. 10.6(a) covered by balls $\mathcal{B}_i$. For all $x \in \mathcal{A}$, $x \in 2\mathcal{B}_i$. Thus, topological dimension $D = 1$. For a line segment of length L in Fig. 10.6(b) covered by

uniform boxes of size ϵ, the number of boxes $N(\epsilon) \sim L/\epsilon$. Therefore,

$$\begin{aligned} D_0 &= \lim_{\epsilon \to 0} \frac{\ln\left(\frac{L}{\epsilon}\right)}{\ln\left(\frac{1}{\epsilon}\right)} = \lim_{\epsilon \to 0} \frac{\ln L - \ln \epsilon}{\ln 1 - \ln \epsilon} \\ &= \lim_{\epsilon \to 0} \frac{\ln \epsilon}{\ln \epsilon} = 1 \,. \end{aligned} \tag{10.19}$$

Therefore, $D_F = D = 1$ or topological dimension is equal to the box counting dimension.

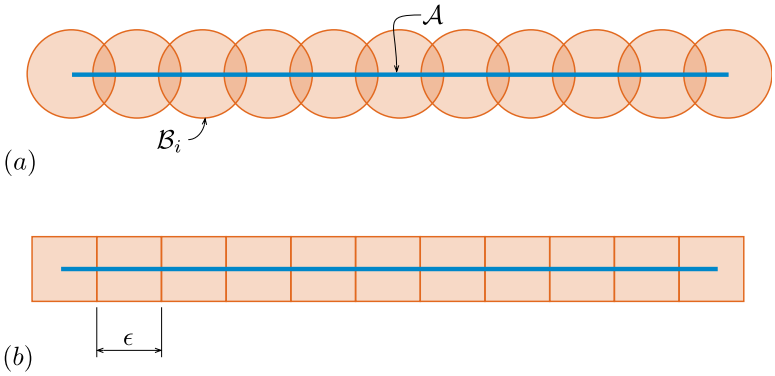

Figure 10.6: A line segment $\mathcal{A}$ covered by balls $\mathcal{B}_i$ (a) and covered by uniform boxes of size ϵ (b)

10.3.2 An Area

We know that the topological dimension for the area $D(\mathcal{A}) = 2$. Now, if we imagine covering this area by N uniform boxes of size ϵ, then

$$N(\epsilon) \approx \frac{A(\mathcal{A})}{A(\mathcal{B}_\epsilon)} = \frac{A}{\epsilon^2} \,, \tag{10.20}$$

where A is the total area. Now, we consider a box counting dimension

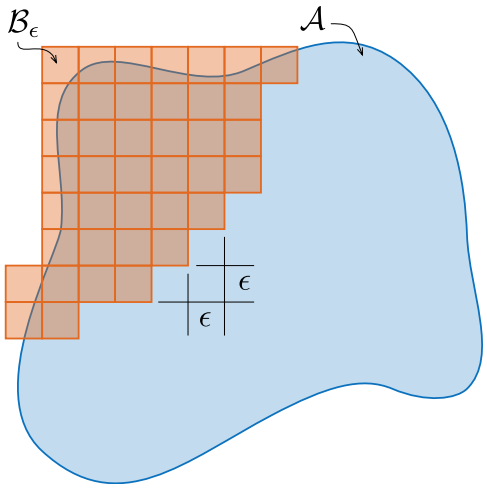

Figure 10.7: An area $\mathcal{A}$ of size A covered by uniform boxes $\mathcal{B}_\epsilon$ of size ϵ

$$
\begin{aligned}
D_0 &= \lim_{\epsilon\to 0} \frac{\ln\left(\frac{A}{\epsilon^2}\right)}{\ln\left(\frac{1}{\epsilon}\right)} = \lim_{\epsilon\to 0} \frac{\ln A - 2\ln\epsilon}{\ln 1 - \ln\epsilon} \\
&= \lim_{\epsilon\to 0} \frac{2\ln\epsilon}{\ln\epsilon} = 2\,.
\end{aligned}
\tag{10.21}
$$

Therefore, $D_F = D = 2$ or, again, we show that topological dimension is equal to the box counting dimension.

10.3.3 Middle Thirds Cantor Set

In this example, $\mathcal{A}$ is generated as shown in Fig. 10.8:

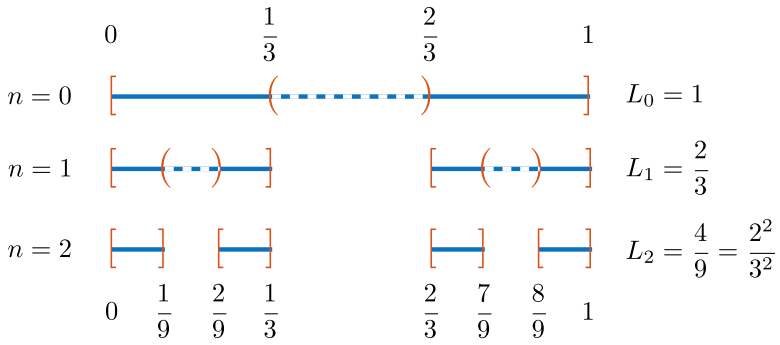

Figure 10.8: Generation of middle thirds Cantor set

1. Take interval $[0,1]$ and remove the interval $(\frac{1}{3},\frac{2}{3})$ to get

$$\left[0,\frac{1}{3}\right]\cup\left[\frac{2}{3},1\right]$$

2. Remove the middle 1/3 of each of those to get

$$\left[0,\frac{1}{9}\right]\cup\left[\frac{2}{9},\frac{1}{3}\right]\cup\left[\frac{2}{3},\frac{7}{9}\right]\cup\left[\frac{8}{9},1\right]$$

3. Repeat this at each step as $n\to\infty$.

The total length after n^{th} step can be expressed as $L_n = \left(\frac{2}{3}\right)^n \to 0$ as $n\to\infty$. Therefore, $\mathcal{A}$ has *zero Lebesque measure*. Then the question is: is $\mathcal{A}$ just a set of *discrete points*? However, $\mathcal{A}$ is also *uncountable*, since we can make a one-to-one correspondence between $\mathcal{A}$ and $[0,1]$ using binary tree:

$$
\begin{array}{llll}
[\quad\quad 0 \quad\quad] & [\quad\quad 1 \quad\quad] & n=1 \\
[\quad 0 \quad] \quad [\quad 1 \quad] & [\quad 0 \quad] \quad [\quad 1 \quad] & n=2 \\
[0] \quad [1] \quad [0] \quad [1] & [0] \quad [1] \quad [0] \quad [1] & n=3
\end{array}
\tag{10.22}
$$

Therefore, all $x\in\mathcal{A}$ can be written as $\{b_1\, b_2\, \ldots\, b_n\}$, where $b_i = 0, 1$. For example, the fist member of $\mathcal{A}$ can be written as an infinite sequence of zeros, and the last member as an infinite sequence of ones. The topological $D = 0$, since all points are in at most one box $\mathcal{B}_i$, given that $\mathcal{A}$ consists of

discrete points. However, for D_0 we can pick $\epsilon = \left(\frac{1}{3}\right)^n \equiv \epsilon_n$ for each iteration n. Then at $n = 1$, $N(\epsilon_1) = 2$; at $n = 2$, $N(\epsilon_1) = 4$. So, $N(\epsilon_n) = 2^n$ and thus,

$$D_0 = \lim_{n\to\infty} \frac{\ln 2^n}{\ln 3^n} = \frac{\ln 2}{\ln 3} \approx 0.6309\,. \tag{10.23}$$

Therefore, the middle thirds Cantor set is a *fractal*.

Remarks

1. We have

$$D_0 = \lim_{n\to\infty} \frac{\ln N(\epsilon)}{\ln(1/\epsilon)}\,, \tag{10.24}$$

thus,

$$N(\epsilon) \sim \epsilon^{-D_0}\,. \tag{10.25}$$

That is, $\mathcal{A}$ is *self similar*.

2. Perfect self similarity is not generally seen in experiments. However, one can see the scaling

$$N(\epsilon) \approx C\epsilon^{-D_q} \tag{10.26}$$

over a range of ϵ.

10.4 Fractal Sets from Dynamical System

The logical question is: can something like a Cantor set arise from dynamics? To answer this, we consider the following one-dimensional map:

$$x_{n+1} = F(x_n)\,, \tag{10.27}$$

where

$$F(x) = \begin{cases} 2\mu x & \text{for} \quad x < \frac{1}{2}\,, \\ 2\mu(x-1)+1 & \text{for} \quad x > \frac{1}{2}\,. \end{cases} \tag{10.28}$$

Please note that for $\mu = 1$, this is a $2x$ mod 1 map. In addition, for $x_0 > 1$, $x_n \to \infty$; and for $x_0 < 0$, $x_n \to -\infty$. Therefore, we focus on $x_n \in [0,1]$ region. Considering the diagram shown in Fig. 10.9, at each iteration the middle strip of width $\left(1 - \frac{1}{2\mu}\right) - \frac{1}{2\mu} = \frac{\mu-1}{\mu} \triangleq \Delta$ leaves $[0,1]$ never to return. We could ask if any points remain as $n \to \infty$. To answer this, we consider uniform probability density of initial conditions x_0, where $p(x_0) = 1$ on $[0,1]$. Then $x_1 = F(x_0)$ for all $x_0 \in [0,1]$. Now consider the integral:

$$\int_{-\infty}^{\infty} p(x_0)\mathrm{d}x_0 = \int_{-\infty}^{\infty} p(x_0)\left(\frac{\mathrm{d}F}{\mathrm{d}x_0}\right)^{-1}\mathrm{d}x_1 \equiv \int_{-\infty}^{\infty} p(x_1)\mathrm{d}x_1 = 1\,. \tag{10.29}$$

So $p(x_1) = p(x_0)\left(\frac{\mathrm{d}F}{\mathrm{d}x_0}\right)^{-1}$. In our case, $\frac{\mathrm{d}F}{\mathrm{d}x_0} = \mu$, thus $p(x_1) = 1/\mu$ *uniformly*. In other words, $p(x_1) = 1 - \frac{\mu-1}{\mu} = \frac{1}{\mu}$, the middle Δ is the "lost" probability. So the probability of remaining in

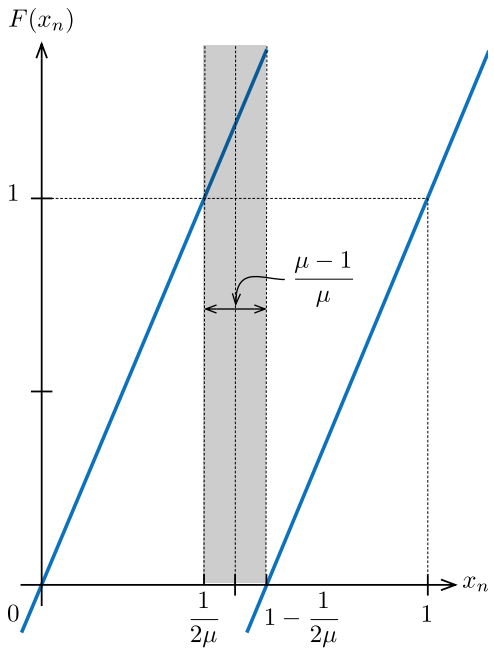

Figure 10.9: Diagram for map defined in Eq. (10.28)

$[0, 1]$ goes to 0 as $\left(\frac{1}{\mu}\right)^n$. Therefore, if points remain, they have zero measure or are a set of isolated points.

A bit of thinking about points that remain after n iterations shows a picture shown in Fig. 10.10. That is, we get a generalized Cantor set. However, this is *not* an "attractor"—in fact, it is unstable, though invariant.

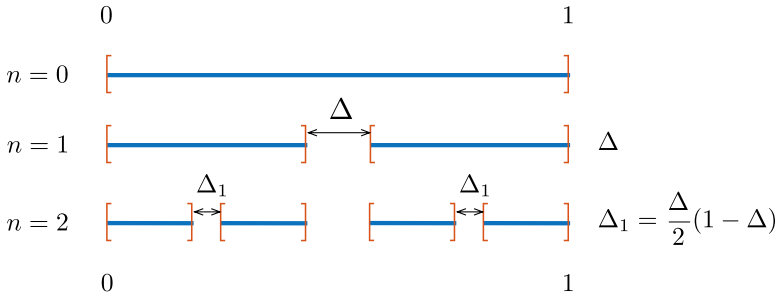

Figure 10.10: Generalization of Cantor set

10.5 Dimensionality of Continuous Attractors

10.5.1 Periodic Orbits

For any periodic response we get periodic orbits in the phase space for which $D = 1 = D_0$.

Paradigm: Any periodic response (see Fig. 10.11).

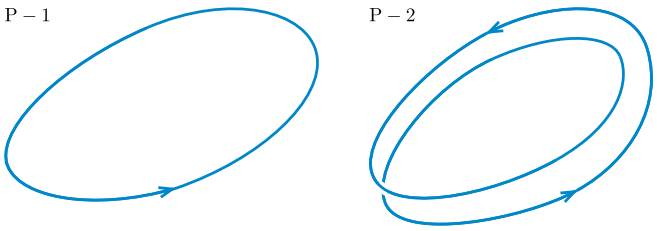

Figure 10.11: Closed orbits of any period $D = 1 = D_0$.

10.5.2 Quasi-periodic Orbits

For a quasi-periodic orbit we need to consider a linear combination of periodic signals with incommensurate periods. This generates a flow on n-torus, where n is the number of incommensurate frequencies. An example for $n = 2$ can be $x(t) = A_1 \sin t + A_2 \sin \sqrt{7}\, t$, which can be visualized as evolving on a surface of a donut (2-torus) as shown in Fig. 10.12. In this case, $D = n = D_0$.

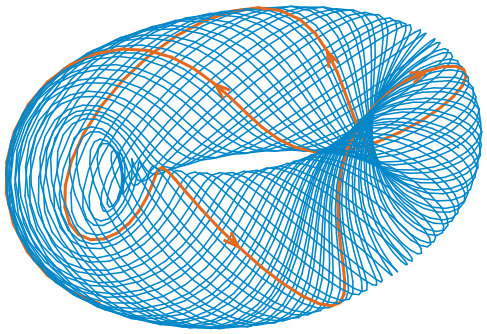

Figure 10.12: A quasi-periodic orbit for $x(t) = A_1 \sin t + A_2 \sin \sqrt{7}\, t$ response

Paradigm: Coupled van der Pol oscillator:

$$\begin{aligned} \ddot{x} + \mu\dot{x}(1 - x) + x &= \mu y \,, \\ \ddot{y} + \mu\dot{y}(1 - y) + y &= \mu x \,, \end{aligned} \tag{10.30}$$

which has quasi-periodic solutions.

10.5.3 Chaotic Orbits

Here we will have a continuous directing along the time axis (or forcing phase) and we would have *fractal directions* usually seen in Poincaré sections as in Fig. 10.13. Here, $D < D_0$, where D_0 is a non-integer.

Paradigms:

1. Driven two-well Duffing chaos ($2 < D_0 < 3$);

2. Lorenz equation ($D_0 \approx 2 + \epsilon$); and

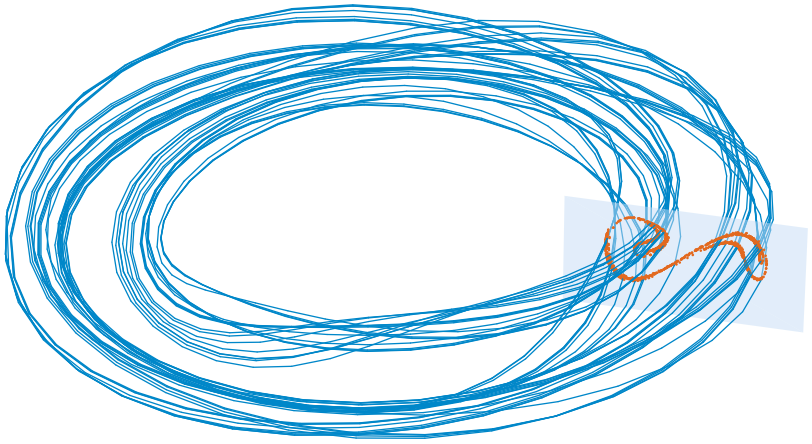

Figure 10.13: A snapshot of a chaotic orbit from a driven two-well Duffing oscillator showing fractal structure in a stroboscopic Poincaré section

3. Chua circuit ($2 < D_0 < 3$).

Please note that a flow must have $D > 2$ dimensions for chaotic behavior to be possible, while maps can be chaotic in one-dimension. The reason is, in $2 - D$ uniqueness of the solutions "traps" bounded orbits into either fixed points or limit cycles. In other words, there is not enough room in $2 - D$ for chaos as shown in Fig. 10.14.

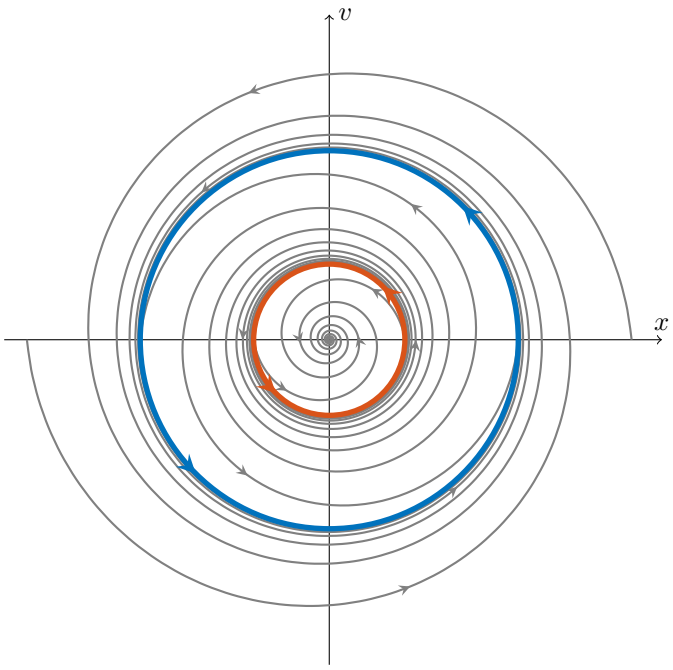

Figure 10.14: Uniqueness of solution in two-dimensions traps bounded flows either to fixed points (e.g., $(0, 0)$ stable spiral) or limit cycles (red is unstable and blue is stable).

10.6 Experimental Estimates of Fractal Dimension

In previous chapter we have discussed the dimension theory and defined the *dimension spectrum*. It was also mentioned that the easiest and most reliable estimation is for D_2, *correlation dimension*, based on:

$$C_2(\epsilon) \equiv C(\epsilon) = \int_{\mathcal{A}} p_\epsilon(\mathbf{u}) \mathrm{d}\mu \,, \tag{10.31}$$

where ϵ is the size of partition on the attractor $\mathcal{A}$, $\mathbf{u} \in \mathcal{A}$, and μ is the corresponding invariant probability measure.

10.6.1 Correlation Dimension Estimate

Given a time scalar series $\{x_n\}_{n=1}^N$, we start by selecting a "good" delay τ and embedding dimension d to get a vector time series $\{\mathbf{x}\}_{n=1}^{N-(d-1)\tau}$. In what follows, for simplicity, we will assume that we have $\{\mathbf{x}\}_{n=1}^N$ (considering that $N \approx N-(d-1)\tau$ for N large). Then, the estimator of $C(\epsilon)$ can be thought of as a *cumulative distribution* of distances $\|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon$ in our data set. Therefore, we consider the following *correlation sum*:

$$\hat{C}(\epsilon) = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \mathbf{\Theta} \left(\epsilon - \|\mathbf{x}_i - \mathbf{x}_j\| \right) \,, \tag{10.32}$$

where $\mathbf{\Theta}$ is the Heaviside function.

Eq. (10.32) is a good estimator for the $C(\epsilon)$ if our data consists of points randomly drawn from our attractor $\mathcal{A}$. However, we usually have a continuously sampled time series, which cannot be considered to be randomly drawn from $\mathcal{A}$. From false nearest neighbors chapter, we know that temporal correlations can lead to skewed or biased estimates. In case of estimating the correlation dimension, the temporal correlations can lead to false low-dimensional results. If we have large amount of data, we can randomly draw points from it to generate the needed randomly sampled data to avoid the effect of temporal correlations. However, the availability of abundant data is only viable in simulations and not in experiments. Then, the solution is to only measure distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ between the points which are separated by time greater than the Theiler window w (i.e., $\Delta_{ij} > w$). Then the corresponding *correlation sum* can be written as:

$$\hat{C}(\epsilon) = \frac{2}{(N-w)(N-w-1)} \sum_{i=1}^{N-w-1} \sum_{j=i+w+1}^{N} \mathbf{\Theta} \left(\epsilon - \|\mathbf{x}_i - \mathbf{x}_j\| \right) \,, \tag{10.33}$$

For small ϵ, we expect $\hat{C}(\epsilon) \sim \epsilon^{D_2}$[2], or $\ln \hat{C}(\epsilon) \sim D_2 \ln \epsilon$. Therefore:

$$D_2 \sim \frac{\mathrm{d}\left(\ln \hat{C}(\epsilon)\right)}{\mathrm{d}\left(\ln \epsilon\right)} \,, \tag{10.34}$$

if we are lucky and our data shows this trend.

[2]Here, $\sim$ means as $\epsilon \to 0$, $N \to \infty$.

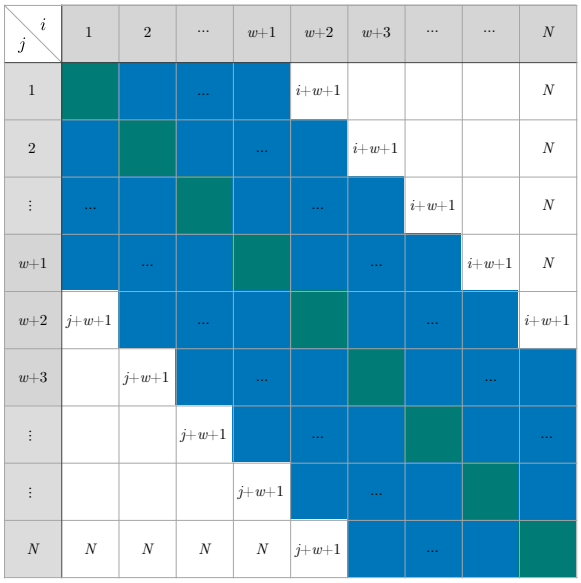

Figure 10.15: Visualization of all the temporarily uncorrelated pairs of points $\mathbf{x}_i$ and $\mathbf{x}_j$ for which the interpoint distance needs to be calculated. The green squares represent $i = j$ and the blue squares are the temporarily correlated points within $|i - j| < w$.

Remarks:

1. Number of point pairs in our data is $\frac{N(N-1)}{2}$, hence the divisor in Eq. (10.32).

2. Number of point pairs that are separated by $\Delta t > w$ in Eq. (10.53) is $\frac{(N-w)(N-w-1)}{2}$, hence the divisor in Eq. (10.53).

3. The Fig. 10.15 shows the basic set up for estimating the correlation sum. It illustrates all the needed distances that we need to calculate for the sum.

4. If we plot $\ln \hat{C}(\epsilon)$ vs $\ln \epsilon$ for a chaotic data, we expect to see three distinct regions (refer to Fig. 10.16):

   (a) For $\epsilon > \epsilon_{\max} = \|\mathbf{x}_i - \mathbf{x}_j\|_{\max}$, $\hat{C}(\epsilon) = 1$, so $\ln \hat{C} \to 0$ as $\epsilon \to \epsilon_{\max}$.

   (b) For $\epsilon < \epsilon_{\min} = \|\mathbf{x}_i - \mathbf{x}_j\|_{\min}$, $\hat{C}(\epsilon) = 0$, so $\ln \hat{C} \to -\infty$ as $\epsilon \to \epsilon_{\min}$. For ϵ small, $\ln \hat{C}(\epsilon)$ no longer looks smooth, since the number of points at those scales is very low.

   (c) Scaling region in which $\ln \hat{C}(\epsilon)$ scales linearly with respect to $\ln \epsilon$.

5. Actually $\mathbf{x}_i = \mathbf{y}_i + \mathbf{n}_i$, where $\mathbf{n}_i$ is noise and $\mathbf{y}_i$ is the true reconstructed point. Now, recall that noise is ∞-dimensional it $\mathbf{n}_i$ are independent. Therefore, $D_2(\text{noise}) \to D_2$ in the limit as $N \to \infty$. So actual $\ln \hat{C}(\epsilon)$ curves would skew towards the ∞ slope for noise dominated scales of ϵ as shown in Fig. 10.16. Therefore, if noise kicks in before the scaling region, we cannot estimate D_2.

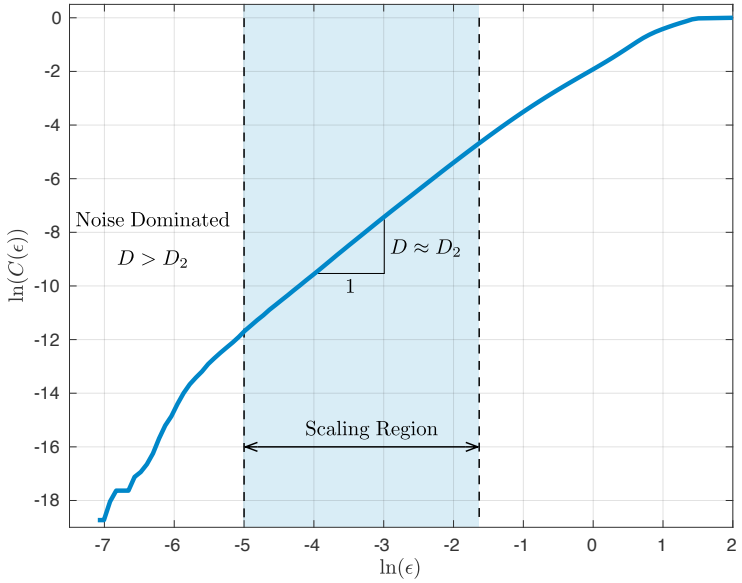

Figure 10.16: Correlation sum estimate $\hat{C}(\epsilon)$ vs ϵ for the Lorenz signal.

6. To estimate D_2, we need to identify the tangent line to $\ln \hat{C}(\epsilon)$ vs $\ln \epsilon$ in the scaling region as shown in Fig. 10.16. It is usually not easy to identify the suitable scaling region even if it is there. One way it can be done is to look at $\frac{\mathrm{d} \ln \hat{C}(\epsilon)}{\mathrm{d}(\ln \epsilon)}$ at each value of $\ln \epsilon$ as shown in Fig. 10.17. We want to ignore any points that show large variability in the beginning due to noise. This should be followed by a smooth and close to horizontal section for the scaling region, which will be followed by the drop in the slope due to distances approaching the attractor size.

10.6.2 Pointwise Dimension

From mathematical perspective, the *pointwise dimension* is used to differentiate between the probability measures μ_1 on a phase space $\mathcal{S}_1$ and μ_2 on another space $\mathcal{S}_2$ which are not equivalent by a locally *bi-Lipschitz* injection. For a point $\mathbf{u} \in \mathcal{S}$ and and a number $\epsilon > 0$, we can define an ϵ-ball as

$$\mathcal{B}(\mathbf{u}, \epsilon) \triangleq \{\mathbf{v} \in \mathcal{S} \,|\, d(\mathbf{u}, \mathbf{v}) < \epsilon\} \,. \tag{10.35}$$

Then we can call a map $F : \mathcal{S}_1 \to \mathcal{S}_2$ bi-Lipschitz if for all $\mathbf{u} \in \mathcal{S}_1$ there exists $r > 0$ and a constant $K \geq 1$ such that for all $\mathbf{x}, \mathbf{y} \in \mathcal{B}(\mathbf{u}, r)$ we have:

$$\frac{1}{K} d_1(\mathbf{x}, \mathbf{y}) \leq d_2\left(F(\mathbf{x}), F(\mathbf{y})\right) \leq K d_1(\mathbf{x}, \mathbf{y}) \,, \tag{10.36}$$

where d_i is the dimension metric in $\mathcal{S}_i$. From this perspective, the measures μ_1 and μ_2 are *equivalent* if there exists a locally bi-Lipschitz injection $F : \mathcal{S}_1 \to \mathcal{S}_2$, such that for all $\mathbf{u} \in \mathcal{S}_2$ we have

$$\mu_1(F^{-1}(\mathbf{u})) = \mu_2(\mathbf{u}) \,. \tag{10.37}$$

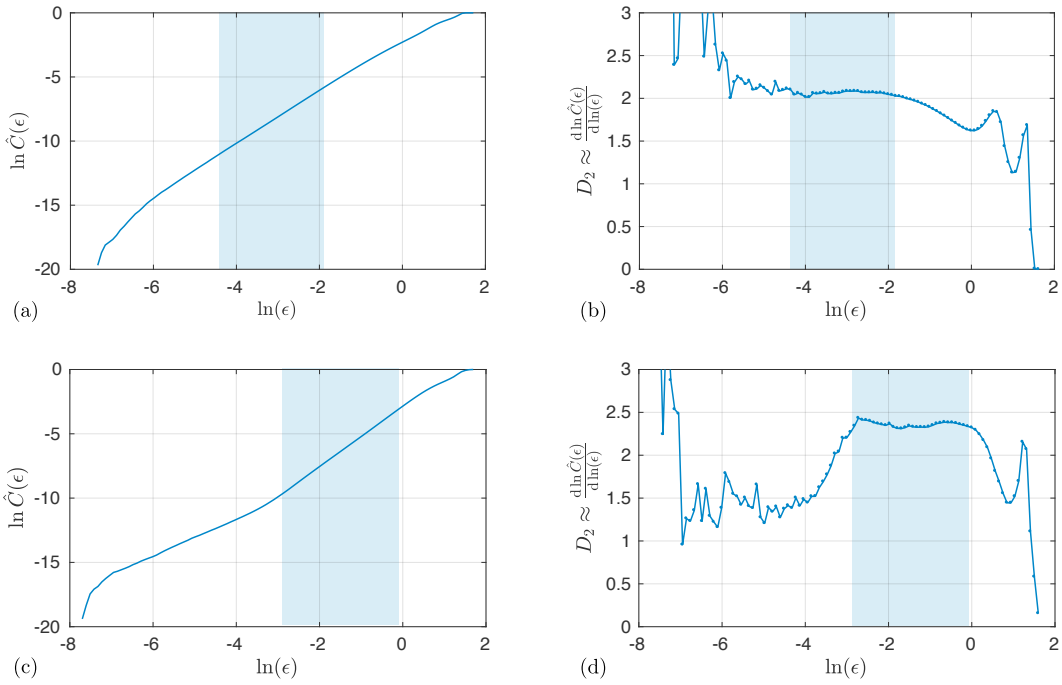

Figure 10.17: Correlation sum and the corresponding estimated of D_2 for the Lorenz (a–b) and two-well Duffing's (c–d) signals. Shaded regions indicate the scaling regions.

Consider geometric objects like a line segment $L \in \mathbb{R}$, area $A \in \mathbb{R}^2$, or, in general, hyper-volume $V \in \mathbb{R}^m$, then for any $\mathbf{u} \in V$ we can observe that (see Fig. 10.18)

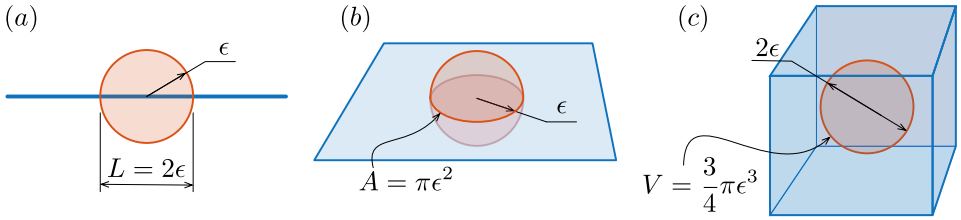

Figure 10.18: The measure inside the ϵ-ball scales like: ϵ for a line segment (a), ϵ^2 for a surface area (b), and ϵ^3 for a three-dimensional volume (c).

$$\mu\left(\mathcal{B}(\mathbf{u}, \epsilon)\right) \sim \epsilon^m \,. \tag{10.38}$$

This motivates the definition of *pointwise dimension* at a point $\mathbf{u} \in \mathcal{A}$ as:

$$D_\mu(\mathbf{u}) = \lim_{\epsilon \to 0} \frac{\ln \mu(\mathcal{B}(\mathbf{u}, \epsilon))}{\ln \epsilon} \,. \tag{10.39}$$

This, in turn, motivates the generalization of *pointwise dimension* over the whole attractor $\mathcal{A}$ as

$$D_P = \int_{\mathcal{A}} D_\mu(\mathbf{u}) \mathrm{d}\mu(\mathbf{u}) = \lim_{\epsilon \to 0} \frac{\int_{\mathcal{A}} \ln \mu_{\mathbf{u}}(\epsilon) \, \mathrm{d}\mu_{\mathbf{u}}(\epsilon)}{\ln \epsilon} = \lim_{\epsilon \to 0} \frac{\langle \ln \mu_{\mathbf{u}}(\epsilon) \rangle_{\mathcal{A}}}{\ln \epsilon} \,, \tag{10.40}$$

where $\mu_{\mathbf{u}}(\epsilon) = \mu(\mathcal{B}(\mathbf{u}, \epsilon))$. There is a clear confusion as to which generalized dimension is the pointwise dimension closely related (J. York [29] claims it is equivalent to information dimension, F.C. Moon [72] claims it is like correlation dimension). However, if we look at the asymptotics of these dimensions:

$$D_1 \sim \frac{\langle \ln p_i \rangle}{\ln \epsilon}, \qquad \text{and} \qquad D_2 \sim \frac{\ln \langle p_i \rangle}{\ln \epsilon}, \tag{10.41}$$

it is clear that pointwise dimension has a similar form to the expression for the *information dimension* D_1, since the probability density inside the ϵ-ball $P_i(\epsilon) = \int_{\mathcal{B}(\mathbf{x}_i,\epsilon)} p_i \, \mathrm{d}\mathbf{x} = \int_{\mathcal{B}(\mathbf{x}_i,\epsilon)} \mathrm{d}\mu = \mu_{\mathbf{x}_i}(\epsilon)$.

10.6.3 Average Pointwise Dimension Estimate

Given a reconstructed phase space vectors $\{\mathbf{x}\}_{n=1}^N$, we can estimate the measure μ for ϵ-ball $\mathcal{B}_i(\epsilon) = \mathcal{B}(\mathbf{x}_i, \epsilon)$ centered at $\mathbf{x}_i$ as:

$$\hat{\mu}_i(\epsilon) = \frac{1}{N} \sum_{j=1}^{N} \mathbf{\Theta} \left(\epsilon - \|\mathbf{x}_i - \mathbf{x}_j\| \right) = \frac{N\left(\mathcal{B}_i(\epsilon)\right)}{N} = \frac{N_i(\epsilon)}{N}, \tag{10.42}$$

where $\mathbf{\Theta}$ is the Heaviside function, and $N_i(\epsilon) = N(\mathcal{B}_i(\epsilon))$ is the number of point inside the ball $\mathcal{B}_i(\epsilon)$. Then, the average log of pointwise measures inside the ϵ-ball can be estimated as:

$$\begin{aligned} \langle \ln \hat{\mu}_i(\epsilon) \rangle &= \frac{1}{N} \sum_{i=1}^{N} \ln \left(\frac{1}{N} \sum_{j=1}^{N} \mathbf{\Theta} \left(\epsilon - \|\mathbf{x}_i - \mathbf{x}_j\| \right) \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} \ln N_i(\epsilon) - \log N \,. \end{aligned} \tag{10.43}$$

Thus, we can estimate the average pointwise dimension as:

$$\hat{D}_P = \lim_{\epsilon \to 0} \frac{\langle \ln \hat{\mu}_i(\epsilon) \rangle}{\ln \epsilon} = \lim_{\epsilon \to 0} \frac{\frac{1}{N} \sum_{i=1}^{N} \ln N_i(\epsilon) - \log N}{\ln \epsilon}, \tag{10.44}$$

or

$$\hat{D}_P \sim \frac{\mathrm{d} \langle \ln \hat{\mu}_i(\epsilon) \rangle}{\mathrm{d} \ln \epsilon} = \frac{\sum_{i=1}^{N} \mathrm{d} \ln N_i(\epsilon)}{N \mathrm{d} \ln \epsilon} \,. \tag{10.45}$$

Algorithms using fixed distance based metrics (e.g., $\hat{D}_P$ estimator is using fixed ϵ to estimate the number of point inside the corresponding ϵ-balls) is called *fixed distance* method or algorithm. There are also algorithms that are called *fixed mass* algorithms. In fixed mass algorithms, instead of counting points inside the fixed size ϵ-balls, we first fix the number of points n we want to be contained inside the balls, and then find the size of these balls $\epsilon_i(n)$ centered at $\mathbf{x}_i$ and containing n points. Using a fixed mass algorithm, we can estimate the average log of $\epsilon_i(n)$-ball as:

$$\langle \ln \epsilon_i(n) \rangle = \frac{1}{N} \sum_{i+1}^{N} \ln \epsilon_i(n) \,, \tag{10.46}$$

where $\epsilon_i(n)$ is defined as the distance between $\mathbf{x}_i$ and its n-th nearest neighbor $\mathbf{x}_{j(n)}$:

$$\epsilon_i(n) = \left\| \mathbf{x}_i - \mathbf{x}_{j(n)} \right\| \,. \tag{10.47}$$

Thus, we can try to estimate fixed mass based average pointwise dimension as:

$$\bar{D}_P = \lim_{n\to 1} \frac{\ln(n/N)}{\langle \ln \epsilon_i(n) \rangle} = \lim_{n\to 1} \frac{N \ln(n/N)}{\sum_{i+1}^{N} \ln \epsilon_i(n)} \,. \tag{10.48}$$

However, as $n \to 1$, $\epsilon_i(n) \to \epsilon_i(1)$ and we cannot make it smaller. Therefore,

$$\bar{D}_P \approx \frac{N \ln(1/N)}{\sum_{i+1}^{N} \ln \epsilon_i(1)} = \frac{-N \ln N}{\sum_{i+1}^{N} \ln \epsilon_i(1)} \,. \tag{10.49}$$

Or more precisely, to let $\langle \epsilon_i(1) \rangle \to 0$, we need to let $N \to \infty$, then

$$\bar{D}_P = \lim_{N\to\infty} \frac{N \ln N}{\sum_{i+1}^{N} \ln(1/\epsilon_i(1))} \,. \tag{10.50}$$

Please note, if we want to use the above expression for the estimation, then we really need to worry about eliminating temporarily correlated nearest neighbors from the calculation. However, if the sampling time period is reasonable and we let $N \to \infty$ or some fairly large number, we may not need to worry about temporal correlations since the nearest neighbors will most likely be there due to geometry of $\mathcal{A}$.

Remarks

In practice, we cannot really let $\epsilon_i(1) \to 0$ or $N \to \infty$, thus we need to estimate D_P using the identified scaling regions as was done for the correlation dimension. Constraints similar to ones identified for the correlation dimension are also present for the estimates of the pointwise dimension. In addition to the artifacts caused by noise, these constraints mainly arise due to the finite nature of the available set of data:

1. For the fixed distance algorithm, and $0 < \epsilon < d_{\min}$, $N_i(\epsilon) = 1$, where the $d_{\min}$ is the minimum nearest neighbor distance of the data. Thus, $\hat{D}_P \to 0$ as $_i(1) \to d_{\min}$, and $\hat{D}_P = 0$ for $0 < \epsilon < d_{\min}$.

2. For the fixed distance algorithm, when $\epsilon > d_{\max}$, $N_i(\epsilon) = N$. Thus, $\hat{D}_P \to 0$ as $\epsilon \to d_{\max}$, and $\hat{D}_P = 0$ as $\epsilon > d_{\max}$

3. For the fixed mass algorithm, as $n \to 1$, $\bar{D}_P \to \frac{N \ln N}{\sum_{i+1}^{N} \ln(1/\epsilon_i(1))}$, which would be an accurate estimate only in the limit as $N \to \infty$.

4. For the fixed mass algorithm, Grassberger (1983) [37] proposed to use $\psi(n) - \ln N$ instead of $\ln n/N$ to compensate for finite sample bias in $\hat{p}_i(\epsilon) = \hat{\mu}_i$ for small ϵ, where ψ is *digamma function*. Therefore, algorithmically we estimate:

$$\bar{D}_P(n) = \frac{N\,(\psi(n) - \ln N)}{\sum_{i+1}^{N} \ln \epsilon_i(n)} \,. \tag{10.51}$$

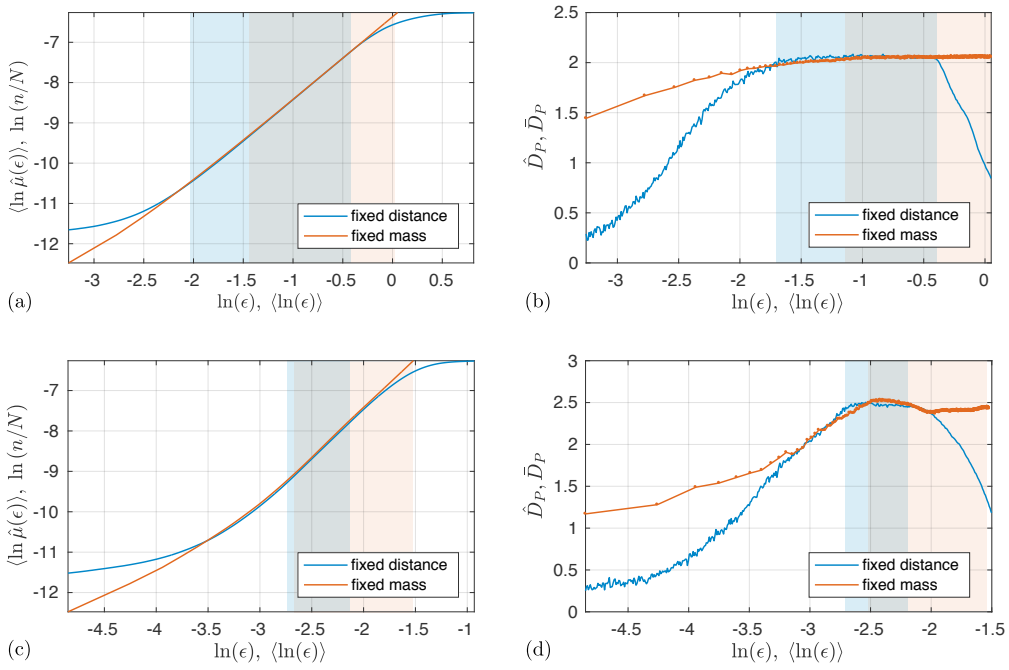

Figure 10.19: Pointwise dimension estimation for the Lorenz (a–b) showing $D_P = 2+\epsilon$ $(0 < \epsilon \ll 1)$, and two-well Duffing's signal (c–d) showing $D_P \approx 2.5$. The shaded areas correspond to the scaling regions.

We may want to remove time correlated neighbors in both algorithms when estimating the probability measure $\hat{\mu}(\epsilon)$ and average ball size $\epsilon(n)$. As $n \to 1$ we may also run into scaling problems and as $\epsilon \to \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_{\max}$ and $D_P \to 0$. This is somewhat alleviated by the Grassberger correction for finite size. Example of the results of these estimations using both fixed distance and fixed mass methods for the Duffing's signal are shown in Fig. 10.19.

10.6.4 Other Generalized Dimension Estimates

The generalized dimension

$$D_q = \lim_{\epsilon \to 0} \frac{1}{q-1} \frac{\ln C_q(\epsilon)}{\ln \epsilon} \tag{10.52}$$

can be estimated for different values of q to explore the multi-fractal nature of the attractor. The estimation of box-counting $D_0 = D_F$ is problematic if we do not have access to the abundant clean data. For the D_1 we can use the pointwise dimension estimates, and for the D_2 we can use the correlation dimension. However, in general, to estimate the generalized dimension D_q for $q \geq 1$ we can use two different approaches: (1) is similar to correlation sum estimation, and (2) uses pointwise dimension procedure. The correlation sum estimation used the distribution of the inter-point lengths

in the data. We can follow the same example and define the estimate of C_q as

$$\hat{C}_q(\epsilon) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{N_i}\sum_{j\not\in[i-w,i+w]}\mathbf{\Theta}\left(\epsilon - \|\mathbf{x}_i - \mathbf{x}_j\|\right)\right]^{q-1}, \tag{10.53}$$

where $N_i = N - w - \min(i-1, w, N-i) - 1$, which is equal to $N_i = N - 2w - 1$ when $w < i < N - w$. For, $q > 2$ this leads to the biased estimator due to nontrivial power in the second sum. In general, this algorithms tends to give lower dimensional estimates than the C_2 algorithm. In addition, it seems that $D_q < D_l$ if $l > q$ even when we do not expect a multi-fractal.

For $q \leq 1$ estimation of fractal dimension is problematic as some of the sums may be zero. Here, a fixed mass algorithm is more appropriate, which will also work for $q > 1$ and provides more consistent results. First, we assume that have total of N phase space points, and we want to use M query points for pointwise estimation. Then, we estimate the measure inside the range of ϵ-balls for each query point $\{\mathbf{x}_i\}_{i=1}^{M}$ as

$$\hat{\mu}_i(\epsilon) = \frac{1}{N}\sum_{k=1}^{K}\mathbf{\Theta}\left(\epsilon - \epsilon_i(k)\right) = \frac{1}{N}\sum_{k=1}^{K}\mathbf{\Theta}\left(\epsilon - \left\|\mathbf{x}_i - \mathbf{x}_{n(k)}\right\|\right), \tag{10.54}$$

where $\epsilon_i(k) = \left\|\mathbf{x}_i - \mathbf{x}_{n(k)}\right\|$, $\mathbf{x}_{n(k)}$ is the k-th nearest neighbor of $\mathbf{x}_i$, and K is large enough so that $\langle\epsilon_i(K)\rangle$ is close to the attractor scale. Then, the generalized correlation sum can be estimated as:

$$\bar{C}_q(\epsilon) = \frac{1}{M}\sum_{i=1}^{M}\left[\frac{1}{N}\sum_{k=1}^{K}\mathbf{\Theta}\left(\epsilon - \epsilon_i(k)\right)\right]^{q-1}. \tag{10.55}$$

Finally, to find D_q, we need to identify a scaling range at intermediate length scales, or look at the plateau of

$$\hat{D}_q(\epsilon) = \frac{\mathrm{d}\ln\hat{C}_q(\epsilon)}{\mathrm{d}\ln\epsilon}, \tag{10.56}$$

or

$$\bar{D}_q(\epsilon) = \frac{\mathrm{d}\ln\bar{C}_q(\epsilon)}{\mathrm{d}\ln\epsilon}. \tag{10.57}$$

These calculations are still problematic if done for $\epsilon < \epsilon_{\max}(1)$. Therefore, we need to make sure that we only consider (or count) points which have nonzero measure for the minimum ϵ used in calculation. Alternatively, we need to increase the minimum ϵ, so that $\epsilon_i(1) > \epsilon_{\min}$ for all i or points considered in the calculation.

10.7 Embedding Dimension from Fractal Dimension

In previous chapter, it was discussed that the estimation of the embedding dimension can also be done by considering how it affects the estimation of some nonlinear metric. Since the correlation and pointwise dimensions are determined by the slope in their scaling regions, insufficiently large dimension will cause the changes in these slopes due to projections. Only after reaching the sufficient

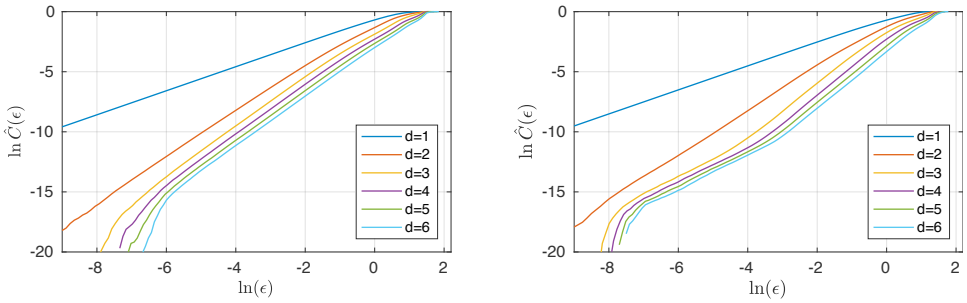

Figure 10.20: Correlation sum for Lorenz and two-well Duffing signals.

embedding dimension, will the slope stabilize and remain the same even if we increase the embedding dimension. Thus, we can use the estimated correlation sum or pointwise measure $\hat{\mu}(\epsilon)$ to find the minimally necessary embedding dimension by observing at which embedding dimension the slopes of the metric becomes invariant to the increase in the embedding dimension. An example of this calculation for the Lorenz's and Duffing's signals are shown in Fig. 10.20 using correlations sum. From the trends in the plots it is clear that using this criterion the minimally necessary embedding dimension for the Lorenz is three and for the Duffing's is four. An example of applying fixed distance and fixed mass based pointwise measure estimation to Lorenz and two-well Duffing's signals are shown in Fig. 10.21.

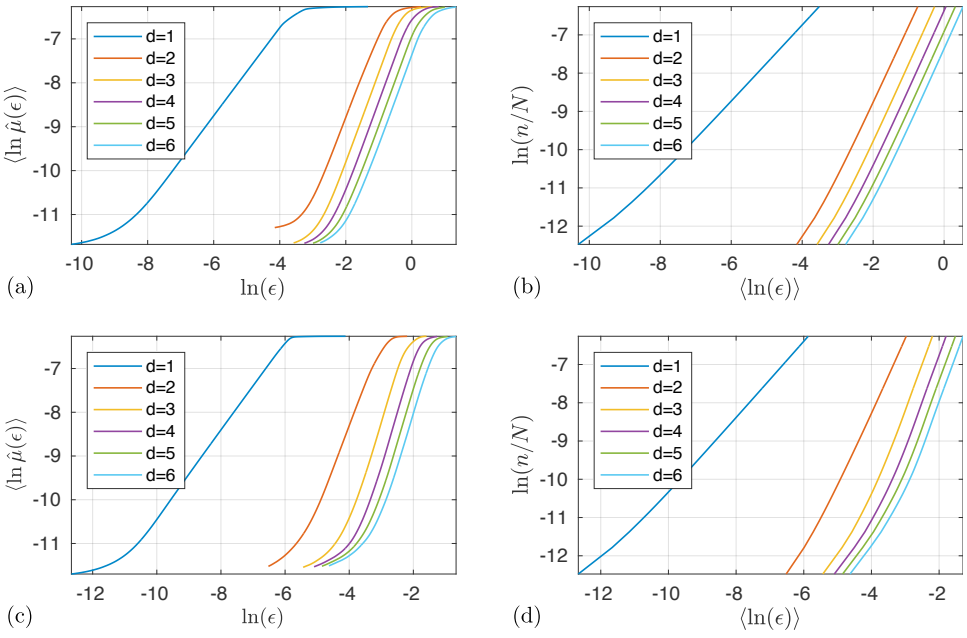

Figure 10.21: Pointwise dimension estimation for the Lorenz (a–b) and two-well Duffing's (c–d) signals using different embedding dimensions for the fixed distance algorithm on the left and fixed mass algorithm or the right

Problems

Problems 10.1

Given a periodic solution, show that $D_2 \approx 1$. Now generate a quasi-periodic time series and show $D_2 \approx n$, where n is the number of incommensurate frequencies. Discuss how closely you are able to get the result. Remember that the estimation of D_2 requires that you choose an embedding dimension d, fix a delay m, and generate C_2. All estimates should be presented in the form, $D_2 = \widehat{D}_2 \pm E$, where E is the standard error. You should also always look at PSD, $S(f)$, and autocorrelation, $R(\tau)$.

Problem 10.2

Take at least two chaotic, or otherwise "complex" time series from Hénnon Map ($a = 1.4$ and $b = 0.3$), Lorenz, double-well Duffing, or Chua's Circuit models and estimate D_2 and $\bar{D}_p$. Discuss the quality of your result and how does it change with the addition of noise. Does the data appear to be "fractal?" Over what range of length scales does your data appear to have a scaling region? Estimate other Generalized dimensions D_q for both $q > 1$ and $q < 1$, what can you say about the results?

Problem 10.3

Now we want to estimate *Kolmogorov-Sinai entropy*, which according to Ruelle (1978) is a lower bound on the sum of Lyapunov exponents discussed in the next chapter. It is usually difficult to calculate, but one can estimate the *correlation entropy*, which is a close lower bound on the K–S entropy. It is given by:

$$K_2 = \lim_{d\to\infty} \lim_{\epsilon\to 0} \lim_{N\to\infty} \log \frac{C_2(d,\epsilon)}{C_2(d+1,\epsilon)}\,, \tag{10.58}$$

where $C_2(d,\epsilon)$ is the correlation sum for an embedding dimension d.

Choose one of the time series from continuous ode's from Problem 10.2. Plot family of curves $\ln C_2(\epsilon)$ vs $\ln \epsilon$ for different values of embedding dimension from 1 to 8. How does the slope of these curves vary with d? Can you estimate the K–S entropy?

Chapter 11

Stability and Lyapunov Exponents

Recall the analysis of *stability* for the *fixed point* of a map:

$$\mathbf{x}_{n+1} = \mathbf{F}\left(\mathbf{x}_n\right) , \tag{11.1}$$

where $\mathbf{x}_n \in \mathbb{R}^m$. The period-one fixed point needs to satisfy $\mathbf{x}^* = \mathbf{F}(\mathbf{x}^*)$. For example, consider a Henon map:

$$\begin{aligned} x_{n+1} &= a - x_n^2 + by_n \\ y_{n+1} &= x_n \, , \end{aligned} \tag{11.2}$$

where $\mathbf{x}_n = [x, y]^T$. Fixed point for the period-one orbit is determined by solving the following equations:

$$\begin{aligned} x &= a - x^2 + by \\ y &= x \, , \end{aligned} \tag{11.3}$$

which results into: $x = a - x^2 + by$ or $x^2 + (1-b)x - a = 0$. This quadratic equation has two solutions

$$x_{1,2}^* = \frac{(b-1) \pm \sqrt{(b-1)^2 + 4a}}{2} \, , \tag{11.4}$$

when $(b-1)^2 + 4a \geq 0$ or $a \geq \frac{-(b-1)^2}{4}$ as shown in Fig. 11.1. That is, as a crosses over the value $\frac{-(b-1)^2}{4}$ we would experience the *saddle-node bifurcation* during which we transition from having no fixed points to having one stable and one unstable fixed points.

11.1 Stability of Periodic Orbits

To study the stability of fixed points, we perturb $\mathbf{x}^*$ solution by a small vector $\mathbf{u} \ll 1$

$$\mathbf{x}_n = \mathbf{x}^* + \mathbf{u}_n \tag{11.5}$$

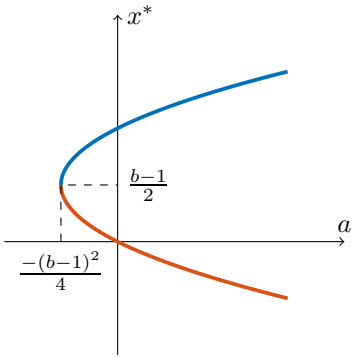

Figure 11.1: Saddle-node bifurcation generating one stable, one unstable fixed points

and study how does this perturbation behave as we let $t \to \infty$. Thus, we plug Eq. (11.5) into Eq. (11.1) to get:

$$\begin{aligned} \mathbf{x}^* + \mathbf{u}_{n+1} &= \mathbf{F}(\mathbf{x}^* + \mathbf{u}_n)\,, \\ &= \mathbf{F}(\mathbf{x}^*) + D\mathbf{F}(\mathbf{x}^*)\mathbf{u}_n + \mathcal{O}(\|\mathbf{u}_n\|^2)\,. \end{aligned} \tag{11.6}$$

Now, remembering that $\mathbf{x}^* = \mathbf{F}(\mathbf{x}^*)$ and ignoring the higher order terms, we get

$$\mathbf{u}_{n+1} = D\mathbf{F}(\mathbf{x}^*)\mathbf{u}_n = \mathbf{A}\mathbf{u}_n \tag{11.7}$$

where

$$\mathbf{A} = D\mathbf{F}(\mathbf{x}^*) = \left[\frac{\partial F_i}{\partial x_j}\right]\Bigg|_{\mathbf{x}=\mathbf{x}^*} \tag{11.8}$$

is the Jacobian matrix evaluated at $\mathbf{x}^*$. Since the perturbations can be expressed as:

$$\mathbf{u}_{n+1} = q_1\hat{\mathbf{e}}_1 + q_2\hat{\mathbf{e}}_2 + \cdots + q_m\hat{\mathbf{e}}_m = \mathbf{V}\mathbf{q}\,, \tag{11.9}$$

where columns of $\mathbf{V} \in \mathbb{R}^{m\times m}$ are $\hat{\mathbf{e}}_i$, and $\hat{\mathbf{e}}_i$ satisfy the following *eigenvalue problem*:

$$\mathbf{A}\hat{\mathbf{e}}_i = s_i\hat{\mathbf{e}}_i\,. \tag{11.10}$$

Then, $\mathbf{x}^*$ is *stable* if $|s_i| \leq 1$ ($|s_i|$ can be complex) and *asymptotically stable* if $|s_i| < 1$. In the former case, $\|\mathbf{u}_n\| < \delta$ for all n if $\|\mathbf{u}_0\| < \epsilon$, and in the latter, $\|\mathbf{u}_n\| \to 0$ as $n \to \infty$. In particular, if $\mathbf{u}_0 = \alpha_0\hat{\mathbf{e}}_i$,

$$\mathbf{u}_1 = \alpha_1\hat{\mathbf{e}}_i = \mathbf{A}\left(\alpha_0\hat{\mathbf{e}}_i\right) = \alpha_0\mathbf{A}\hat{\mathbf{e}}_i = s_i\alpha_0\hat{\mathbf{e}}_i\,, \tag{11.11}$$

so

$$\alpha_{n+1} = s_i\alpha_n\,. \tag{11.12}$$

Thus,

$$\alpha_n = s_i^n\alpha_0\,. \tag{11.13}$$

Remark

The eigenvalues s_i are *stability numbers*. Often, people like to write them in terms of an *exponential* as

$$s_i = e^{\lambda_i} \qquad \text{or} \qquad \lambda_i = \ln s_i \,, \tag{11.14}$$

and λ_i are called *stability exponents*. In this case, stability is determined by the familiar requirement, for continuous time systems: $\Re\{\lambda_i\} \leq 0$.

We can also look for Period-k fixed points (periodic orbits):

$$\{\cdots, \mathbf{x}_m, \mathbf{x}_{m+1}, \mathbf{x}_{m+2}, \cdots, \mathbf{x}_{m+k}, \cdots\} \,, \tag{11.15}$$

where

$$\mathbf{x}_{m+k} = \mathbf{x}_m = \mathbf{x}^* \,. \tag{11.16}$$

In other words,

$$\mathbf{x}^* = \underbrace{\mathbf{F}(\mathbf{F}(\cdots(\mathbf{F}}_{k \text{ times}}(\mathbf{x}^*))\cdots) \,, \tag{11.17}$$

or

$$\mathbf{x}^* = \mathbf{F}^k(\mathbf{x}^*) \,. \tag{11.18}$$

Then stability of Period-k orbit is determined by mapping

$$\mathbf{u}_{n+1} = \mathbf{A}\mathbf{u}_n \,, \tag{11.19}$$

where

$$\mathbf{A} = D\mathbf{F}^k(\mathbf{x}^*) = \left[\frac{\partial F_i^k}{\partial x_j}\right]\Bigg|_{\mathbf{x}=\mathbf{x}^*} \,. \tag{11.20}$$

Therefore, we look for *stability numbers*, s_i, or *exponents*, $\lambda_i = \ln s_i$, where

$$\mathbf{A}\hat{\mathbf{e}}_i = s_i \hat{\mathbf{e}}_i \,. \tag{11.21}$$

11.2 Stability of Chaotic Orbits

We have shown how the *stability* of periodic orbits is evaluated. However, What if we have an orbit that never repeats? Can we still talk about its stability? The answer is yes, we can do it as outlined below.

Consider a 'fiducial' trajectory, that is a sample steady state trajectory starting from an initial point on the attractor $\mathbf{x}_0^*$,

$$\{\mathbf{x}_0^*, \mathbf{x}_1^*, \cdots, \mathbf{x}_n^*, \cdots\} \,. \tag{11.22}$$

By definition $\mathbf{x}_{n+1}^* = \mathbf{F}(\mathbf{x}_n^*)$. As before, we can always consider a perturbation about this fiducial trajectory, $\mathbf{x}_n = \mathbf{x}_n^* + \mathbf{u}_n$, so

$$\begin{aligned} \mathbf{x}_{n+1}^* + \mathbf{u}_{n+1} &= \mathbf{F}(\mathbf{x}_n^* + \mathbf{u}_n) \\ &= \mathbf{F}(\mathbf{x}_n^*) + D\mathbf{F}(\mathbf{x}_n^*)\mathbf{u}_n + \mathcal{O}(\|\mathbf{u}_n\|^2) \,, \end{aligned} \tag{11.23}$$

so, as before, neglecting higher order terms, we get

$$\mathbf{u}_{n+1} = D\mathbf{F}(\mathbf{x}_n^*)\mathbf{u}_n = \mathbf{A}_n\mathbf{u}_n\,, \tag{11.24}$$

where, now, the matrix $\mathbf{A}_n$ depends on n. Unfortunately, we cannot just look at the eigenvalues of $\mathbf{A}_n$, since it changes at each step. However, since

$$\begin{aligned}\mathbf{u}_n &= \mathbf{A}_n\mathbf{A}_{n-1}\mathbf{A}_{n-2}\cdots\mathbf{A}_1\mathbf{u}_0 = \prod_{i=1}^{n}\mathbf{A}_i\mathbf{u}_0 \\ &\triangleq \mathbf{A}^n\mathbf{u}_0 = \mathbf{A}^n(\mathbf{x}_0^*)\mathbf{u}_0\,,\end{aligned} \tag{11.25}$$

$\mathbf{A}^n$ depends on $\mathbf{x}_0^*$ *only*. Then we could look at eigenvalues of $\mathbf{A}^n$. Unfortunately, in practice, that is not a simple task. Instead, it is better to look at the behavior of $\|\mathbf{u}_n\|$ *directly*:

$$\begin{aligned}\|\mathbf{u}_n\|^2 = \mathbf{u}_n^T\mathbf{u}_n &= (\mathbf{A}^n\mathbf{u}_0)^T(\mathbf{A}^n\mathbf{u}_0) \\ &= \mathbf{u}_0^T\,><?\,>?\,>?\,>??\,> ????(\mathbf{A}^n)^T(\mathbf{A}^n)\mathbf{u}_0 \triangleq \mathbf{u}_0^T\mathbf{B}_n\mathbf{u}_0\,,\end{aligned} \tag{11.26}$$

where $\mathbf{B}_n = (\mathbf{A}^n)^T\mathbf{A}^n$ is a positive, real, Hermitian (symmetric) matrix. Therefore, all eigenvalues and eigenvectors are real and eigenvalues are all nonnegative. Now, we can denote these eigenvalues as s_{ni}^2 to get

$$\mathbf{B}_n\hat{\mathbf{e}}_{ni} = s_{ni}^2\hat{\mathbf{e}}_{ni}\,. \tag{11.27}$$

Now, we assume a particular perturbation $\mathbf{u}_0 = \alpha_0\hat{\mathbf{e}}_{ni}$, then

$$\frac{\|\mathbf{u}_n\|^2}{\|\mathbf{u}_0\|^2} = \frac{\mathbf{u}_0^T\mathbf{B}_n\mathbf{u}_0}{\|\mathbf{u}_0\|^2} = \frac{\alpha_0\hat{\mathbf{e}}_{ni}^T\left(s_{ni}^2\alpha_0\hat{\mathbf{e}}_{ni}\right)}{\alpha_0^2} = s_{ni}^2 \triangleq \left(\bar{s}_i^n\right)^2\,. \tag{11.28}$$

In other words, we define an *average constant* $\bar{s}_i$, such that if iterated n times would give the same value: $\bar{s}_i^n = s_{ni}$. Therefore, we can write

$$\bar{s}_i^n \triangleq \left(e^{\bar{\lambda}_i}\right)^n = e^{n\bar{\lambda}_i} = \frac{\|\mathbf{u}_n\|}{\|\mathbf{u}_0\|} \tag{11.29}$$

or

$$\bar{\lambda}_i = \frac{1}{n}\ln\left(\frac{\|\mathbf{u}_n\|}{\|\mathbf{u}_0\|}\right)\,, \tag{11.30}$$

where $\mathbf{u}_0$ is chosen such that $\mathbf{u}_0 \parallel \hat{\mathbf{e}}_{ni}$, and $\bar{\lambda}_i$ is called n-step *stability exponent*. In addition, we can write

$$\bar{\lambda}_i = \frac{1}{n}\ln\left(\frac{\left(\mathbf{u}_0^T\mathbf{B}_n\mathbf{u}_0\right)^{\frac{1}{2}}}{\|\mathbf{u}_0\|}\right) = \frac{1}{2n}\ln\left(\mathbf{u}_0^T\mathbf{B}_n\mathbf{u}_0\right) - \frac{1}{n}\ln\|\mathbf{u}_0\|\,, \tag{11.31}$$

where the last term $\frac{1}{n}\ln\|\mathbf{u}_0\| \to 0$ for large n. Then, for each $\mathbf{u}_0 \parallel \hat{\mathbf{e}}_i$, we *define*

$$\lambda_i = \lim_{n\to\infty}\frac{1}{2n}\ln\left(\mathbf{u}_0^T\mathbf{B}_n\mathbf{u}_0\right) \tag{11.32}$$

to be the *Lyapunov Exponent*, which we can always order as

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d\,, \tag{11.33}$$

which we call *Lyapunov Spectrum*.

Remarks

1. Another way to write Eq. (11.32):

$$\lambda_i = \lim_{n\to\infty} \frac{1}{2n} \ln \left\| D\mathbf{F}^n(\mathbf{x}_0^*)\mathbf{u}_0 \right\|^2 , \tag{11.34}$$

or

$$\lambda_i = \lim_{n\to\infty} \frac{1}{n} \ln \left\| D\mathbf{F}^n(\mathbf{x}_0^*)\mathbf{u}_0 \right\| , \tag{11.35}$$

where $\mathbf{u}_0 \parallel \hat{\mathbf{e}}_i$, and $D\mathbf{F}^n(\mathbf{x}_0^*) = \prod_{i=0}^{n} D\mathbf{F}(\mathbf{x}_i^*)$.

2. The following theorem guarantees that these limits exist:

> **Multiplicative Ergodic Theorem (Oseledec):** Given an ergodic invariant measure on $\mathcal{A}$, the following *Oseledec matrix*
>
> $$\mathbf{H} = \lim_{n\to\infty} (\mathbf{B}_n)^{\frac{1}{2n}} \tag{11.36}$$
>
> exists and is independent of $\mathbf{x}_0^*$ except on a set of measure zero.

Note that:

$$\mathbf{H}_n^{2n} \triangleq \mathbf{B}_n , \tag{11.37}$$

where $\mathbf{B}_n$ is a positive matrix, so it has a square root. Since

$$\mathbf{B}_n \hat{\mathbf{e}}_{ni} = s_{ni}^2 \hat{\mathbf{e}}_{ni} = (\bar{s}_i^n)^2 \hat{\mathbf{e}}_{ni} , \tag{11.38}$$

we know

$$\mathbf{H}_n^{2n} \hat{\mathbf{e}}_{ni} = \bar{s}_i^{2n} \hat{\mathbf{e}}_{ni} , \tag{11.39}$$

and, therefore,

$$\mathbf{H}_n \hat{\mathbf{e}}_{ni} = \bar{s}_i \hat{\mathbf{e}}_{ni} . \tag{11.40}$$

Thus, the Lyapunov numbers (and exponents) are those of $\mathbf{H} = \lim_{n\to\infty} \mathbf{H}_n$.

3. Since the λ_i are independent of initial conditions (almost for all $\mathbf{x}_0$), then the λ_i are *properties* of the attractor.

4. If $\lambda_1 > 0$ and $\sum \lambda_i < 0$, the attractor is chaotic. In other words, it combines the features of stability (it is an attractor after all) and instability ($\|\mathbf{u}_0\|$ grows in at least one direction $\hat{\mathbf{e}}_1$).

5. In addition, λ_i are *invariant* under diffeomorphisms and, therefore, under delay coordinate embedding.

6. Since $\bar{\lambda}_i = \ln \bar{s}_i$,

$$\sum_{i=1}^{m} \bar{\lambda}_i = \ln \left(\prod_{i=1}^{m} \bar{s}_i \right) = \ln \left(\det \mathbf{H}_n \right) = \frac{1}{n} \ln \left[\left| \det \left(D\mathbf{F}^n(\mathbf{x}_0) \right) \right| \right] , \tag{11.41}$$

so, if the system is *dissipative* (i.e., the phase space volumes collapse in time), we have

$$\sum_{i=1}^{m} \bar{\lambda}_i = \frac{1}{n} \ln \left[\left| \det \left(D\mathbf{F}^n(\mathbf{x}_0) \right) \right| \right] < 0 \, . \tag{11.42}$$

For example, recall Henon map:

$$\begin{aligned} x_{n+1} &= a - x_n^2 + b y_n \\ y_{n+1} &= x_n \, , \end{aligned} \tag{11.43}$$

where $\mathbf{x}_n = [x, y]^T$. Then

$$D\mathbf{F} = \begin{pmatrix} 2x_n & b \\ 1 & 0 \end{pmatrix} , \tag{11.44}$$

so $\det(D\mathbf{F}) = -b$ and is not a function of $\mathbf{x}_n$. Thus, we can evaluate

$$\frac{1}{n} \ln \left[\left| \det \left(D\mathbf{F}^n(\mathbf{x}_0) \right) \right| \right] = \frac{1}{n} \ln |b|^n = \ln |b| \, . \tag{11.45}$$

Therefore, $\lambda_1 + \lambda_2 = \ln |b|$ and it is negative if $|b| < 1$.

7. Flows versus maps:

   - For flows, one $\lambda_i = 0$, corresponding to the direction along the orbit ($e^0 = 1$).
   - Also $\sum \lambda_i = \operatorname{div} \mathbf{f}(\mathbf{x}) = \nabla \mathbf{f}(\mathbf{x})$, where $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$.
   - For Poincaré map, $\lambda_{\max} = \lambda_{\text{flow}} \, \bar{t}_\Sigma$, where $\bar{t}_\Sigma$ is the average time between crossings of section Σ.
   - Relationship to Dimensions:

$$D_L = k + \frac{1}{\lambda_{k+1}} \sum_{i=1}^{k} \lambda_i \, , \tag{11.46}$$

     where k is the largest integer such that $\sum_{i=1}^{k} \lambda_i \geq 0$. $D_L \equiv D_{KY}$ is the Lyapunov or *Kaplan-Yorke* dimension. There is also Kaplan-Yorke conjecture: $D_L \equiv D_1$, which is proved for two-dimensional maps.

11.3 Experimental Lyapunov Exponents

Now we will discuss how can we experimentally estimate Lyapunov exponent as defined by Eq. (11.35),

$$\lambda_i = \lim_{n \to \infty} \frac{1}{n} \ln \left\| D\mathbf{F}^n(\mathbf{x}_0) \mathbf{u}_0 \right\| , \tag{11.47}$$

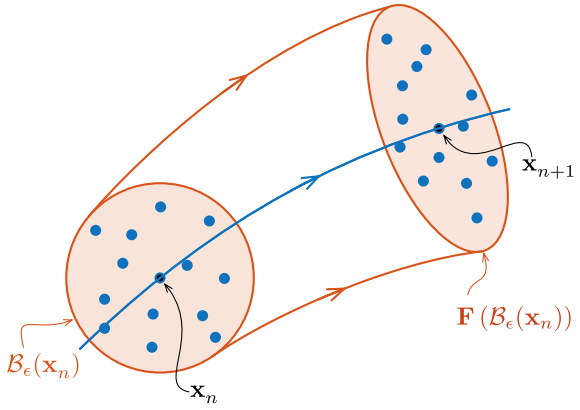

Figure 11.2: Estimating Jacobian through regression

from experimentally measured scalar time series or measured trajectory. As was the case with the fractal dimensions, we start from the phase space trajectory $\{\mathbf{x}_n\}_{n=1}^N$. Generally one might try to estimate $D\mathbf{F}(\mathbf{x}_n)$ at each phase space point using regression on its nearest neighbor points contained inside an ϵ-ball $\mathcal{B}_\epsilon(\mathbf{x}_n)$ and the corresponding points at next time step contained in $\mathbf{F}\left(\mathcal{B}_\epsilon(\mathbf{x}_n)\right)$ as shown in Fig. 11.2. In other words, we minimize

$$e^2 = \frac{1}{N_\epsilon} \sum_{\mathbf{x}_j \in \mathcal{B}_\epsilon(\mathbf{x}_n)} \|\mathbf{x}_{j+1} - \mathbf{F}(\mathbf{x}_j)\|^2 \,. \tag{11.48}$$

However, in general, we do not know the form of $\mathbf{F}$. Therefore, if ϵ is small enough ($\|\mathbf{x}_j - \mathbf{x}_n\| < \epsilon$)

$$\mathbf{x}_{j+1} = \mathbf{F}(\mathbf{x}_j) \approx \mathbf{F}(\mathbf{x}_n) + D\mathbf{F}(\mathbf{x}_n)(\mathbf{x}_j - \mathbf{x}_n) = \mathbf{x}_{n+1} + D\mathbf{F}(\mathbf{x}_n)(\mathbf{x}_j - \mathbf{x}_n)\,, \tag{11.49}$$

or

$$\mathbf{x}_{j+1} = \mathbf{A}_n \mathbf{x}_j + \mathbf{b}_n \,, \tag{11.50}$$

where $\mathbf{A}_n = D\mathbf{F}(\mathbf{x}_n)$, and $\mathbf{b}_n = \mathbf{x}_{n+1} + D\mathbf{F}(\mathbf{x}_n)\mathbf{x}_n$. Therefore, we can estimate these parameters (i.e., $\mathbf{A}_n$ and $\mathbf{b}_n$) by minimizing:

$$e^2 = \frac{1}{N_\epsilon} \sum_{\mathbf{x}_j \in \mathcal{B}_\epsilon(\mathbf{x}_n)} \left\|\mathbf{x}_{j+1} - \hat{\mathbf{A}}\mathbf{x}_j - \hat{\mathbf{b}}\right\|^2 \,. \tag{11.51}$$

Solution to this minimization problem can be determined using matrix algebra. First, let us define two matrices containing point inside the corresponding balls:

$$\mathbf{X}_n = [\mathbf{x}_j(1), \mathbf{x}_j(2), \cdots, \mathbf{x}_{j(N_\epsilon)}]\,, \qquad \text{and} \qquad \mathbf{X}_{n+1} = [\mathbf{x}_{j(1)+1}, \mathbf{x}_{j(2)+1}, \cdots, \mathbf{x}_{j(N_\epsilon)+1}]\,. \tag{11.52}$$

Then we are looking for such $\hat{\mathbf{A}}_n$ and $\hat{\mathbf{b}}_n$ that the following is satisfied in the least squares sense:

$$\mathbf{X}_{n+1} = \hat{\mathbf{A}}_n \mathbf{X}_n + \hat{\mathbf{b}}_n = \hat{\mathbf{B}}_n \tilde{\mathbf{X}}_n \,, \tag{11.53}$$

where $\hat{\mathbf{B}}_n = [\hat{\mathbf{A}}_n, \hat{\mathbf{b}}]$ and $\tilde{\mathbf{X}}_n$ is the same matrix as $\mathbf{X}_n$ with additional row of ones at the bottom. Now if we multiply this equation with $\tilde{\mathbf{X}}_n^T$ from the right-hand side, we get:

$$\mathbf{X}_{n+1}\tilde{\mathbf{X}}_n^T = \hat{\mathbf{B}}_n \tilde{\mathbf{X}}_n \tilde{\mathbf{X}}_n^T \,. \tag{11.54}$$

which can be solved as

$$\hat{\mathbf{B}}_n = \mathbf{X}_{n+1}\tilde{\mathbf{X}}_n^T \left(\tilde{\mathbf{X}}_n\tilde{\mathbf{X}}_n^T\right)^{-1} . \tag{11.55}$$

The inverse in the above expression is possible because the auto-corelation matrix $\tilde{\mathbf{X}}_n\tilde{\mathbf{X}}_n^T/N_\epsilon$ is square.

Remarks

1. $\hat{\mathbf{A}}_n$ and $\hat{\mathbf{b}}_n$ define a *local linear map* (which would be globally nonlinear) that approximates the true nonlinear map $\mathbf{F}$ at $\mathbf{x}_n$ point in the reconstructed phase space.

2. That is, if we have a *new* data point $\mathbf{x} \in \mathcal{B}_\epsilon(\mathbf{x}_n)$, we can predict it goes to $\mathbf{y} = \hat{\mathbf{A}}_n\mathbf{x} + \hat{\mathbf{b}}_n$. We will talk more on this later in predictive modeling chapter.

3. It is often better, for Lyapunov exponents estimation purposes, to use higher order fits and then take only the linear piece, which would be $\hat{\mathbf{A}}_n$.

4. If $\mathbf{x}_n$ comes from delay reconstruction,

$$\begin{aligned} \mathbf{x}_n &= \left(x_n, x_{n-\tau}, \ldots, x_{n-(d-1)\tau}\right)^T , \\ \mathbf{x}_{n+k} &= \left(x_{n+k}, x_{n+k-\tau}, \ldots, x_{n+k-(d-1)\tau}\right)^T . \end{aligned} \tag{11.56}$$

   where d is the embedding dimension, τ is integer delay. If $k = \tau$:

$$\begin{aligned} \mathbf{x}_{n+\tau} &= \left(x_{n+\tau}, x_n, \ldots, x_{n+\tau-(d-2)\tau}, x_{n+\tau-(d-1)\tau}\right)^T , \\ &= \left(x_{n+\tau}, \underbrace{x_n}_{\mathbf{x}_n^{(1)}}, \ldots, \underbrace{x_{n-(d-3)\tau}}_{\mathbf{x}_n^{(d-2)}}, \underbrace{x_{n-(d-2)\tau}}_{\mathbf{x}_n^{(d-1)}} \right)^T . \end{aligned} \tag{11.57}$$

   Now we can write

$$\begin{aligned} \mathbf{x}_{n+\tau}^{(1)} &= x_{n+\tau} = f(x_n, x_{n-\tau}, \ldots, x_{n-(d-1)\tau}) = f(\mathbf{x}_n) , \\ \mathbf{x}_{n+\tau}^{(2)} &= \mathbf{x}_n^{(1)} = x_n , \\ \mathbf{x}_{n+\tau}^{(3)} &= \mathbf{x}_n^{(2)} = x_{n-\tau} , \\ &\vdots \\ \mathbf{x}_{n+\tau}^{(d)} &= \mathbf{x}_n^{(d-1)} = x_{n-(d-2)\tau} . \end{aligned} \tag{11.58}$$

   Therefore,

$$\mathbf{x}_{n+\tau} \cong \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1(d-1)} & a_{1d} \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \mathbf{x}_n + \mathbf{b}_n . \tag{11.59}$$

Given an estimate $\hat{\mathbf{A}}^n = \prod_{k=1}^{n} \hat{\mathbf{A}}_k$, we can look at eigenvalues of $\left(\hat{\mathbf{A}}^n\right)^T \left(\hat{\mathbf{A}}^n\right)$. However, there is a problem: the resulting matrix is inherently *ill conditioned* since the eigenvalues are $e^{\lambda_1 n}, e^{\lambda_2 n}, \ldots, e^{\lambda_d n}$, and, therefore $e^{\lambda_1 n}/e^{\lambda_2 n} = e^{(\lambda_1 - \lambda_2)n} \to \infty$ very rapidly for large n. Results of this estimation depend strongly on quality of data and your milage may vary. However, given any perturbation $\mathbf{u}$ defined by

$$\mathbf{u} = \sum_{j=k}^{d} \alpha_j \hat{\mathbf{e}}_j \,, \tag{11.60}$$

where we assume that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$, then

$$\lambda_k = \lim_{n\to\infty} \frac{1}{n} \ln \|D\mathbf{F}^n(\mathbf{x}_0)\mathbf{u}\| \tag{11.61}$$

with probability one. In particular,

$$\lambda_1 = \lim_{n\to\infty} \frac{1}{n} \ln \|D\mathbf{F}^n(\mathbf{x}_0)\mathbf{u}\| = \lim_{n\to\infty} \frac{1}{n} \ln \frac{\|\mathbf{u}_n\|}{\|\mathbf{u}_0\|} \,. \tag{11.62}$$

for *any* $\mathbf{u}_0$ with probability one.

11.3.1 The Largest Shot-Time Lyapunov Exponent Estimate

Similar to the pointwise dimension estimation, Lyapunov exponents can be estimated using either fixed distance or fixed mass methods. For a fixed distance methodology, let $\|\mathbf{u}_0\| \equiv \delta_0$ and $\|\mathbf{u}_n\| = \delta_n$, then Eq. (11.62) says $n\lambda_1 \sim \ln(\delta_n/\delta_0)$, or

$$\delta_n \sim \delta_0 e^{n\lambda_1} \tag{11.63}$$

for n *large*. At the same time, we do not want n *too* large since we expect δ_n to saturate at the scale of the attractor. We can also write:

$$\ln \frac{\delta_n}{\delta_0} \sim n\lambda_1 \,, \tag{11.64}$$

or if we average over many or all data points $\mathbf{x}_0 = \mathbf{x}_k$:

$$\left\langle \ln \frac{\delta_n}{\delta_0} \right\rangle \sim n\lambda_1 \,. \tag{11.65}$$

More specifically, we can estimate the average $\Delta_n = \langle \ln(\delta_n/\delta_0) \rangle$ as:

$$\Delta_n = \frac{1}{N} \sum_{k=1}^{N} \ln \left[\frac{1}{N_\epsilon} \sum_{\mathbf{x}_m \in \mathcal{B}_\epsilon(\mathbf{x}_k)} \|\mathbf{x}_{k+n} - \mathbf{x}_{m+n}\| \right] \,, \tag{11.66}$$

where N_ϵ is the number of points inside the ϵ-ball neighborhood of $\mathbf{x}_k$, $\mathcal{B}_\epsilon(\mathbf{x}_k)$ and the indices are:

k**:** "initial condition" or reference point;

m**:** points inside $\mathcal{B}_\epsilon(\mathbf{x}_k)$;

n**:** number of steps ahead.

Then plotting Δ_n vs n should give us a slope λ_1 per sampling time.

Fixed mass method for estimating Lyapunov exponents fixes the number r of the nearest neighbor points of $\mathbf{x}_k$ and then calculates:

$$\Delta_n = \frac{1}{N}\sum_{k=1}^{N} \ln\left[\frac{1}{r}\sum_{m=1}^{r} \left\|\mathbf{x}_{k+n} - \mathbf{x}_{m(r)+n}\right\|\right], \tag{11.67}$$

or

$$\Delta_n = \frac{1}{N}\sum_{k=1}^{N} \ln\left[\sum_{m=1}^{r} \left\|\mathbf{x}_{k+n} - \mathbf{x}_{m(r)+n}\right\|\right] - \ln r\,, \tag{11.68}$$

where $\{i(m)\}_{m=1}^{r}$ are the indices of all r nearest neighbor points of $\mathbf{x}_k$.

Remarks

1. For a fixed distance method, one should be careful in choosing the ϵ: if it is too small many points might not have any neighbors inside the ϵ-ball neighborhood.

2. For a fixed mass method, r should be chosen conservatively, otherwise the corresponding neighborhood size will be too large for accurate estimation.

3. One might combine the important features of both fixed mass and fixed distance methods using the weighted sum as

$$\Delta_n = \frac{1}{N}\sum_{k=1}^{N} \ln\left[\frac{\sum_{m=1}^{r} w(r)\left\|\mathbf{x}_{k+n} - \mathbf{x}_{m(r)+n}\right\|}{\sum_{m=1}^{r} w(r)}\right], \tag{11.69}$$

   where $w(r)$ is appropriately chosen weighting function that weighs points closest to $\mathbf{x}_k$ more than the ones further from it.

The results of the fixed mass based Lyapunov exponent estimation for our sample signals are shown in Fig. 11.3

Problems

Problem 11.1

Using the same time series of Problem 10.2, find the largest Lyapunov exponent of a candidate chaotic time series. Can you conclude in fact that the system is chaotic?

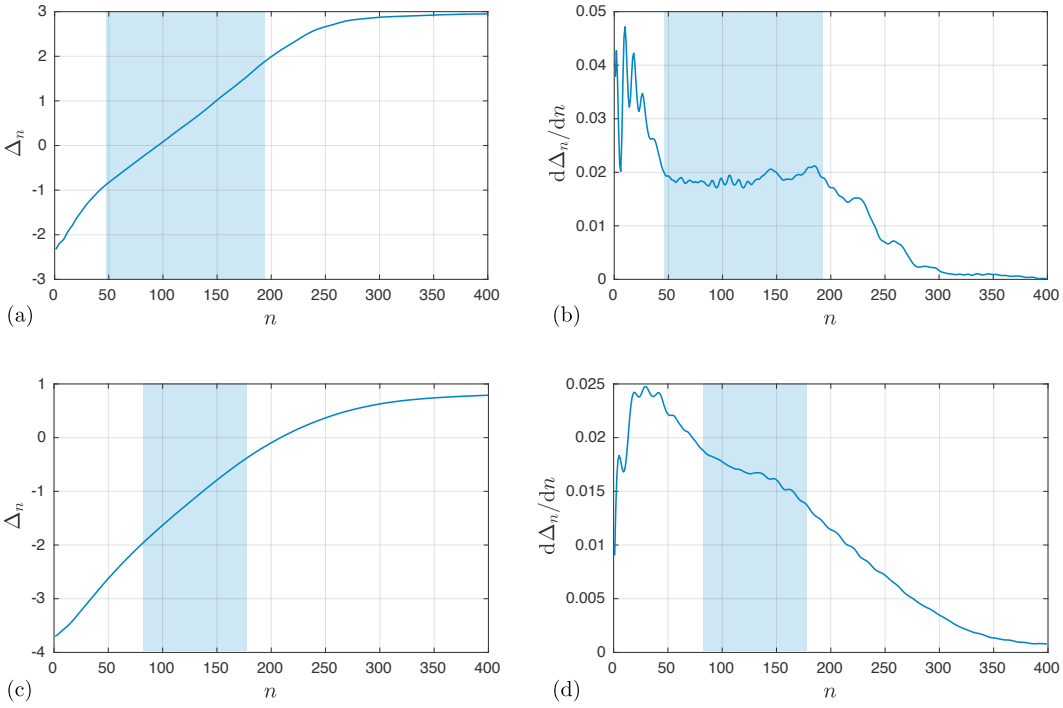

Figure 11.3: Lyapunov exponent estimation for the Lorenz (a–b), and two-well Duffing's (c–d) signals. The shaded areas correspond to the scaling regions.

Chapter 12

Multivariate Analysis

Multivariate analysis is based on the statistical principles of *multivariate statistics*, which involve the process of simultaneously analyzing multiple independent variables using matrix algebra [58, 79]. It has many uses in data and model reduction, blind source signal separation, identification of the modal structure of dynamical systems, identifying important dominant variables in data, etc. Here, we will mainly focus on *proper orthogonal decomposition* (POD) and *smooth orthogonal decomposition* (SOD) as they are applied to the recorded measurements of a *scalar field*. The measurements can be taken from the scalar field using sensors, or obtained by simulating a dynamical system. The recorded measurements or their phase space reconstructions form the data matrices used in the multivariate analysis.

The objective of POD is to obtain the best low-dimensional approximate description of high-dimensional data in the least squares sense [13, 59, 74]. In contrast, SOD is aimed at obtaining the smoothest low-dimensional approximations of high-dimensional dynamical processes [12, 15, 16]. While POD only focuses on the spatial characteristics of the data, SOD considers both its temporal and spatial characteristics.

12.1 Proper Orthogonal Decomposition

The POD (also refered to as *Karhunen-Loéve decomposition* or *principal component analysis*) aims at finding a decomposition of the scalar field $u(\mathbf{x},t)$, $\mathbf{x}\in\Omega\subset\mathbb{R}^n$ and $t\in T\subset\mathbb{R}$, into an orthonormal basis functions $\xi_i(\mathbf{x})$ with time coefficients $q_i(t)$:

$$u(\mathbf{x},t)=\sum_{i=1}^{\infty}p_i(t)\xi_i(\mathbf{x})\,, \tag{12.1}$$

such that, $\xi_1(\mathbf{x})$ maximizes the projection of the field $u(\mathbf{x},t)$ onto itself:

$$\max_{\xi_1}\left[\frac{1}{T}\int_0^T\left(\int_\Omega u(\mathbf{x},t)\xi_1(\mathbf{x})d\mathbf{x}\right)^2 dt\right]; \tag{12.2}$$

$\xi_2(\mathbf{x})$ maximizes the projection of the field into the subspace orthogonal to the first basis function (or its null space), and etc. This is equivalent to following a constrained maximization problem:

$$\max_{\xi_i} \left[\frac{1}{T} \int_0^T \left(\int_\Omega u(\mathbf{x},t)\xi_i(\mathbf{x})d\mathbf{x} \right)^2 dt \right], \quad \text{with} \quad \int_\Omega \xi_i(\mathbf{x})\xi_j(\mathbf{x})d\mathbf{x} = \delta_{ij} \tag{12.3}$$

This itself is reduced to the following integral eigenvalue problem:

$$\int_\Omega R(\mathbf{x},\mathbf{y})\,\xi_k(\mathbf{y})d\mathbf{y} = \sigma_k^2 \xi_k(\mathbf{x}), \tag{12.4}$$

where

$$R(\mathbf{x},\mathbf{y}) = \frac{1}{T}\int_0^T u(\mathbf{x},t)u(\mathbf{y},t)dt, \tag{12.5}$$

is the auto-correlation function of our scalar field. The resultant eigenfunctions $\xi_k(\mathbf{x})$ are called *proper orthogonal modes* (POMs) and the corresponding eigenvalues σ_k^2 are called *proper orthogonal values* (POVs). Now, the corresponding time dependent coefficients or *proper orthogonal coordinates* (POCs) are obtained as:

$$p_i(t) = \int_\Omega u(\mathbf{x},t)\xi_i(\mathbf{x})d\mathbf{x}. \tag{12.6}$$

Remark

The POD assures us that m first POMs are most optimal projections of energy available in the scalar field. However, if the field is contaminated by noise (systematic or measurement) then the POMs would try to maximize the noise content too. In addition, this method is marred with notorious energy leakage problem.

12.2 Smooth Orthogonal Decomposition

The basic idea of SOD is that given the same scalar field $u(\mathbf{x},t)$ we are looking for a linear projection of that field:

$$q(t) = \int_\Omega u(\mathbf{x},t)\psi^T(\mathbf{x})d\mathbf{x}, \tag{12.7}$$

such that the resultant time coordinate $q(t)$ is minimally rough and has maximal variance. The minimal roughness or maximal smoothness can be characterized by the roughness metric that has been extensively used in smoothing splines [23, 75, 76, 81], where the γ-weighed roughness is minimized

$$RF(D^i f) = \int_a^b \gamma(t)(D^i f)(t)^2 dt = \langle (D^i f)(t)^2 \rangle \tag{12.8}$$

over all f functions with i-derivatives defined on the $[a \ldots b]$ time interval, given the natural number i, and the non-negative integrable function γ. Most of the time $i = 2$ is used in splines approximations. However, for our purposes $i = 1$ shows good results and is less noisy when estimated from simulated or experimental data. As for the weighting, we just use $\gamma = 1$, since our function f is empirical and would incorporate γ naturally.

Therefore, the SOD idea translates into the following maximization problem

$$\max_{\psi(\mathbf{x})} \frac{\langle q(t)^2 \rangle}{\langle (D^i q)(t)^2 \rangle} , \tag{12.9}$$

which can be written as constrained maximization problem

$$\max_{\psi(\mathbf{x})} \langle q(t)^2 \rangle , \quad \text{under} \quad \langle (D^i q)(t)^2 \rangle = 1 . \tag{12.10}$$

This in turn translates into the following Euler-Lagrange equation:

$$\delta \langle q(t)^2 \rangle = \lambda^2 \delta \langle (D^i q)(t)^2 \rangle , \tag{12.11}$$

where variations are in terms of $\psi(\mathbf{x})$. We can evaluate the arguments of this Euler-Lagrange equation as

$$\begin{aligned} \langle q(t)^2 \rangle &= \frac{1}{T} \int_0^T \int_\Omega u(\mathbf{x},t)\psi(\mathbf{x}) d\mathbf{x} \int_\Omega u(\mathbf{y},t)\psi(\mathbf{y}) d\mathbf{y}\ dt \\ &= \int_\Omega \int_\Omega \left[\frac{1}{T} \int_0^T u(\mathbf{x},t) u(\mathbf{y},t)\ dt \right] \psi(\mathbf{x}) d\mathbf{x}\ \psi(\mathbf{y}) d\mathbf{y} \\ &= \int_\Omega \int_\Omega R(\mathbf{x},\mathbf{y}) \psi(\mathbf{x}) d\mathbf{x}\ \psi(\mathbf{y}) d\mathbf{y} , \end{aligned} \tag{12.12}$$

and, similarly,

$$\begin{aligned} \langle D^i q(t)^2 \rangle &= \frac{1}{T} \int_0^T \int_\Omega D^i u(\mathbf{x},t)\psi(\mathbf{x}) d\mathbf{x} \int_\Omega D^i u(\mathbf{y},t)\psi(\mathbf{y}) d\mathbf{y}\ dt \\ &= \int_\Omega \int_\Omega \left[\frac{1}{T} \int_0^T D^i u(\mathbf{x},t) D^i u(\mathbf{y},t)\ dt \right] \psi(\mathbf{x}) d\mathbf{x}\ \psi(\mathbf{y}) d\mathbf{y} \\ &= \int_\Omega \int_\Omega D^i R(\mathbf{x},\mathbf{y}) \psi(\mathbf{x}) d\mathbf{x}\ \psi(\mathbf{y}) d\mathbf{y} , \end{aligned} \tag{12.13}$$

where

$$D^i R(\mathbf{x},\mathbf{y}) = \frac{1}{T} \int_0^T D^i u(\mathbf{x},t) D^i u(\mathbf{y},t)\ dt \tag{12.14}$$

is the the i-th derivative of the auto-correlation function, or auto-correlation function of the time derivative of our field. Therefore, the Euler-Lagrange equation is transformed into the generalized integral eigenvalue problem:

$$\int_\Omega R(\mathbf{x},\mathbf{y}) \psi_k(\mathbf{y}) d\mathbf{y} = \lambda_k^2 \int_\Omega D^i R(\mathbf{x},\mathbf{y}) \psi_k(\mathbf{y}) d\mathbf{y} \tag{12.15}$$

The solution to this eigenvalue problem will yield generalized eigenvalues λ_k or *smooth orthogonal values* (SOVs) and corresponding generalized set of eigenfunctions $\psi_k(\mathbf{x})$ or *smooth projective modes* (SPMs). Please note, that SPMs are not required to be orthogonal. Finally, we can reconstruct our field in terms of *smooth orthogonal modes* (SOMs) as

$$u(\mathbf{x},t) = \sum_{i=1}^{\infty} q_i(t) \phi_i(\mathbf{x}) , \tag{12.16}$$

where SOMs $\phi_i(\mathbf{x})$ are bi-orthogonal set of SPMs $\psi_i(\mathbf{x})$

$$\int_\Omega \phi_i(\mathbf{x})\psi_j(\mathbf{x})d\mathbf{x} = \delta_{ij}\,, \tag{12.17}$$

and SOCs $q_i(t)$ also form an orthogonal set of functions:

$$\frac{1}{T}\int_0^T q_i(t)q_j(t)dt = c_i\delta_{ij}\,. \tag{12.18}$$

12.3 Practical Calculation of POD and SOD

In the experimental context we usually sample the field $u(\mathbf{x},t)$ in discrete time intervals t_i $(i = 1,\ldots,m)$ at discrete spacial $\mathbf{x}_j$ $(j = 1,\ldots,n)$. Now if we introduce the following notation $u_j^i = u(\mathbf{x}_j,t_i)$ then we can express the correlation matrix components as

$$R_{ij} = R(\mathbf{x}_i,\mathbf{x}_j) = \frac{1}{m-1}\sum_{k=1}^{N} u_i^k u_j^k\,, \tag{12.19}$$

Now if we define a data matrix $\mathbf{Y}\in\mathbb{R}^{m\times n}$, with components $y_{ij} = u_j^i$, then the corresponding $n\times n$ auto-correlation matrices can be written as

$$\mathbf{R} = \frac{1}{m-1}\mathbf{Y}^T\mathbf{Y}, \quad \text{and} \quad D^k\mathbf{R} = \frac{1}{m-1}(D^k\mathbf{Y})^T D^k\mathbf{Y}. \tag{12.20}$$

If the measurements for $D^k\mathbf{Y}$ are not available, we can use a finite difference based differential operators to estimate $D^k\mathbf{Y}$.

Given the estimate of the auto-correlation matrix, our *discrete generalized eigenvalue problem* for POD is

$$\mathbf{R}\boldsymbol{\xi}_k = \sigma_k^2\boldsymbol{\xi}_k\,, \tag{12.21}$$

where the orthogonal set of $\{\boldsymbol{\phi}_k\}_{k=1}^n$ are POMs and σ_k^2 are the corresponding POVs. In practice, it is easier and more robust to just do the *singular value decomposition* of the data matrix $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{P}\boldsymbol{\Sigma}\boldsymbol{\Xi}^T\,, \tag{12.22}$$

where columns of $\mathbf{P}$ are POCs $\mathbf{p}_k$, $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_k)$ is a diagonal matrix of *singular values* (square of a singular value is a POV), and columns of $\boldsymbol{\Xi}$ are POMs.

For SOD we would need to estimate the corresponding time derivative of the data matrix, and then solve the following *generalized eigenvalue problem*:

$$\mathbf{R}\boldsymbol{\psi}_k = \lambda_k^2(D^i\mathbf{R})\boldsymbol{\psi}_k\,. \tag{12.23}$$

In practice, similarly to POD, it is more expedient and robust to just calculate economic *generalized singular value decomposition* of the matrix pair $\mathbf{Y}$ and $D^k\mathbf{Y}$ as

$$\mathbf{Y} = \mathbf{U}\mathbf{C}\boldsymbol{\Phi}^T\,, \qquad \text{and} \qquad D^k\mathbf{Y} = \mathbf{V}\mathbf{S}\boldsymbol{\Phi}^T\,, \tag{12.24}$$

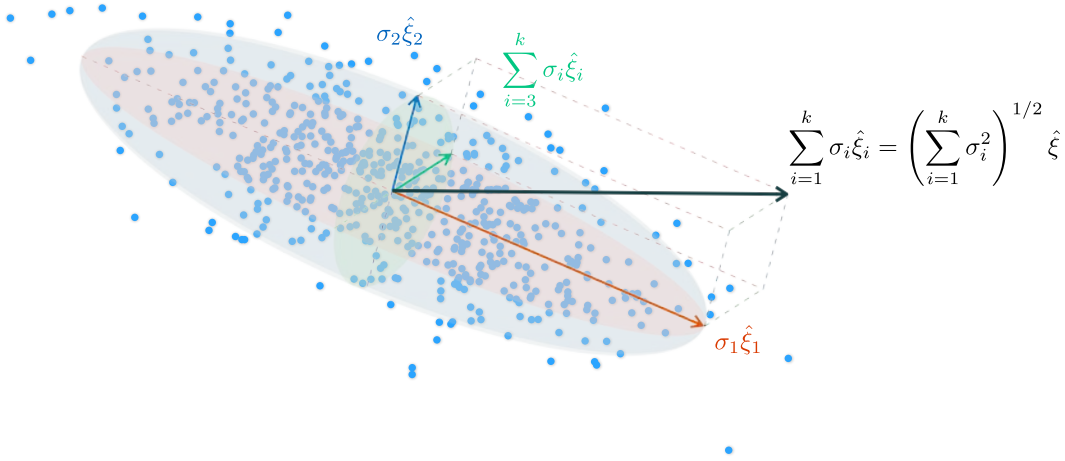

Figure 12.1: Geometric interpretation of POD

where $\mathbf{U} \in \mathbb{R}^{m\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times n}$ are matrices with orthonormal columns, $\mathbf{C} = \text{diag}(c_i) \in \mathbb{R}^{n\times n}$ and $\mathbf{S} = \text{diag}(s_i) \in \mathbb{R}^{n\times n}$ are diagonal matrices, and columns of $\mathbf{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_n] \in \mathbb{R}^{n\times n}$ contain the unit norm SOMs $\boldsymbol{\phi}_i$. SPMs are then given by the columns of

$$\mathbf{\Psi} = \mathbf{\Phi}^{-T}\,, \tag{12.25}$$

and the corresponding SOVs or discrete generalized eigenvalues are given by squaring the following ratio

$$\lambda_i = \frac{c_i}{s_i}\,. \tag{12.26}$$

The corresponding SOCs are orthogonal and form the columns of $\mathbf{Q} = \mathbf{UC} = \mathbf{Y\Psi}$, or

$$\mathbf{q}_i = \mathbf{Y}\boldsymbol{\psi}_i\,, \tag{12.27}$$

and

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{C}^T\mathbf{U}^T\mathbf{UC} = \mathbf{C}^2\,. \tag{12.28}$$

12.4 Geometric Interpretation of POD

To illustrate geometry of POD, we consider a matrix $\mathbf{Y} \in \mathbb{R}^{m\times k}$ containing a cloud of m points generated by our k-dimensional trajectory, such that each column of $\mathbf{Y}$ has zero mean. Then POD can be pictured as fitting a best ellipsoid to this data in the least squares sense as shown in Fig. 12.1. The resultant singular values $\{\sigma_n\}_{n=1}^k$ are the lengths of the semi-axes of the ellipsoid, and the corresponding directions are span by the POMs $\boldsymbol{\xi_n}$. Signal energy or variance in n-th POM is σ_n^2 and the total signal energy is captured by $\sum_{n=1}^k \sigma_n^2$.

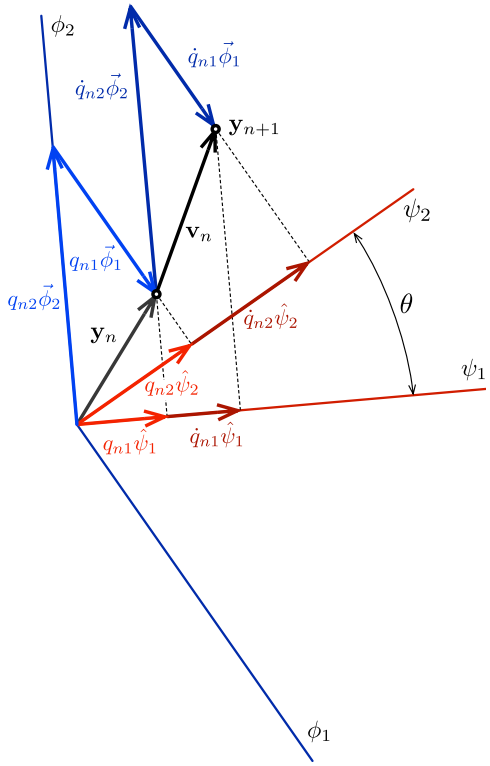

Figure 12.2: Geometric interpretation of smooth orthogonal decomposition

12.5 Geometric Interpretation of SOD:

Now we consider a generic pair of consecutive points $\mathbf{y}_n$ and $\mathbf{y}_{n+1}$ from a scalar field matrix $\mathbf{Y}$ composed of trajectory points sampled with rate of $\Delta t = 1$ as shown in Fig. 12.2. Then the velocity vector for $\mathbf{y}_n$ can be approximated as $\mathbf{v}_n = \mathbf{y}_{n+1} - \mathbf{y}_n$. Now, if we follow the SOD maximization procedure, first we would like to identify a unit vector $\hat{\boldsymbol{\psi}}_1$, such that we maximize the expected ratio of the projections onto this vector $q_{n1} = \mathbf{y}_n^T \hat{\boldsymbol{\psi}}_1$ over $\dot{q}_{n1} = \mathbf{v}_n^T \hat{\boldsymbol{\psi}}_1$. In other words $\hat{\boldsymbol{\psi}}_1$ corresponds to the stationary value of this fraction:

$$\lambda_1^2 = \max_{\hat{\boldsymbol{\psi}}_1} \left\{ \frac{\left\langle \left| \mathbf{y}_n^T \hat{\boldsymbol{\psi}}_1 \right|^2 \right\rangle}{\left\langle \left| \mathbf{v}_n^T \hat{\boldsymbol{\psi}}_1 \right|^2 \right\rangle} \right\} = \max_{\hat{\boldsymbol{\psi}}_1} \left\{ \frac{\langle q_{n1}^2 \rangle}{\langle \dot{q}_{n1}^2 \rangle} \right\}, \tag{12.29}$$

where the resultant stationary value λ_1^2 is the smooth orthogonal value corresponding to the dominant $\hat{\boldsymbol{\psi}}_1$ SOM and $\mathbf{q}_1 = \mathbf{Y}\hat{\boldsymbol{\psi}}_1$ SOC. Once, we identify $\hat{\boldsymbol{\psi}}_1$, we can look for the second $\hat{\boldsymbol{\psi}}_2$ which is linearly independent of $\hat{\boldsymbol{\psi}}_1$ and would yield us the second stationary value or SOV for:

$$\lambda_2^2 = \max_{\hat{\boldsymbol{\psi}}_2 \nparallel \hat{\boldsymbol{\psi}}_1} \left\{ \frac{\left\langle \left| \mathbf{y}_n^T \hat{\boldsymbol{\psi}}_2 \right|^2 \right\rangle}{\left\langle \left| \mathbf{v}_n^T \hat{\boldsymbol{\psi}}_2 \right\rangle \right|^2} \right\} = \max_{\hat{\boldsymbol{\psi}}_2 \nparallel \hat{\boldsymbol{\psi}}_1} \left\{ \frac{\langle q_{n2}^2 \rangle}{\langle \dot{q}_{n2}^2 \rangle} \right\}, \tag{12.30}$$

and so on until we obtain a full set of $\{\hat{\boldsymbol{\psi}}_k\}_{k=1}^n$. Then the corresponding bi-orthogonal set of vectors are obtained by

$$\boldsymbol{\Phi} = \hat{\boldsymbol{\Psi}}^{-T}, \tag{12.31}$$

where $\boldsymbol{\Psi} = \left[\hat{\boldsymbol{\psi}}_1, \hat{\boldsymbol{\psi}}_2, \ldots, \hat{\boldsymbol{\psi}}_n\right] \in \mathbb{R}^{n\times n}$ is SPMs matrix and $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_n] \in \mathbb{R}^{n\times n}$ are SOMs (no longer unit norm) that can be used to describe any point in our matrix $\mathbf{Y}$ as:

$$\mathbf{y}_n = \sum_{i=1}^{k} q_{ni}\boldsymbol{\phi}_i\,, \quad \text{and} \quad \dot{\mathbf{y}}_n = \sum_{i=1}^{k} \dot{q}_{ni}\boldsymbol{\phi}_i\,. \tag{12.32}$$

where $\mathbf{q}_i = \mathbf{Y}\hat{\boldsymbol{\psi}}_i$ or $q_{ni} = \mathbf{y}_n^T\hat{\boldsymbol{\psi}}_i$. On the figure we only show two pairs of basis vectors. As shown, $\boldsymbol{\phi}_1$ is perpendicular to $\hat{\boldsymbol{\psi}}_2$, and $\boldsymbol{\phi}_2$ is perpendicular to $\hat{\boldsymbol{\psi}}_1$. While $\hat{\boldsymbol{\psi}}_n$ are used during the maximization process, $\boldsymbol{\phi}_n$ are actually used to describe our data.

Problems

Problem 12.1

1. Generate a vector valued time series of harmonic functions with different frequency and magnitude contents. Make sure you consider at least seven ($n = 7$) different frequencies and seven different magnitudes. Record the vector valued time series so that it includes at least several full periods of the highest frequency component. As a result you should have a matrix $Y \in \mathbb{R}^{m\times n}$, where $m > n$.

2. Apply POD and SOD to this time series and identify the corresponding time coordinates, modes, and eigenvalues.

3. Interpret the POMs and SOMs and the corresponding modes.

4. Now take the same matrix Y and make sure each column is composed of harmonics that have exactly the same unit magnitude.

5. Redo the POD and SOD analysis. Please explain discrepancies, if any.

Chapter 13

Nonlinear Noise Reduction

Many natural and engineered systems generate nonlinear deterministic time series that are contaminated by random measurement or dynamic noise. Even "clean" time series generated from simulations have inherent digitization noise, which magnitude is at least half of the digital resolution. Generally, one can express the measured time series as:

$$x_n = h(\mathbf{x}_n + \eta_n) + \gamma_n \,, \tag{13.1}$$

where h is a scalar smooth measurement function, η_n random variable is the source of systematic noise and reflects the uncertainty in the underlying dynamic state variable, and γ_n random variable is additive measurement noise. Generally, dealing with measurement noise is less problematic since, in reality, there is actual deterministic trajectory underlying the measurements. In contrast, the dynamic noise can reflect some random variations in system parameters, and, thus, we cannot claim that there is a deterministic attractor for the system. In some cases, the practical effect of both noise sources can be treated as just an additive noise. But, in general, the type and source of noise source will determine the extent of predictability and properties of the system. In the remainder of this chapter we will assume that our measurements are contaminated by the measurement noise and if there is any dynamical noise it can be treated as additive measurement noise.

Chaotic time series have inherently broad spectra and are not amenable to conventional spectral noise filtering. Two main methods for filtering noisy chaotic time series [51, 52, 61] are: (1) model based filtering [6], and (2) projective noise reduction [51]. While model based reduction has some merit for systems with known analytical models [6], projective noise reduction provides a superior alternative [51] for experimental data. There are other methods like adaptive filtering or the wavelet shrinkage [90] method that can work in certain cases but require an opportune selection of parameters or some *a priori* knowledge of the system or the noise mechanisms [33].

In this chapter, we describe projective methods for nonlinear noise reduction that are based on a local subspace identification [16, 30, 51] in the reconstructed phase space [64, 79]. In particular,

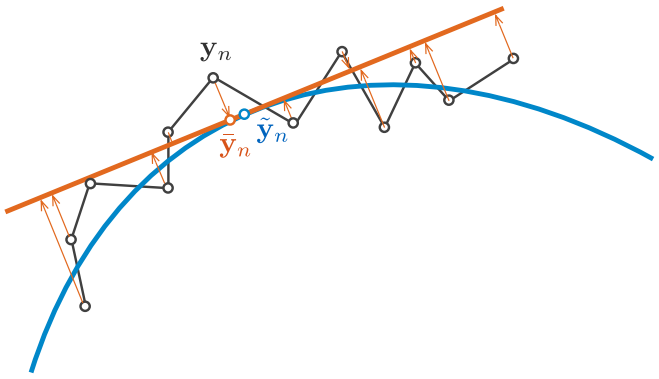

Figure 13.1: Schematic illustration of projective noise reduction idea. The blue curve shows the true phase space trajectory, and the tangent space to it at point $\tilde{\mathbf{y}}_n$ is shown by the red straight line. The noisy measured trajectory is shown by the black dots and line. The noise reduction idea is to project noisy $\mathbf{y}_n$ onto the tangent space to obtain better approximation $\bar{\mathbf{y}}_n \approx \tilde{\mathbf{y}}_n$.

we will describe POD and SOD based projective noise reductions.

13.1 Projective Noise Reduction

Both POD-based local projective noise reduction [51], and SOD-based smooth projective noise reduction [17] work in the reconstructed phase space of the system generating the noisy time series. The basic idea is that, at any point on the attractor, noise leaks out to higher dimensions than the actual local dimension of the attractor's tangent space at that point. Thus, by embedding data into the higher d-dimensional space and then projecting it down to the tangent subspace of q-dimensions reduces the noise in the data as illustrated in Fig. 13.1. The differences between the methods are in the way this tangent space is identified.

For both noise reduction methods, the noisy time series $\{x_i\}_{i=1}^{n}$ is embedded into a d-dimensional global phase space using delay coordinate embedding generating the vector-valued trajectory:

$$\mathbf{y}_i = \left[x_i,\, x_{i+\tau},\, \ldots\, x_{i+(d-1)\tau}\right]^T, \tag{13.2}$$

where τ is the delay time, and we get a set of $\{\mathbf{y}_i\}_{i=1}^{n-(d-1)\tau}$ reconstructed phase space points.

The selection of the global embedding dimension d, and the dimension of the tangent subspace q are important steps in both noise reduction schemes. For a noise-free time series, a method of false nearest neighbors [1, 77] is usually used to estimate the minimum *necessary* embedding dimension D as described in Chapter 9. However, for the experimental time series we want proceed conservatively by embedding in large d and observing results for various q-dimensional projections.

While POD based projective noise reduction works for both map or flow generated time series,

SOD based noise reduction is only applicable to continuous flow generated data. This is due to the fact that, for the SOD, we need to estimate data-based time derivative of the time series, which would be problematic for the map generated data. In addition, the described SOD method can also be modified to use POD instead of SOD when dealing with the continuous time series to obtain the energy based subspace instead of smooth subspace for noise reduction.

13.1.1 POD Based Local Projective Noise Reduction

Standard methodology to approximate nonlinear map governing the evolution of a deterministic signal is to use local linear maps. Assuming the original map $\mathbf{F}(\mathbf{y})$ is a smooth nonlinear function of the reconstructed coordinates, we can linearize it about a point of interest:

$$\mathbf{F}(\mathbf{y}) \approx \mathbf{F}(\mathbf{y}_i) + D\mathbf{F}(\mathbf{y}_i)(\mathbf{y} - \mathbf{y}_i) = \mathbf{y}_{i+1} + D\mathbf{F}(\mathbf{y}_i)(\mathbf{y} - \mathbf{y}_i)\,, \tag{13.3}$$

or

$$\mathbf{y}_{i+1} = \mathbf{A}_i \mathbf{y}_{i+1} + \mathbf{b}_i\,, \tag{13.4}$$

where $\mathbf{A}_i$ is the Jacobian $D\mathbf{F}(\mathbf{y}_i)$. Usually $\mathbf{A}_i$ is determined by a least squares fit to experimental data.

The basic idea is to adjust $\mathbf{y}_i$ by projecting it down to a subspace span by the dominant eigenvectors of $\mathbf{A}_i$ which should approximate the tangent space to the actual dynamics at time i as

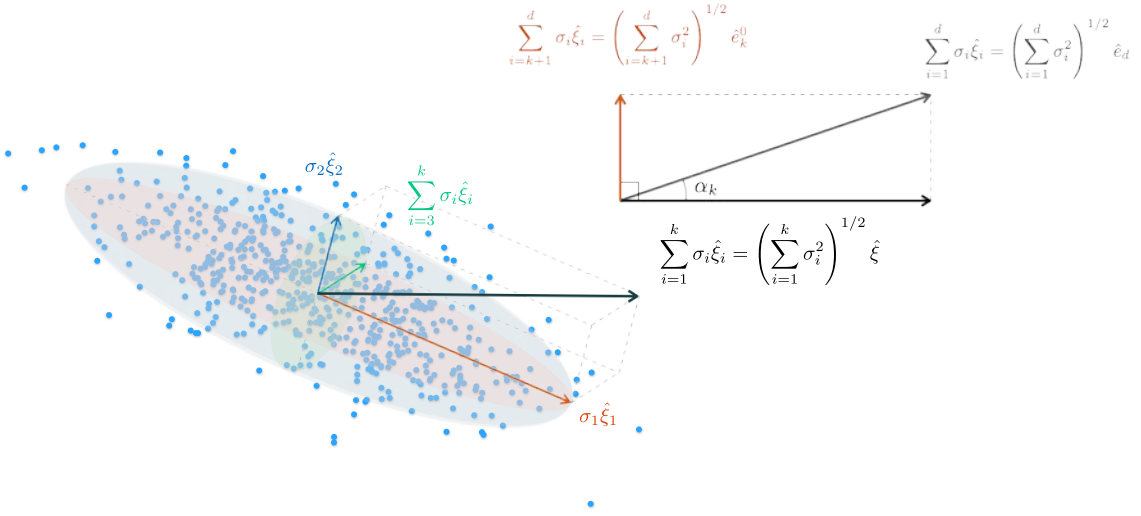

Figure 13.2: Schematic of the local projective noise reduction. Consider a cloud of points on a k-dimensional hypersurface which are corrupted by noise. The noise spreads the points above and below the hypersurface so that in d-dimensional space they form a d-dimensional object of small but finite thickness. We can approximate the distribution of points by an ellipsoid and thus identify the direction of smallest extent. Then we project the points onto the subspace orthogonal to it.

illustrated in Fig. 13.2. In the local projective noise algorithm [48], for each point $\mathbf{y}_i$ in the reconstructed d-dimensional phase space r temporarily uncorrelated nearest neighbor points $\left\{\mathbf{y}_i^{(j)}\right\}_{j=1}^{r}$ are determined. This is usually done utilizing fast nearest neighbor search algorithms such a kd-tree based searches as was the case for the false nearest neighbor and fixed mass based algorithms. The POD is applied to this cloud of nearest neighbor points, and the optimal q-dimensional projection of the cloud is obtained providing the needed filtered $\bar{\mathbf{y}}_i$ (which is also adjusted to account for the shift in the projection due to the trajectory curvature at $\mathbf{y}_i$ [52]). The filtered time series $\{\bar{x}_i\}_{i=1}^{n}$ is obtained by averaging the appropriate coordinates in the adjusted vector-valued time series $\{\bar{\mathbf{y}}_i\}$.

POD Subspace Identification and Data Projection

POD is solved using *singular value decomposition* of a matrix $\mathbf{Y}_i = \left[\mathbf{y}_i^{(1)}, \mathbf{y}_i^{(2)}, \ldots, \mathbf{y}_i^{(r)}\right] \in \mathbb{R}^{d\times r}$ containing the r nearest neighbors of point $\mathbf{y}_i$:

$$\mathbf{Y}_i^T = \mathbf{P}_i \mathbf{\Sigma}_i \mathbf{\Xi}_i^T \,, \tag{13.5}$$

where POMs are given by the columns of $\mathbf{\Xi}_i \in \mathbb{R}^{d\times d}$, POCs are given by the columns of $\mathbf{P}_i \in \mathbb{R}^{r\times d}$ and SOVs are diagonal terms of $\mathbf{\Sigma}_i$ squared. We will assume that the POVs are arranged in descending order ($\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_d^2$).

The most energy in q-dimensional approximation to $\mathbf{Y}$ ($q < d$) can be obtained by retaining only q columns of interest in the matrices $\mathbf{P}$ and $\mathbf{\Xi}_i$, and reducing $\mathbf{\Sigma}_i$ to the corresponding $q \times q$ minor. In effect, we are looking for a projection of $\mathbf{Y}_i$ onto a q-dimensional most energetic subspace. The embedding of this q-dimensional projection into d-dimensional space ($\bar{\mathbf{Y}}_i$) is constructed using the corresponding reduced matrices: $\bar{\mathbf{P}}_i \in \mathbb{R}^{n\times q}$, $\bar{\mathbf{\Sigma}}_i \in \mathbb{R}^{q\times q}$, and $\bar{\mathbf{\Xi}}_i \in \mathbb{R}^{d\times q}$, as follows

$$\bar{\mathbf{Y}}_i^T = \bar{\mathbf{P}}_i \bar{\mathbf{\Sigma}}_i \bar{\mathbf{\Xi}}_i^T \,. \tag{13.6}$$

Adjusting for Bias due to Curvature

The above approximation would be acceptable if our true clean trajectory passed through the center of mass of the cloud of nearest neighbors. However, due to the inherent trajectory curvature, the clean trajectory would pass through a point shifted outwards from the center of mass. Therefore, the identified POD subspaces will not be tangent to the actual trajectory's manifold, they would actually intersect it. Thus, we need to adjust our linear projection subspace towards the point of our interest by eliminating the shift caused by the curvature. As suggested by Kantz and Schreiber [53] one can do so by moving the projection as $2\bar{\mathbf{y}}_n - \bar{\bar{\mathbf{y}}}_n$, where $\bar{\bar{\mathbf{y}}}_n = (\bar{\mathbf{y}}_{n-1} + \bar{\mathbf{y}}_{n+1})/2$ is the average of the neighborhood mass centers as illustrated in Fig. 13.3.

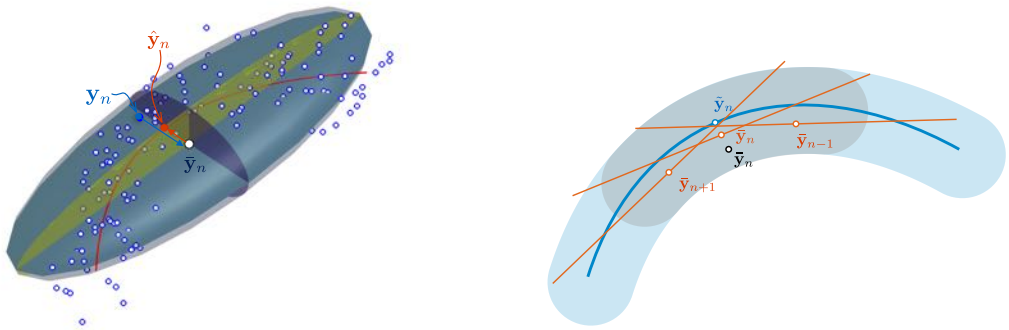

Figure 13.3: Consider a cloud of noisy points $\mathbf{y}_k$ around a one dimensional curve going through our noise free point of interest $\mathbf{y}_n$. The local linear approximations are not tangents to the actual trajectory and the centre of mass ($\bar{\mathbf{y}}$) of the neighborhood and is shifted inward with respect to the curvature. We can adjust for it by shifting the centre of mass $\bar{\mathbf{y}}_n$ outward with respect to the curvature by $2\bar{\mathbf{y}}_n - \bar{\bar{\mathbf{y}}}_n$.

13.1.2 SOD Based Noise Reduction

In contrast, smooth projective noise reduction works with short strands of the reconstructed phase space trajectory. Therefore, this method can only be applicable for data coming from a continuous flow and not a map. These strands are composed of $(2l+1)$ time-consecutive reconstructed phase space points, where $l > d$ is a small natural number. Namely, each reconstructed point $\mathbf{y}_k$ has an associated strand $\mathbf{S}_k = [\mathbf{y}_{k-l}, \ldots, \mathbf{y}_{k+l}]$, and an associated bundle of r nearest neighbor strands $\{\mathbf{S}_k^j\}_{j=1}^r$, including the original. This bundle is formed by finding $(r-1)$ nearest neighbor points $\{\mathbf{y}_k^j\}_{j=1}^{r-1}$ for $\mathbf{y}_k$ and the corresponding strands. SOD is applied to each of these strands in the bundle and the corresponding q-dimensional smoothest approximations to the strands $\{\tilde{\mathbf{S}}_k^j\}_{j=1}^r$ are obtained. The filtered $\bar{\mathbf{y}}_k$ point is determined by the weighted average of points $\{\tilde{\mathbf{y}}_k^j\}_{j=1}^r$ in the smoothed strands of the bundle. Finally, the filtered time series $\{\bar{x}_i\}_{i=1}^n$ are obtained just as in the local projective noise reduction. The procedure can be applied repeatedly for further smoothing or filtering.

13.1.3 Smooth Subspace Identification and Data Projection

SOD is solved using a *generalized singular value decomposition* of the matrix pair $\mathbf{S}_i$ and $\dot{\mathbf{S}}_i$:

$$\mathbf{S}_i^T = \mathbf{U}_i \mathbf{C}_i \mathbf{\Phi}_i^T, \quad \dot{\mathbf{S}}_i^T = \mathbf{V}_i \mathbf{G}_i \mathbf{\Phi}_i^T, \quad \mathbf{C}_i^T \mathbf{C}_i + \mathbf{G}_i^T \mathbf{G}_i = \mathbf{I}, \tag{13.7}$$

where SOMs are given by the columns of $\mathbf{\Phi}_i \in \mathbb{R}^{d\times d}$, SOCs are given by the columns of $\mathbf{Q}_i = \mathbf{U}_i \mathbf{C}_i \in \mathbb{R}^{n\times d}$ and SOVs are given by the term-by-term division of $\mathrm{diag}(\mathbf{C}_i^T \mathbf{C}_i)$ and $\mathrm{diag}(\mathbf{G}_i^T \mathbf{G}_i)$. The resulting SPMs are the columns of the inverse of the transpose of SOMs: $\mathbf{\Psi}_i^{-1} = \mathbf{\Phi}_i^T \in \mathbb{R}^{d\times d}$. In this paper, we will assume that the SOVs are arranged in descending order $(\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d)$.

The magnitude of the SOVs quadratically correlates with the smoothness of the SOCs [16].

The smoothest q-dimensional approximation to $\mathbf{S}_i$ ($q < d$) can be obtained by retaining only q columns of interest in the matrices $\mathbf{U}_i$ and $\mathbf{\Phi}_i$, and reducing $\mathbf{C}_i$ to the corresponding $q \times q$ minor. In effect, we are looking for a projection of $\mathbf{S}_i$ onto a q-dimensional smoothest subspace. The embedding of this q-dimensional projection into d-dimensional space ($\bar{\mathbf{S}}_i$) is constructed using the corresponding reduced matrices: $\bar{\mathbf{U}}_i \in \mathbb{R}^{n\times k}$, $\bar{\mathbf{C}}_i \in \mathbb{R}^{k\times k}$, and $\bar{\mathbf{\Phi}}_i \in \mathbb{R}^{d\times k}$, as follows

$$\bar{\mathbf{S}}_i^T = \bar{\mathbf{U}}_i \bar{\mathbf{C}}_i \bar{\mathbf{\Phi}}_i^T. \tag{13.8}$$

13.1.4 Data Padding to Mitigate the Edge Effects

SOD-based noise reduction is working on $(2l+1)$-long trajectory strands. Therefore, for the first and last l points in the matrix $\mathbf{Y}$, we will not have full-length strands for calculations. In addition, the delay coordinate embedding procedure reconstructs the original n-long time series into $\mathbf{Y}$ which has only $[n-(d-1)\tau]$-long rows. Therefore, the first and last $(d-1)\tau$ points in $\{x_i\}_{i=1}^n$ will not have all d components represented in $\mathbf{Y}$, while other points will have their counterparts in each column of $\mathbf{Y}$. To deal with these truncations, which cause unwanted edge effects in noise reduction, we pad both ends of $\mathbf{Y}$ by $[l+(d-1)\tau]$-long trajectory segments. These trajectory segments are identified by finding the nearest neighbor points to the starting and the end points inside $\mathbf{Y}$ itself (e.g., $\mathbf{y}_s$ and $\mathbf{y}_e$, respectively). Then, the corresponding trajectory segments are extracted from $\mathbf{Y}$: $\mathbf{Y}_s = [\mathbf{y}_{s-l-(d-1)\tau}, \ldots, \mathbf{y}_{s-1}]$ and $\mathbf{Y}_e = [\mathbf{y}_{e+1}, \ldots, \mathbf{y}_{e+l+(d-1)\tau}]$, and are used to form a padded matrix $\widehat{\mathbf{Y}} = [\mathbf{Y}_s, \mathbf{Y}, \mathbf{Y}_e]$. Now, the procedure can be applied to all points in $\widehat{\mathbf{Y}}$ starting at $l+1$ and ending at $n+(d-1)\tau+l$ points.

13.2 Smooth Noise Reduction Algorithm

Smooth noise reduction algorithm is schematically illustrated in Fig. 13.4. While we usually work with data in column form, in this figure—for illustrative purpose—we arranged arrays in row form. The particular details are explained as follows:

1. Delay Coordinate Embedding

   (a) Estimate the appropriate delay time τ and the minimum *necessary* embedding dimension D for the time series $\{x_i\}_{i=1}^n$.

   (b) Determine the local (i.e., projection), $q \le D$, and the global (i.e., embedding), $d \ge D$, dimensions for trajectory strands.

   (c) Embed the time series $\{x_i\}_{i=1}^n$ using global embedding parameters (τ, d) into the reconstructed phase space trajectory $\mathbf{Y} \in \mathbb{R}^{d\times(n-(d-1)\tau)}$.

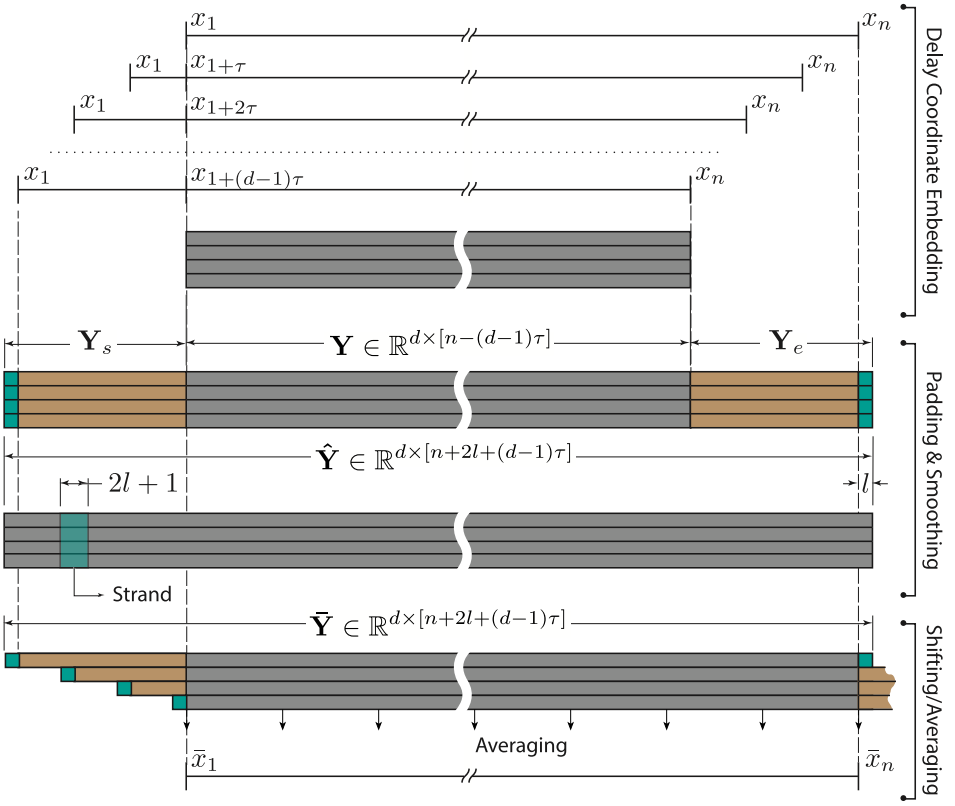

Figure 13.4: Schematic illustration of smooth projective noise reduction algorithm

(d) Partition the embedded points in $\mathbf{Y}$ into a kd-tree for fast searching.

2. Padding and Filtering:

(a) Pad the end points of $\mathbf{Y}$ to have appropriate strands for the edge points, which results in $\widehat{\mathbf{Y}} \in \mathbb{R}^{d\times[n+2l+(d-1)\tau]}$. In addition, initialize a zero matrix $\bar{\mathbf{Y}} \in \mathbb{R}^{d\times[n+2l+(d-1)\tau]}$ to hold the smoothed trajectory.

(b) For each point $\{\hat{\mathbf{y}}_i\}_{i=l+1}^{n+(d-1)\tau+l}$ construct a $(2l+1)$-long trajectory strand $\mathbf{S}_i^1 = [\hat{\mathbf{y}}_{i-l}, \ldots, \hat{\mathbf{y}}_{i+l}] \in \mathbb{R}^{d\times(2l+1)}$. The next step is optional intended for dense but noisy trajectories only.

(c) If $r > 1$, for the same point $\hat{\mathbf{y}}_i$, look up $(r-1)$ nearest neighbor points that are temporarily uncorrelated, and the corresponding nearest neighbor strands $\{\mathbf{S}_i^j\}_{j=2}^r$.

(d) Apply SOD to each of the r strands in this bundle, and obtain the corresponding q-dimensional smooth approximations to all d-dimensional strands in the bundle $\{\tilde{\mathbf{S}}^j\}_{j=1}^r$.

(e) Approximate the base strand by taking the weighted average of all these smoothed strands $\bar{\mathbf{S}}_i = \langle \tilde{\mathbf{S}}_i^j \rangle_j$, using weighting that diminishes contribution with the increase in the distance from the base strand.

(f) Update the section corresponding to $\hat{\mathbf{y}}_i$ point in the filtered trajectory matrix as

$$\bar{\mathbf{Y}}_i = \bar{\mathbf{Y}}_i + \bar{\mathbf{S}}_i \mathbf{W}\,, \tag{13.9}$$

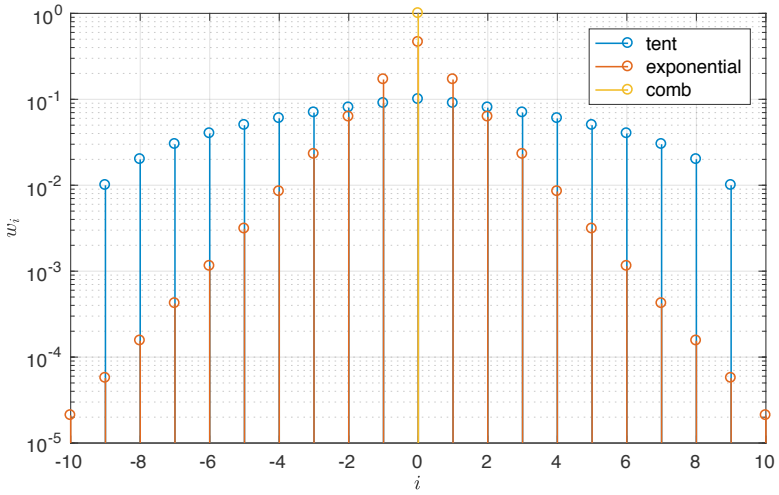

Figure 13.5: Sample weighting functions for $l = 10$ size strand

where $\bar{\mathbf{Y}}_i$ is composed of $(i-l)$-th through $(i+l)$-th columns of $\bar{\mathbf{Y}}$:

$$\bar{\mathbf{Y}}_i = [\bar{\mathbf{y}}_{i-l}, \ldots, \bar{\mathbf{y}}_i, \ldots, \bar{\mathbf{y}}_{i+l}] \,, \tag{13.10}$$

and $\mathbf{W} = \text{diag}(w_i)$ $(i = -l, \ldots, l)$ is the appropriate weighting along the strand with the peak value at the center as illustrated in Fig. 13.5.

3. Shifting and Averaging:

   (a) Replace the points $\{\hat{\mathbf{y}}_i\}_{i=1+l}^{n+(d-1)\tau+l}$ at the center of each base strand by its filtered approximation $\bar{\mathbf{y}}_i$ determined above to form $\bar{\mathbf{Y}} \in \mathbb{R}^{d\times[n+2l+(d-1)\tau]}$.

   (b) Average each d smooth adjustment to each point in the time series to estimate the filtered point:

$$\bar{x}_i = \frac{1}{d}\sum_{k=1}^{d} \bar{Y}(k, i+l+(k-1)\tau) \,.$$

4. Repeat the first three steps until data is smoothed out, or some preset criterion is met.

13.3 Examples of the Nonlinear Noise Reduction

In this section we evaluate the performance of the algorithm by testing it on a time series generated by Lorenz model [87] and a double-well Duffing oscillator [62]. The Lorenz model used to generate chaotic time series is:

$$\dot{x} = -\frac{8}{3}x + yz \,, \quad \dot{y} = -10\,(y-z) \,, \quad \text{and} \quad \dot{z} = -xy + 28y - z \,. \tag{13.11}$$

The chaotic signal used in the evaluation is obtained using the following initial conditions $(x_0, y_0, z_0) = (20, 5, -5)$, and the total of 60,000 points were recorded using 0.01 sampling time period. The

double-well Duffing equation used is:

$$\ddot{x} + 0.25\,\dot{x} - x + x^3 = 0.3\cos t\,. \tag{13.12}$$

The steady state chaotic response of this system was sampled thirty times per forcing period and a total of 60,000 points were recorded to test the noise reduction algorithms.

In addition, a 60,000 point, normally distributed, random signal was generated, and was mixed with the chaotic signals in 1/20, 1/10, 1/5, 2/5, and 4/5 standard deviation ratios, which respectively corresponds to 5, 10, 20, 40, and 80 % noise in the signal or SNRs of 26.02, 20, 13.98, 7.96, and 1.94 dB. This results in total of five noisy time series for each of the models. The POD-based projective and SOD-based smooth noise reduction procedures were applied to all noise signals using total of ten iterations.

To evaluate the performance of the algorithms, we used several metrics that include improvements in SNRs and power spectrum, estimates of the correlation sum and short-time trajectory divergence rates (used for estimating the correlation dimension and short-time largest Lyapunov exponent, respectively [47]). In addition, phase portraits were examined to gain qualitative appreciation of noise reduction effects.

13.4 Results and Discussion

13.4.1 Lorenz Model Based Time Series

Using average mutual information estimated from the clean Lorenz time series, a delay time of 12 sampling time periods is determined. The false nearest neighbors algorithm (see Fig. 13.6) indicates that the attractor is embedded in $D = 3$ dimensions for the noise-free time series. Even for the noisy time series the trends show clear qualitative change around four or five dimensions. Therefore, six is

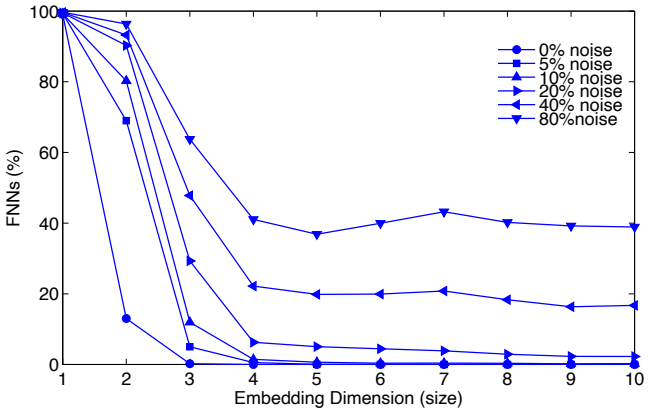

Figure 13.6: False nearest neighbor algorithm results for Lorenz time series

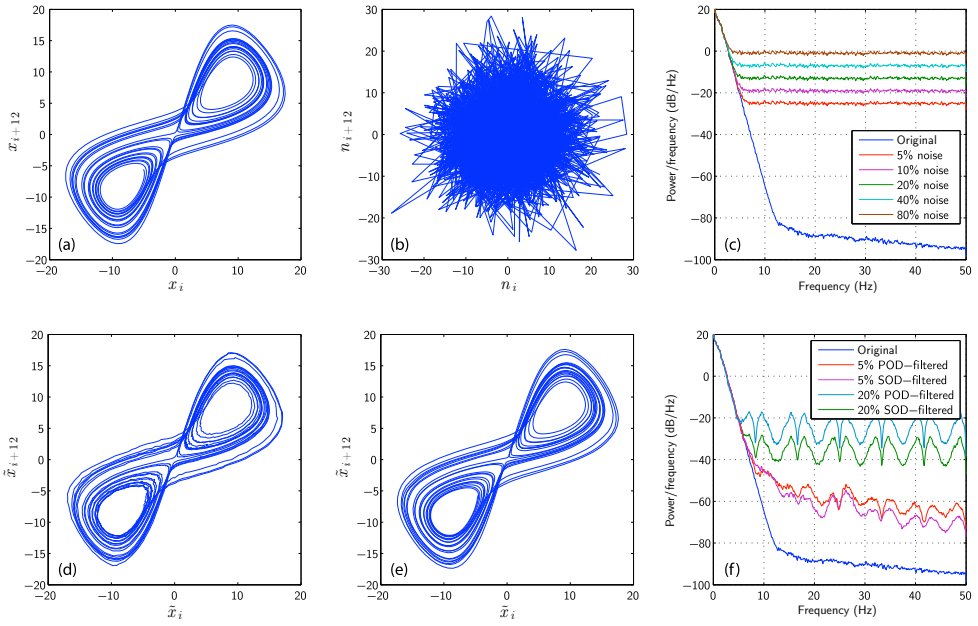

Figure 13.7: Phase portrait from Lorenz signal (a); random noise phase portrait (b); power spectrum of clean and noisy signals (c); POD-filtered phase portrait (d) and SOD-filtered phase portrait (e) of 5% noise added data; and the corresponding power spectrums (f).

used for global embedding dimension ($d = 6$), and $k = 3$ is used as local projection dimension. The reconstructed phase portraits for the clean Lorenz signal and the added random noise are shown in Fig. 13.7 (a) and (b), respectively. The corresponding power spectral densities for the clean and the noise-added signals are shown in Fig. 13.7 (c). The phase portraits of POD- and SOD-filtered signals with 5% noise are shown in Fig. 13.7 (d) and (e), respectively. Fig. 13.7 (f) shows the corresponding power spectral densities. In all these and the following figures, 64 nearest neighbor points are used for POD-based algorithm, and bundles of 9 eleven-point-long trajectory strands are used for SOD-based algorithm. The filtered data shown is for 10 successive applications of the noise reduction algorithms. The decrease in the noise floor after filtering is considerably more dramatic for the SOD algorithm when compared to the POD.

The successive improvements in SNRs after each iteration of the algorithms are shown in Fig. 13.8, and the corresponding numerical values are listed in Table 13.1 for the 5% and 20% noise added signals. While the SNRs are monotonically increasing for the SOD algorithm after each successive application, they peak and then gradually decrease for the POD algorithm. In addition, the rate of increase is considerably larger for the SOD algorithm compared to the POD, especially during the initial applications. As seen from the Table 13.1, the SOD-based algorithm provides about 13 dB improvement in SNRs, while POD-based algorithm manages only $7 \sim 8$ dB improvements.

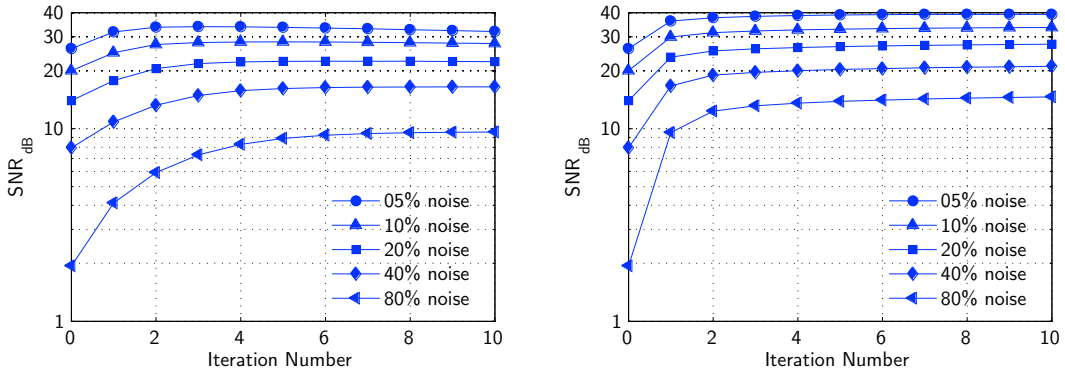

Figure 13.8: Noisy Lorenz time series SNRs versus number of application of POD-based (left) and SOD-based (right) nonlinear noise reduction procedures

The estimates of the correlation sum and short-time trajectory divergence for the noise-reduced data and the original clean data are shown in Fig. 13.9. For the correlation sum, both POD- and SOD-based algorithms show similar scaling regions and improve considerably on the noisy data. For the short-time divergence both methods manage to recover some of the scaling regions, but SOD does a better job at small scales, where POD algorithm yields large local fluctuations in the estimates.

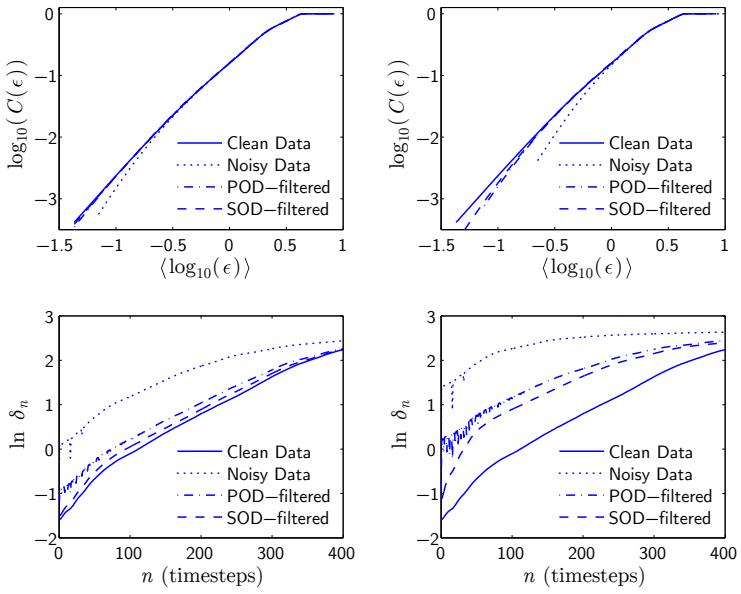

Figure 13.9: Correlation sum (top plots) and short-time divergence (bottom plots) for 5% (left plots) and 20% (right plots) noise contaminated Lorenz time series

The qualitative comparison of the phase portraits for both noise reduction methods are shown in Fig. 13.10. While POD does a decent job at low noise levels, it fails at higher noise levels where

Table 13.1: SNRs for the noisy time series from Lorenz model, and for noisy data filtered by POD- and SOD-based algorithms

Signal	Lorenz + 5% Noise			Lorenz + 20% Noise		
Algorithm	None	POD	SOD	None	POD	SOD
SNR (dB)	26.0206	33.8305	39.1658	13.9794	22.3897	27.4318

no deterministic structure is recovered at 80% noise level. In contrast, SOD provides smoother and cleaner phase portraits. Even at 80% noise level, the SOD algorithm is able to recover large deterministic structures in the data. While SOD still misses small scale features at 80% noise level, topological features are still similar to the original phase portrait in Fig. 13.7(a).

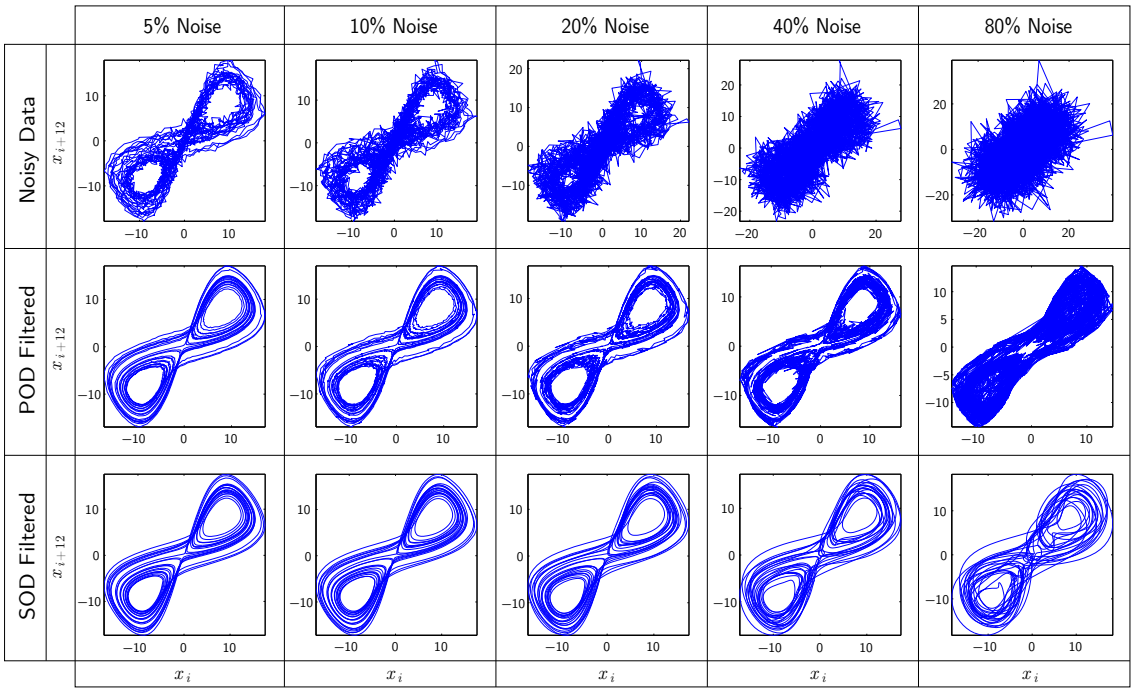

Figure 13.10: Phase space reconstructions for noisy and noise-reduced Lorenz time series after ten iterations of the POD- and SOD-based algorithms

13.4.2 Duffing Equation Based Time Series

Using the noise-free Duffing time series and the average mutual information, a delay time of seven sampling time periods is determined. In addition, false nearest neighbors algorithm indicates that the attractor is embedded in $D = 4$ dimensions for 0 % noise (see Fig. 13.11). Six is used for global embedding dimension ($d = 6$), and $k = 3$ is used as local projection dimension. The reconstructed

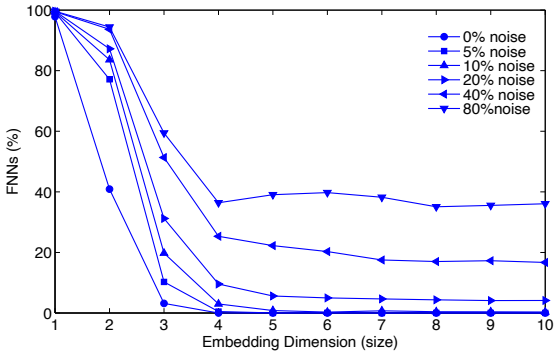

Figure 13.11: False nearest neighbor algorithm results for Duffing time series

phase portraits for the clean Duffing signal and the added random noise are shown in Fig. 13.12 (a) and (b), respectively. The corresponding power spectral densities for the clean and noise-added signals are shown in Fig. 13.12 (c). The phase portraits of POD- and SOD-filtered signals with 5% noise are shown in Fig. 13.12 (d) and (e), respectively. Fig. 13.12 (f) shows the corresponding power spectral densities. In all these and the following figures, 64 nearest neighbor points were used for POD-based algorithm, and bundles of nine 11-point-long trajectory strands were for SOD-based algorithm. The filtered data shown is after 10 successive applications of the noise reduction algorithms. As before, the decrease in the noise floor after filtering is considerably more dramatic for the SOD algorithm when compared to the POD.

The successive improvements in SNRs after each iteration of the algorithms are shown in Fig. 13.13,

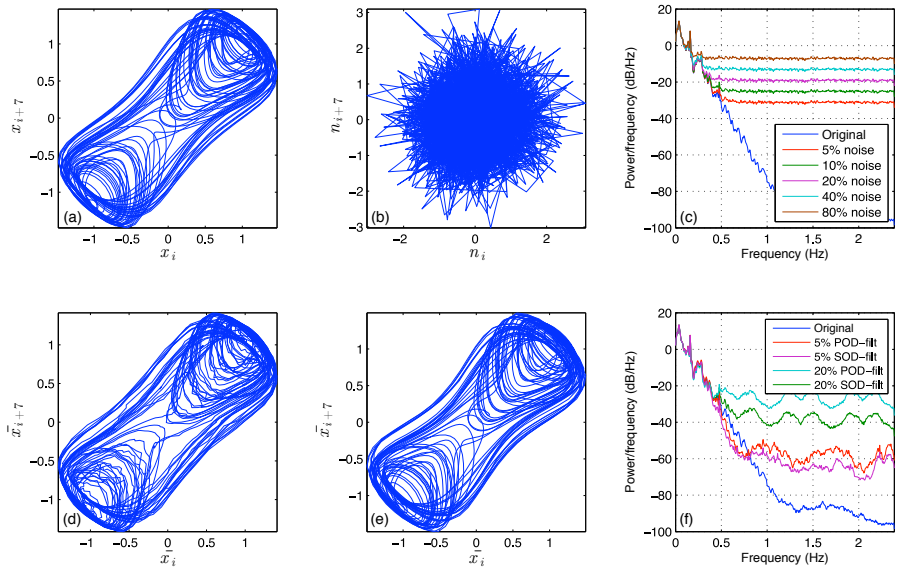

Figure 13.12: Reconstructed phase portrait from Duffing signal (a); random noise phase portrait (b); power spectrum of clean and noisy signals (c); POD-filtered phase portrait of 5% noise added (d); SOD-filtered phase portrait of 5% noise added (e); and the corresponding power spectrums (f).

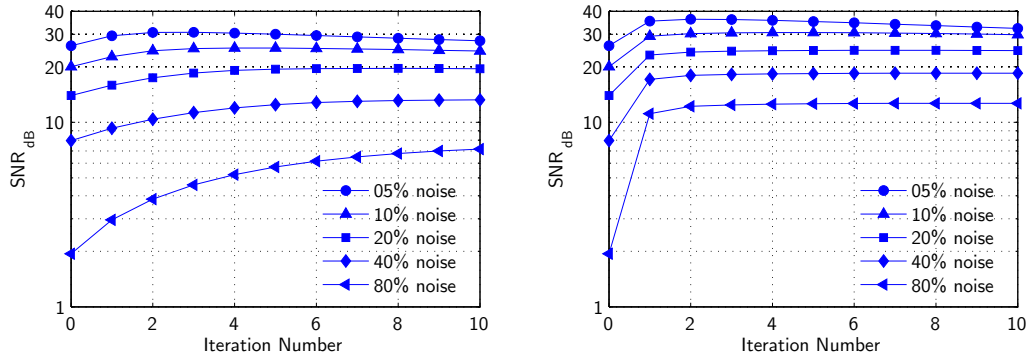

Figure 13.13: Noisy Duffing time series SNRs versus number of application of POD-based (left) and SOD-based (right) nonlinear noise reduction procedures

and the corresponding numerical values are listed in Table 13.2 for 5% and 20% noise added signals. Here, SNRs for both methods are peaking at some point and then gradually decrease. In addition, the rate of increase is considerably larger for the SOD compared to the POD algorithm, especially during the initial iterations. SOD is usually able to increase SNR by about 10 dB, while POD provides only about 4 dB increase on average.

Table 13.2: SNRs for the noisy time series from Duffing model, and for noisy data filtered by POD- and SOD-based algorithms

Signal	Duffing + 5% Noise			Duffing + 20% Noise		
Filtering	None	POD	SOD	None	POD	SOD
SNR (dB)	26.0206	30.7583	36.1687	13.9794	19.5963	24.5692

The estimates of the correlation sum and short-time trajectory divergence for the noisy, noise reduced data, and the original clean data from Duffing oscillator are shown in Fig. 13.14. For the correlation sum, both POD- and SOD-based algorithms show similar scaling regions and improve substantially on the noisy data, with SOD following the noise trend closer than POD. For the short-time divergence rates both methods manage to recover some of the scaling regions, but SOD does a better job at small scales, where POD algorithm causes large local fluctuations in the estimates.

The qualitative comparison of the Duffing phase portraits for both noise reduction methods are shown in Fig. 13.15. For consistency all noise-reduced phase portraits are obtained after 10 consecutive applications of the algorithms. While POD does a decent job at low noise levels, it fails at high noise levels where no deterministic structure is recovered at 40% or 80% noise levels. In contrast, SOD provides smoother and cleaner phase portraits at every noise level. Even at 80% noise level, the SOD algorithm is able to recover some large deterministic structures in the data,

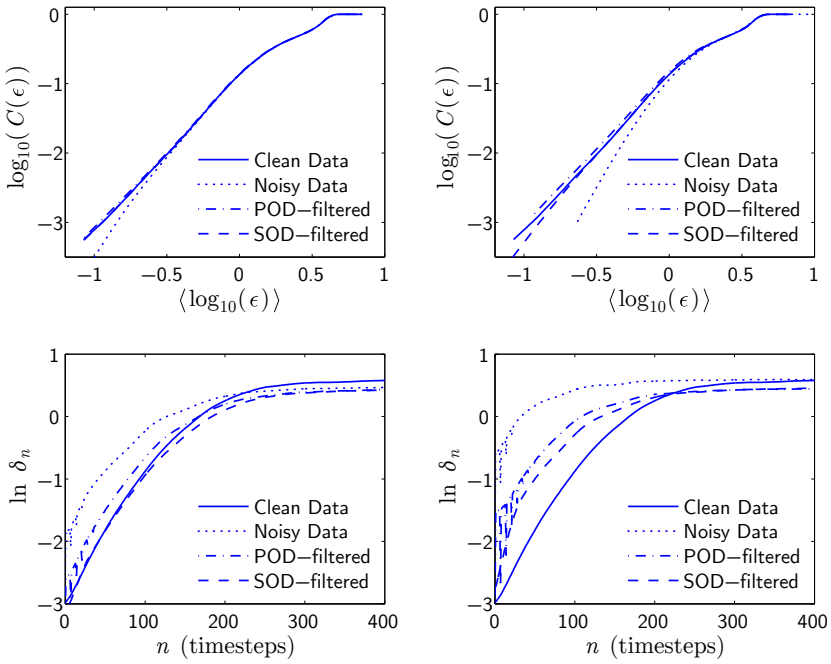

Figure 13.14: Correlation sum (top plots) and short-time divergence (bottom plots) for 5% (left plots) and 20% (right plots) noise contaminated Duffing time series

providing topologically similar phase portrait.

In summary, the decrease in the noise floor observed in the power spectral density for the SOD-based method was at least 20 dB larger then for the POD-based method, which itself improved on the original signals noise floor at best by 10 dB. SNRs show similar trends, where SOD-based method manages to improve it by about 10 dB, while POD-based manages only $4 \sim 5$ dB increase. In addition, SOD accomplishes this with considerably fewer applications of the noise reduction scheme, while POD works more gradually and requires many more iterations at higher noise levels.

Estimating nonlinear metrics used in calculations of correlation dimension and short-time Lyapunov exponent showed that both methods are adequate at low noise levels. However, SOD performed better at small scales, where POD showed large local fluctuations. At larger noise levels both methods perform similarly for the correlation sum, but SOD provides larger scaling regions for short-time trajectory divergence rates. POD still suffers from large variations at small length/time scales for larger noise levels.

The above quantitative metrics do not tell a whole story, however; we also need to look at the trajectories themselves. Visual inspection of the reconstructed phase portraits showed superior results for SOD. POD trajectories had small local variations even at low noise levels, and completely failed to recover any large dynamical structures at large noise levels. In contrast, SOD method provides very good phase portraits up to 20% noise levels, and for larger noise levels the reconstructions still have clear topological consistency with the original phase portrait. However, SOD also fails to

capture finer phase space structures for 80% noise level.

Problems

Problem 13.1

Create an artificial time series by integrating the Chua's circuit by Matsumoto *et al.* (1985):

$$\dot{x} = \alpha \left(y - x + bx + \frac{1}{2}(a-b)\left(|x+1| - |x-1|\right) \right) \tag{13.13}$$

$$\dot{y} = x - y + z \tag{13.14}$$

$$\dot{z} = -\beta y \tag{13.15}$$

using these parameters: $\alpha = 9$, $\beta = 100/7$, $a = 8/7$, and $b = 5/7$. Start the simulation using the following initial conditions: $x(0) = 0$, $y(0) = 0$, and $z(0) = 0.6$. In particular, using the z time series containing at least 60,000 points:

1. Make sure your time series is appropriately sampled, make it zero mean and unit standard deviation.
2. Determine the needed embedding parameters.
3. Make noisy time series with 1 %, 5 %, and 20 % additive Gaussian noise (percents are in terms of relative standard deviations of the signal and the noise).
4. Apply both projective and smooth noise reduction schemes to the noisy time series.
5. Compare the results using phase portraits, power spectrum, correlation sum, and Lypunov exponents.
6. Interpret your results.

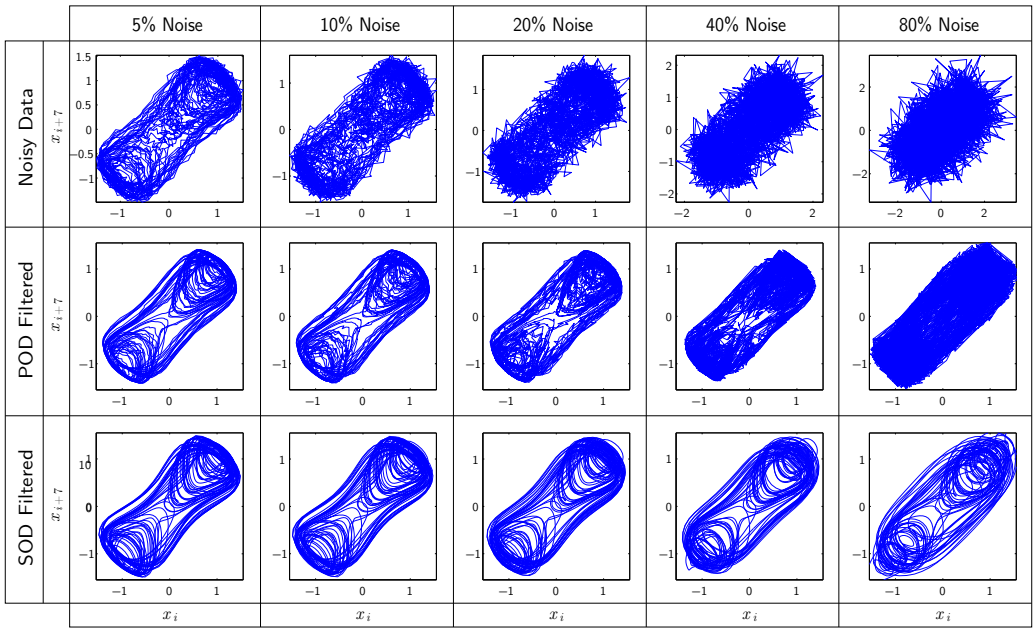

Figure 13.15: Phase space reconstructions for noisy and noise-reduced Duffing time series after ten iterations of the POD- and SOD-based algorithms

Chapter 14

Modeling and Predicting Chaotic Time Series

To understand the behavior of a dynamical system in terms of some meaningful parameters we seek the appropriate mathematical model that captures the important physical characteristics of the system that are observable experimentally. Ideally, we would like to have a simplified mathematical description of natural behavior that is consistent with the empirical observations, but can also be used to explore wider implications under different parameters or environmental conditions. Such models are called *phenomenological*, as opposed to *first principles* based models that are usually derived from fundamental physical principles (e.g., classical mechanics models). Obviously, the first principles models also need to be consistent with empirical evidence otherwise we need to reexamine our fundamental understanding of the underlying physics.

In this chapter, we will discuss the development of data-based models. In many cases, we would also have some tangential and incomplete information about the systems or the processes generating the data we use for modeling. However, we may need to proceed with just the available scalar or multivariate measurements and not much else. In any case, we cannot expect there to be a unique solution to this modeling problem and we have to often use other criteria in selecting specific model (e.g., the simples possible model consistent with observations is usually selected). There may be other metrics we can use to evaluate the usefulness of a model such as stability, convergence, or computability.

The main advantage of the first-principles models is that they inherently have physical meaning and parameters. In contrast, for models based on time series, we cannot expect to have parameters that have any physical meaning. However, a successful data-based model can faithfully reproduce empirically observed dynamics and the associated properties or statistics and, in this regard, is usually superior to the phenomenological model.

The process of data-based model building has two main components: (1) selection of particular model type or equation, and (2) identification of model parameters be fitting or updating the model predictions to empirically available data. If the identified model parameters have any phenomenological meaning the model is called *parametric*, otherwise it is called *nonparametric*. The selection of model equation is usually guided by the purpose of the model, which could have conflicting objectives. The usual objectives for model building are prediction of future evolution, noise reduction, control, system identification, or generation of some synthetic data sharing the same statistical or dynamical properties for the empirically available data.

The basic sources of predictability in the data are linear and nonlinear correlations and determinism. For linear systems, the determinism ie equivalent to having linear correlations. Therefore, usual choices for linear models are *moving average* (MA), *autoregressive* (AR), or combination of AR and MA (ARMA). For nonlinear systems, there is no such connection and to model chaotic time series we cannot just relay on autocorrelations. The natural choice for nonlinear systems is to extend linear models by introduction of local or global nonlinearity. For example, *threshold autoregressive* (TAR) models use piecewise liner modeling paradigm: they employ set of regular AR models that are valid only in small local neighborhoods of the vector space. These will be the members of local linear models. In contrast, we can extend the AR model by inclusion of nonlinear terms to create a globally nonlinear model. Thus, these would be classified as global nonlinear models.

In this chapter we will focus on models for deterministic or mainly deterministic data. Readers who are interested in AR, MA, ARMA, or their extensions can refer to any time series analysis books such as Refs. [5, 40, 65]. In what follows, we will discuss nonlinear models build in the reconstructed phase space of dynamical systems.

14.1 Model Development in Reconstructed Phase Space

We assume that the dynamics attractor can be reconstructed through the delay coordinate embedding of scalar time series using the delay τ and embedding dimension d. The process should also work and may benefit by having vector-valued time series too. Thus, we assume that we are working in the reconstructed phase space and we have total N points $\{\mathbf{x}_k\}_{k=1}^{N}$, which are governed by some nonlinear map:

$$\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n)\,, \tag{14.1}$$

where $\mathbf{F} : \mathbb{R}^d \to \mathbb{R}^d$. Then our objective is to approximate this map with a model:

$$\hat{\mathbf{x}}_{n+1} = \mathbf{P}(\mathbf{x}_n)\,. \tag{14.2}$$

To determine the appropriate form of $\mathbf{P}$ and its parameters, we need a criterion to evaluate the quality of the model. This could be accomplished by minimizing some cost function, selection of

which would depend on the purpose of the model. If, for example, we are looking for best possible predictions in the least squares sense, we may want to minimize mean squared *prediction error*:

$$e_k = \sqrt{\frac{1}{N}\sum_{n+1}^{N-k} \left(\mathbf{x}_{n+k} - \mathbf{P}^k(\mathbf{x}_n)\right)^2}\,, \tag{14.3}$$

where we are minimizing k-step prediction error. In many cases, we would just minimize for $k = 1$. However, there is a particular choice of $k = \tau$ (which is usually the case for data generated by maps when $\tau = 1$) that simplifies the modeling considerably. In particular, only one coordinate in $\mathbf{x}_{n+\tau}$ is different from $\mathbf{x}_n$, very similar to the Eq. (11.59) in Lyapunov exponent chapter. Therefore, Eq. (14.2) can be written as

$$\hat{\mathbf{x}}_{n+\tau} = \left[\hat{x}_{n+\tau}, x_n, x_{n-\tau}, \ldots, x_{n-(d-2)\tau}\right]^T = \left[P(\mathbf{x}_n), x_n, x_{n-\tau}, \ldots, x_{n-(d-2)\tau}\right]^T, \tag{14.4}$$

or just as

$$\hat{x}_{n+\tau} = P(\mathbf{x}_n)\,. \tag{14.5}$$

In many cases, even when data is coming from flow and not map, one does not bother to choose an optimal delay τ and the just uses the unit delay ($\tau = 1$). In this situation, and when the data are generated by continuous flow, successive measurements will be strongly correlated. Therefore, it is better to use the following form of a model:

$$\hat{x}_{n+1} = x_n + \Delta t\, P(\mathbf{x}_n)\,, \tag{14.6}$$

where Δt is the sampling time.[1]

14.2 Method of Analogues and Simple Averaging

If the data is clean and plentiful, the local models have some advantages over the global models. Local models usually require more computation at each prediction step, but can be more accurate if data is dense and has little noise. The simplest of local models or algorithms is Lorenz's *method of analogues*. This really does not involve any modeling, to predict the future evolution of a point in the phase space, all we need to do is lookup its nearest neighbor in our data and use its future image as a prediction. However, if we use this for long predictions, we would just reproduce our base trajectory, which would yield a periodic time series. To mitigate this problem, the *simple averaging* method looks up several nearest neighbors of the initial point, and uses the average of their future images as the prediction as illustrated in Fig. 14.1. This method could be effective when dealing with high dimensional state space or when we have large stochastic component added to our data.

[1] This is similar to the Euler integration of differential equations

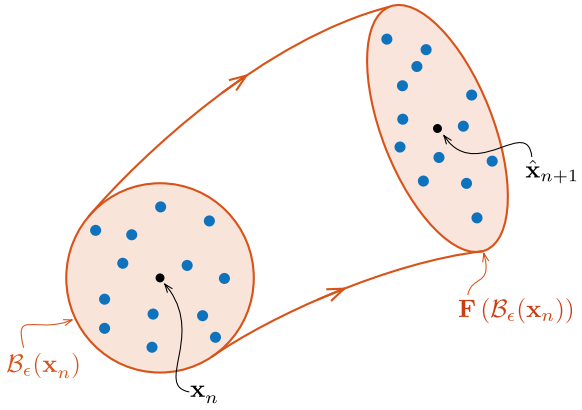

Figure 14.1: Simple averaging method looks up several nearest neighbors of $\mathbf{x}_n$ in its neighborhood and uses the average of their future images as the prediction $\hat{\mathbf{x}}_{n+1}$

14.3 Local Linear Models

We can do better than the simple averaging method by building *local linear models* which show how neighborhoods about each data point are mapped forward in time. Therefore, we can write a simple local linear map relating the state at time n, $\mathbf{x}_n$ to the state $\mathbf{x}_{n+k}$ at time $n+k$ as:

$$\hat{\mathbf{x}}_{n+k} = \mathbf{A}_n^k \mathbf{x}_n + \mathbf{b}_n^k \,, \tag{14.7}$$

where the model parameter matrix $\mathbf{A}_n^k$ and parameter vector $\mathbf{b}_n^k$ are determined by regression at each point in the data set using either a fixed distance method or fixed mass method to define local neighborhoods $\mathcal{B}_\epsilon(\mathbf{x}_n)$.

The basic procedure for modeling is explained schematically in Fig. 14.2. Given a point $\mathbf{x}_n$ in the phase space of the systems, we select a fixed number of points (fixed mass method), say r, in the neighborhood of that point $\mathcal{B}_\epsilon(\mathbf{x}_n)$. These same points, k time steps later, lie in some other small region of phase space $\mathbf{F}^k\mathcal{B}_\epsilon(\mathbf{x}_n)$ (k cannot be too large, since we expect to have a positive Lyapunov exponent). Then, as affine map Eq. (14.7) fitted in the least squares sense between these two clusters of points provides locally linear approximation to the dynamics of the system. Please note, the smaller the $\mathcal{B}_\epsilon(\mathbf{x}_n)$, the more accurate the fitted affine map is expected to be. Note, also, that for any fixed number r of neighboring points, the size of the $\mathcal{B}_\epsilon(\mathbf{x}_n)$ will be smaller if the data set is larger.

The collection of these local linear models gives us the global nonlinear model of the system. In addition, individual local linear models can be viewed as linearization of this global model at the points of interest in the phase space. Therefore, we implicitly require that the global nonlinear models to be not only continuous at out points of linearization but also have a first derivative. Thus, local linear models in effect are approximating a tangent space to the global map $\mathbf{F}(\mathbf{x}_n)$ at each $\mathbf{x}_n$. However, if the size of $\mathcal{B}_\epsilon(\mathbf{x}_n)$ is large compared to the inverse of the curvature of the

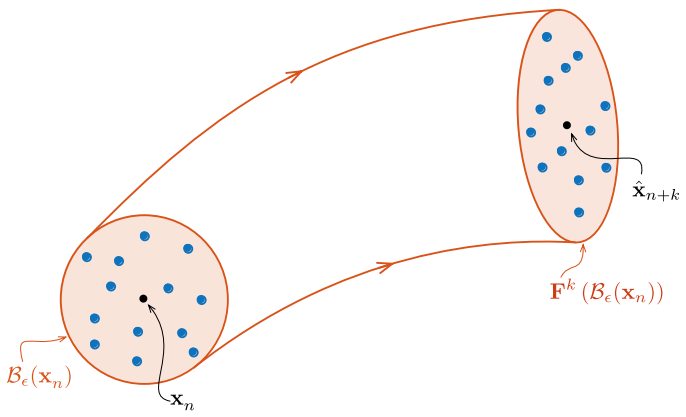

Figure 14.2: Schematic illustration of the basic idea about local linear models

true attractor at $\mathbf{x}_n$, then the approximation will be poor. We try to circumvent this problem of having small neighborhoods as well as sufficient enough nearest neighbors for estimating parameters by incorporating weighting in the procedure. Basing our model on r nearest neighbors ($\mathbf{x}_{j(i)}$, $i = 1, \ldots, r$) of the point of interest $\mathbf{x}_n$, we can find the optimal model parameters by minimizing

$$e_k^2 = \sum_{i=1}^{r} w_i \left\| \mathbf{x}_{j(i)+k} - \mathbf{A}_n^k \mathbf{x}_{j(i)} - \mathbf{b}_n^k \right\|^2 , \tag{14.8}$$

where w_i is a suitable weighting function that assigns greater weight to the points $\mathbf{x}_{j(i)}$ that are closest to the $\mathbf{x}_n$ within the needed neighborhood size. If we define $\mathbf{B}_n^k = \left[\mathbf{A}_n^k, \mathbf{b}_n^k\right]$ and the following matrices

$$\mathbf{X}_n = \left(\mathbf{x}_{j(1)}, \mathbf{x}_{j(2)}, \ldots, \mathbf{x}_{j(r)}\right) , \tag{14.9}$$

$$\mathbf{W}_n = \mathrm{diag}(w_1, w_2, \ldots, w_r) , \tag{14.10}$$

$$\tilde{\mathbf{X}}_n = \begin{pmatrix} \mathbf{x}_{j(1)} & \mathbf{x}_{j(2)} & \ldots & \mathbf{x}_{j(r)} \\ 1 & 1 & \ldots & 1 \end{pmatrix} , \tag{14.11}$$

$$\tag{14.12}$$

Then we can rewrite Eq. (14.7) as:

$$\mathbf{X}_{n+k} \approx \mathbf{B}_n^k \tilde{\mathbf{X}}_n . \tag{14.13}$$

If we use weighting and multiply the above from the right-hand side with $\mathbf{W}\tilde{\mathbf{X}}_n^T$ we get:

$$\mathbf{X}_{n+k} \mathbf{W} \tilde{\mathbf{X}}_n^T \approx \mathbf{B}_n^k \tilde{\mathbf{X}} \mathbf{W} \tilde{\mathbf{X}}_n^T , \tag{14.14}$$

Now, it is easy to show that the optimal model parameters are given by

$$\mathbf{B}_n^k = \mathbf{X}_{n+k} \mathbf{W} \tilde{\mathbf{X}}_n^T \left(\tilde{\mathbf{X}}_n \mathbf{W} \tilde{\mathbf{X}}_n^T \right)^{-1} . \tag{14.15}$$

An example of simple local linear prediction is shown in Fig. 14.3 for a Chua's circuit signal, using delay 6, embedding dimension 3 and using 16 nearest neighbors.

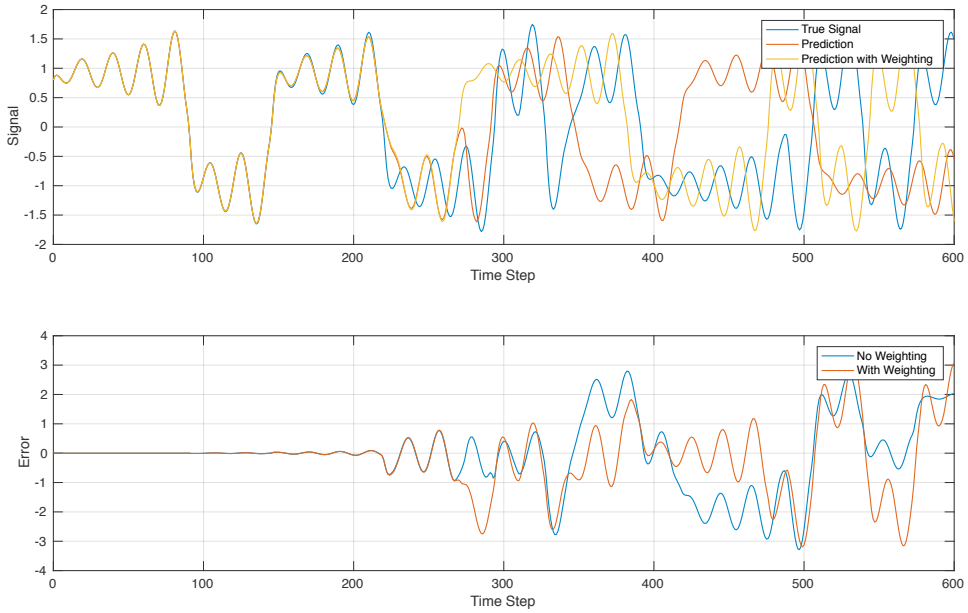

Figure 14.3: Local linear prediction for Chua's circuit x time series

There could be two major problems for local linear models: (1) we could have too few points to have sufficiently small neighborhoods for some points in the phase space, and (2) if the magnitude of noise is larger that the maximum allowable neighborhood size we will fit just noise not the deterministic component. In these cases, we may want to consider global nonlinear models such as ones using polynomial or radial basis functions. There is also a large body of literature utilizing neural networks for modeling chaotic behavior. The problem with the "black box" approaches, like neural networks, is that we loose any physical intuition about how and why they work and they may not be generalizable if we have even small changes in the system's parameters or operating conditions.

Problems

Problem 14.1

Use the same time series of Problem 10.2 and build a predictive local linear model from data. Split your data into two pieces, one to train the model, and the other to test the prediction. What are the in-sample and out-of-sample RMS prediction errors?

Chapter 15

Surrogate Data Analysis

In many practical situations we may want to classify or characterize our time series with few numbers (e.g., fractal dimension, Lyapunov exponent, model prediction error, or some entropy). These characteristic numbers then can be used for determining if the underlying dynamical systems is operating as expected or needs an attention. They could also be used as measures of some underlying hidden processes that may evolve on a slower time scales than the observed dynamics. Some examples include damage identification and monitoring, fault detection and localization, medical diagnosis, and etc. In all these cases we need to have trust in the estimated metrics, are they really reflecting the underlying dynamics or just giving us some numbers as a result of algorithmic calculations that have no physical meaning?

Many times we do not have a strong indication or evidence that what we are estimating can be trusted or is actually measure of some deterministic process. In this situations, we need to evaluate the probability that our measures are reflecting the physical reality by statistical hypothesis testing. However, due to the limited size of the experimental data we usually are limited in the ability to estimate the probability distribution of our metrics using, for example, different subsets of our data. Furthermore, it is usually fairly difficult to come up with simple null hypothesis for testing the complicated metrics we are using here. Due to all of this, we usually resort to *surrogate data analysis*.

The main idea behind the surrogate data analysis is that we want to compare our particular nonlinear metric estimated from the available time series to the distribution of the same metric obtained from a large number of time series (i.e, surrogate data) that satisfying some null hypothesis. This null hypothesis could be that the time series are generated by some linear stochastic process (i.e., autoregressive moving average or ARMA). Then, we can estimate the probability that our metric is real or obtained purely by chance by observing if and where it falls in the obtained distribution obtained from the surrogates. Then, if the probability of it being true is high enough we can reject

the null hypothesis.

The confidence level with which the null hypothesis can be rejected will depend on the number of surrogate signals used. We allow for a chance that null hypothesis could be rejected even if it is true. For example, if we require a 90% confidence level, we accept a 10% chance that the null hypothesis could be falsely rejected. To quantify this, we assign a value of 0.10 (10 %) to the parameter α. The number of surrogate data sets required is then determined by α. Statistics used to differentiate metrics, can be either single sided or double sided. Depending on which, the number of surrogates, β, will vary according to the following rules:

Single Sided Statistic: We expect our data statistic to either be always higher or lower than our statistic measured from the surrogates:

$$\beta = \frac{1}{\alpha} - 1$$

Double Sided Statistic: We expect our surrogate statistic to be higher or lower than our data statistic:

$$\beta = \frac{2}{\alpha} - 1$$

For this method to be effective our surrogate data needs to have some properties that are identical or similar to our time series to establish the corresponding null hypothesis. Since, we are mainly interested in testing for nonlinear metrics, we usually require that all or some of the linear properties of the test time series and surrogates be shared. This would support a powerful null hypothesis that the time series are generated by the ARMA model. Generally, we could consider these three null hypothesis or their combinations:

1. The data is composed of random numbers drawn from some fixed, but unknown distribution.
2. The data is drawn from a stationary linear stochastic process with Gaussian inputs.
3. The data is drawn from a stationary linear stochastic process with unknown distribution.

Each null hypothesis has a method of creating surrogate data sets associated with it. Each surrogate needs to have the same number of points as the original signal for the test to be valid. The three basic methods surrogate data generation methods are:

Unconstrained randomization of time series: This is a very simple form of surrogate data set. We will simply shuffle the time series in some random order.

Phase randomized surrogates: Here we require that the surrogates have the same power spectrum as our data, which is equivalent of requiring to having identical linear correlations. Hence, if there are nonlinear correlations in our test data, we should be able to reject the hypothesis. This method will preserve the power spectrum amplitudes of the time series while randomizing its phase. In particular,

1. Taking the digital fast Fourier transform (DFFT) of our measured time series $\{x_n\}_{n=1}^{N}$:
$$X_n = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} x_k e^{\frac{i2\pi nk}{N}} , \tag{15.1}$$
2. Replace the complex components of X_n by random phases, $X'_n = |X_n| e^{i\phi_n}$, where ϕ_n are uniformly distributed in $[0, 2\pi]$.
3. Compute an inverse digital fast Fourier transform (IDFFT) to get the surrogate:
$$x'_n = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} X'_k e^{\frac{-i2\pi nk}{N}} . \tag{15.2}$$

Iteratively refined surrogates: This is using the iterated amplitude adjusted Fourier transformed surrogates method as described in Ref [82]. This method preserves all the linear statistical and spectral properties in the surrogates (i.e., power spectrum, mean, variance, autocorrelation, probability distribution, etc.). However, any nonlinear structures in the data will be destroyed by the randomization procedure, which uses two step iterative method.

1. We start by sorting the original time series $\{x_n\}_{n=0}^{N-1}$ by its magnitude in ascending order into $\{y_n\}_{n=0}^{N-1}$ and storing it with the corresponding Fourier transform magnitudes:
$$X_n = \frac{1}{N} \left| \sum_{k=0}^{N-1} x_k e^{\frac{i2\pi nk}{N}} \right| . \tag{15.3}$$
2. The iterative process is initiated by randomly reshuffling $\{x_n\}$ to get $\{r_n^{(0)}\}$:
3. Take DFFT of $\{r_n^{(i)}\}_{n=0}^{N-1}$ to get:
$$R_n^{(i)} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} r_k^{(i)} e^{\frac{i2\pi nk}{N}} , \tag{15.4}$$
4. Now replace the magnitudes $\left|R_n^{(i)}\right|$ with X_n, but keep the phase $\psi_n^{(i)} = \arg R_n^{(i)}$ the same and take IDFFT to get
$$x_n^{(i)} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k e^{i\psi_k^{(i)}} e^{-\frac{i2\pi nk}{N}} . \tag{15.5}$$
5. Finally, transform $\{x_n^{(i)}\}$ into $\{r_n^{(i+1)}\}$ by rank-ordering according to stored $\{y_n\}$:
$$r_n^{(i+1)} = y_{\mathrm{rank}\left(x_n^{(i)}\right)} . \tag{15.6}$$

The idea is that iteratively we are converging to a fixed point $r_n^{(i+1)} = r_n^{(i)}$ for large i, which is usually reached after finite number of iterations.

15.1 Examples

To investigate the usefulness and abilities of this method we are going to look at two separate time series. The first time series will be a simple linear stochastic process, and the second will come from a simulation of the Lorenz attractor.

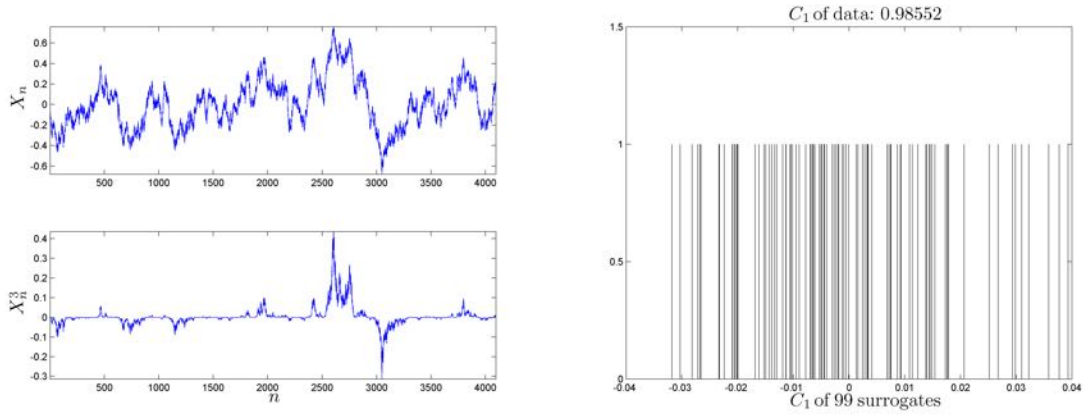

Figure 15.1: Stochastic process with its observation (left plots), and autocorrelation of the data in comparison with the autocorrelation of 99 surrogates (right).

15.1.1 Linear Stochastic System

The linear process can be described by the following:

$$x_n = 0.99x_{n-1} + \eta_n \,, \tag{15.7}$$

where η_n are independent random numbers. To make this problem more interesting for our application, we will observe the process through a nonlinear function,

$$s_n = x_n^3 \,. \tag{15.8}$$

It should also be noted that the mean of the signal is removed prior to the nonlinear transformation. The process along with the nonlinear observation can be seen in Fig. 15.1 (left). It is very possible to see a signal like this in real life. Many times, especially in biological systems, we do not know the governing equations and processes of a system. When we take some measurement of that system there could easily be a nonlinear rescaling. In this case it would be inappropriate to analyze the time series using nonlinear time series analysis techniques. For the sake of this problem we are going to assume we know nothing about the underlying process and only see the nonlinear observation.

The first approach will be to test the null hypothesis (number 1) that the data is of a random process. The statistic used will be the autocorrelation at lag one, and the surrogate method will be unconstrained randomization of the time series.

We want to pass every test in this paper with a 99 % confidence level. This requires that $\alpha = 0.01$, and therefore, $\beta = \frac{1}{0.01} - 1 = 99$ surrogates for single sided tests and $\beta = \frac{2}{0.01} - 1 = 199$ surrogates for double sided tests. The autocorrelation is expected to be much lower for a random signal than a linear stochastic signal, so this test will be single sided requiring 99 surrogates. Fig. 15.1 (right) shows the results of the statistic test. On the horizontal axis is the value of the autocorrelation and

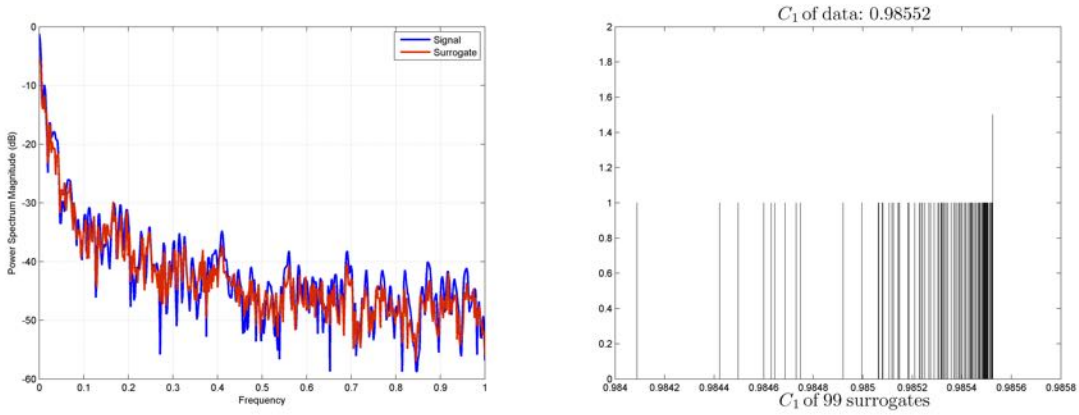

Figure 15.2: PSD of the original time series and one surrogate data set (left), and autocorrelation of 99 phase randomized surrogates plotted along with the autocorrelation of the data (right).

the vertical axis is for the purpose of illustration only. All surrogates are given a value of 1 and the data is given a value of 1.5 for ease of picking out the differences. If the autocorrelation of the data was close to the range of the distribution of the surrogates it would be plotted here as well. However it is obvious that the autocorrelation of all surrogates is much lower than that of the data. The null hypothesis is then rejected with 99 % confidence level and we can declare that the observation is not a random distribution of numbers.

This test does not tell us about the nonlinearity of the system. The next null hypothesis we can take is that the time series is a stationary linear stochastic process with Gaussian inputs. Again we will use the autocorrelation as the statistic with 99 surrogates. The surrogates will be generated using the phase randomized process. The power spectrum is well preserved as can be seen in the PSD plot of Fig. 15.2 (left).

The autocorrelation is then computed for the time series and 99 of its surrogates and plotted in Fig. 15.2 (right). From this test the null hypothesis cannot be rejected and the assumption that this process is actually a linear stochastic process with Gaussian inputs can hold. On the other hand, the statistic could also not be powerful enough to distinguish the nonlinearity.

The final null hypothesis taken here is that the data is drawn from a linear stochastic process with unknown distribution. For this test we will be using the error of a one time-step-ahead prediction as our statistic. This test is double sided and will require 199 surrogates. The surrogate sets will be generated using the iteratively refined (polished) randomization scheme. Here the distribution and the power spectrum are preserved as described earlier. Fig. 15.3 (left) shows the PSDs of the signal and one surrogate. The two match fairly well together. This method will not produce perfect matching by nature, but can be refined with an increase of iterations. Here, the iterations were kept relatively low for the sake of computational time. For predictive modeling, both the original time

series and surrogates were embedded in three dimensional space with unit delay. Fig. 15.3 (right)

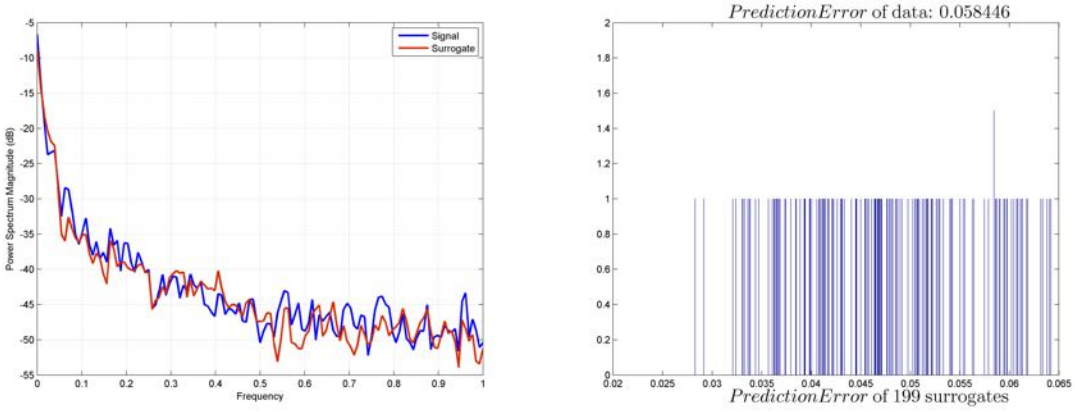

Figure 15.3: PSD of signal and one polished surrogate (left) and prediction error (right).

shows that the null hypothesis cannot be rejected with 99 % confidence here either. We can now reasonably assume that the data comes from a linear stochastic process, which we know is true.

15.1.2 Lorenz Time Series Example

The surrogates are generated using the refined iterative randomization method. Fig. 15.4 shows the results of the randomization. The preservation characteristics can be seen in Fig. 15.5 showing the

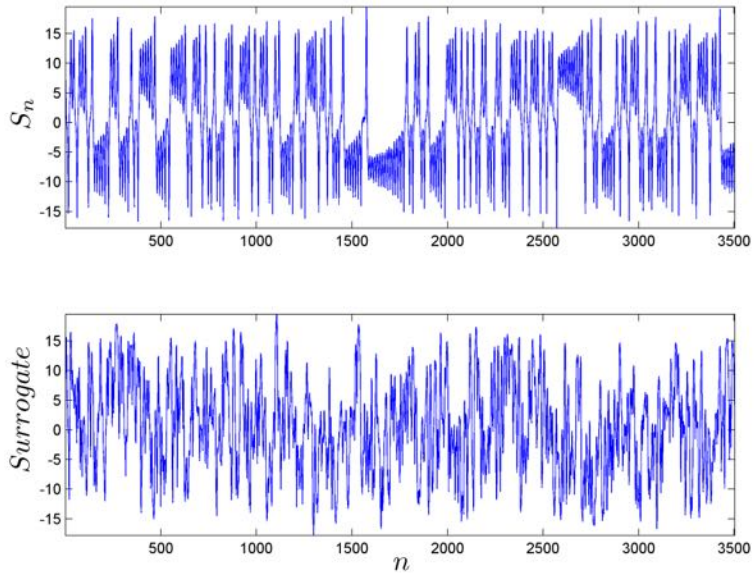

Figure 15.4: Lorenz data (top), surrogate (bottom)

PSD and histogram of the data and one surrogate. Noise is also added to the time series, $s_n = x_n + \eta_n$,

to make the problem more realistic. All signals are embedded using standard embedding parameters and the prediction algorithm is run identically. Fig. 15.6 shows the results of the statistical test.

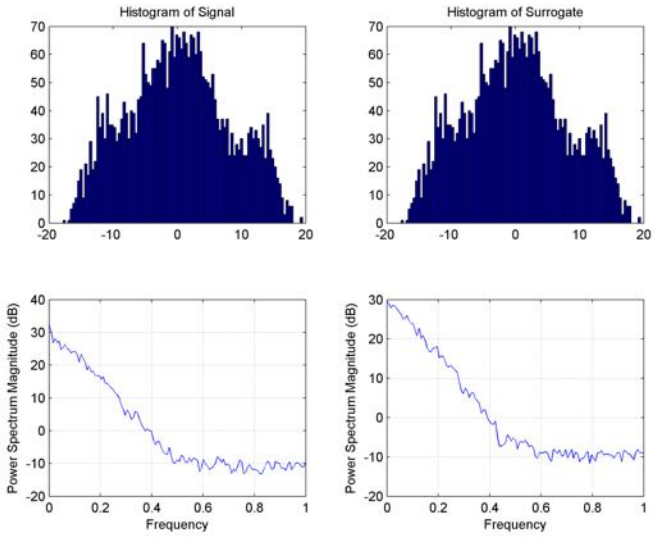

Figure 15.5: Comparison of distribution and PSD of Lorenz signal (left) and one surrogate (right)

Clearly, the null hypothesis can be rejected in every case giving us a 99 % confidence level that this signal is not a linear stochastic process measured through a nonlinear function.

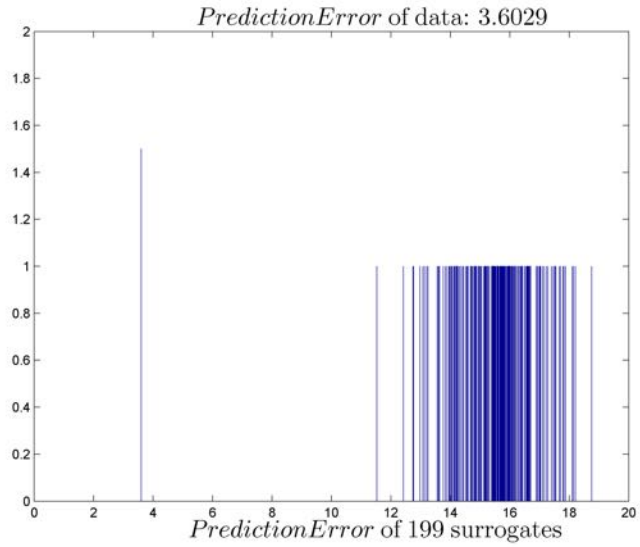

Figure 15.6: Lorenz attractor time series prediction error versus 199 surrogates

Chapter 16

Invariant Measures and Nonlinear Metrics Zoo

Classifying and identifying nonlinear dynamical systems are some of basic tasks in nonlinear time-series analysis [3, 26, 54]. Most generic approach is to rely on some feature metric that is invariant under the evolution of the dynamics. Conventional methods often characterize a dynamical system using long-time invariant quantities such as already discussed Lyapunov exponents or fractal dimensions [28, 46, 50, 73]. Fractal dimensions provide bounds on the number of active degrees-of-freedom and reflect the complexity of the system. Lyapunov exponents characterize the average exponential divergence rates of infinitesimally close trajectories on the attractor, and are used to infer if the underlying system is integrable or chaotic.

Application of these metrics in vibrations-based structural health monitoring has been studied extensively [34, 92, 93]. However, reliable estimation of Lyapunov exponents from the observed time series is difficult in the presence of noise, complicated by the fact that they are not exactly defined for noisy data [95]. Literature survey of techniques developed for the computation of Lyapunov exponents is extensive [4]. Most of the methods monitor the evolution of system dynamics for thousands of time steps. Therefore, large experimental data requirements and extensive computation time impose limits on their practical applications. To deal with these problems, Clement and Laurens proposed the Jacobian Feature Vector [22] which is based on Lyapunov exponent's algorithm but has faster computation. Other interesting feature extracted from measured data has been studied by Todd *et al.* [91], where local attractor variance ratio is described as a change in geometric properties of an attractor.

All these quantities describe the long-time behavior of a dynamical system. While they are able to detect sudden changes in a system, they are ill-suited for continuous tracking of slowly changing parameters responsible for nonstationarity in a dynamical systems. In Refs. [18, 19], a

concept of *phase space warping* (PSW) was proposed to characterize changes in slow parameter drifts in the fast-time flow. A one-to-one relationship between PSW-based tracking vectors and actual slow-time variables causing these changes has been demonstrated. However, this procedure requires considerable computational time and resources for estimation.

In many practical application like in MEMS-based sensors (e.g., in atomic force microscopy), observed data is accumulated at such high rates that it is impractical to record the raw time series or do complicated online, real-time analysis. Hence, it is more practical to do simple, easy to calculate feature extraction locally on the sensor and only output feature time series at much lower sampling rates. However, local processing has limited storage and computational resources, and it requires fairly simple and easy to estimate features (most use mean, variance, or resonant frequency estimations). These simple features do not capture important nonlinear behaviors and have limited application scope. To address this problem, a new class of simple nonlinear features that are fast to calculate and do not require large data are discussed here. These metrics are applicable in situations where fast online computation is needed.

16.1 A New Class of Nonlinear Metrics

Lyapunov exponents and fractal dimensions are rooted in Ergodic theory [96] which is originally developed to identify and classify invariant measures under the time evolution. In the following, one of the important theorems in Ergodic theory, Birkhoff Ergodic Theorem, is restated for completeness. Then a new class of characteristics of nonlinear dynamics based on this theorem is discussed.

16.1.1 Birkhoff Ergodic Theorem

Consider the time evolution of a dynamical system, measured in discrete time units, given by a transformation $\mathcal{T} : X \to X$, so that if $\mathbf{x} \in X \subset \mathbb{R}^n$ is the current state then $\mathcal{T}(\mathbf{x})$ is the state of the system after one time unit. If the system is at the steady state, by the invariant property $\mathcal{T}(X) \subseteq X$ then $\mathcal{T}$ is a measure preserving transformation. For example, let X be the phase space of a mechanical system. Then every point of X represents the values of position and momentum variables. A measurement of the system (e.g., velocity, acceleration) can be defined by a function $f : X \to \mathbb{R}$. To measure a quantity of the system, one usually takes n successive measurements $f(\mathbf{x}), f(\mathcal{T}(\mathbf{x})), \ldots, f(\mathcal{T}^n(\mathbf{x}))$ and looks at their average. The question is if the average exists and is invariant when $n \to \infty$. Ergodic theory was originally developed to answer this question.

> **Birkhoff Ergodic Theorem for measure preserving transformation** [96]: Let (X, B, μ) be a finite measure space. Let $\mathcal{T} : X \to X$ be a measure preserving transfor-

mation. For any $f \in L^1(X, B, \mu)$ the following limit

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} f\left(\mathcal{T}^k(\mathbf{x})\right) \tag{16.1}$$

exists and is invariant.

Using this theorem, we can define useful invariant metrics for a dynamical system's attractor which are fast, easy to calculate, robust to noise, and do not require large data or computational resources.

16.1.2 Generalized Distances

Let $g_q : X \times \mathbb{Z} \to \mathbb{R}$ be a measurable function, which is the *Minkowski distance* between a point $\mathbf{x} \in X \subset \mathbb{R}^m$ and its image after k time steps $\mathbf{y} = \mathcal{T}^k\mathbf{x} \in X \subset \mathbb{R}^m$,

$$g_q(\mathbf{x}, k) = \left\|\mathbf{x} - \mathcal{T}^k\mathbf{x}\right\|_q = \left(\sum_{i=1}^{m} |x_i - y_i|^q\right)^{\frac{1}{q}} . \tag{16.2}$$

Then a scalar *generalized distance* $d_q(k)$ can be defined as a limit

$$d_q(k) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} g_q(\mathcal{T}^i(\mathbf{x}), k) . \tag{16.3}$$

For example, this metric has some connection to auto-correlation or auto-covariance for $q = 2$:

$$\begin{aligned} d_2(k) &= E\left[\|\mathbf{x}_i - \mathbf{x}_{i+k}\|_2\right]_i = E\left[\sqrt{(\mathbf{x}_i - \mathbf{x}_{i+k})^T(\mathbf{x}_i - \mathbf{x}_{i+k})}\right] \\ &= \frac{1}{n-k} \sum_{i=0}^{n-k} \sqrt{\sum_{j=0}^{m-1} \left(x_{i+j\tau} - x_{i+k+j\tau}\right)^2} . \end{aligned} \tag{16.4}$$

For $q \neq 2$:

$$\begin{aligned} d_q(k) &= E\left[\|\mathbf{x}_i - \mathbf{x}_{i+k}\|_q\right]_i = E\left[\left(\sum_{j=0}^{m-1} |x_{i+j\tau} - x_{i+k+j\tau}|^q\right)^{\frac{1}{q}}\right]_i \\ &= \frac{1}{n-k} \sum_{i=0}^{n-k} \left(\sum_{j=0}^{m-1} |x_{i+j\tau} - x_{i+k+j\tau}|^q\right)^{\frac{1}{q}} . \end{aligned} \tag{16.5}$$

is no longer a linear statistic. One can obtain a full spectrum of these characteristic lengths for different k values at each q. However, the information in this metric becomes redundant once the characteristic length reaches the attractor size (similar to the largest Lyapunov exponent, or a correlation integral). These metrics have applications in determining the appropriate delay times from time series and can also provide information about nonlinear correlations in the data.

16.1.3 Characteristic Distance and Position

Let $f : X \to \mathbb{R}$ be a measurable function, which is the Euclidean distance between a point $\mathbf{x} \in X \subset \mathbb{R}^m$ and some fixed point $\mathbf{y} \in \mathbb{R}^m$,

$$f(\mathbf{x}) = \|\mathbf{x} - \mathbf{y}\|_2 \,. \tag{16.6}$$

Then a scalar *characteristic distance* $D(\mathbf{y})$ can be defined as a limit

$$D(\mathbf{y}) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} f(\mathcal{T}^i(\mathbf{x})) \,. \tag{16.7}$$

By the Birkhoff Ergodic Theorem, the characteristic distance is an invariant measure on an attractor. It can be seen as an average distance from an arbitrary fixed point in the phase space to an attractor.

There is nothing particularly special about this characteristic distance metric other than it is easy to define and estimate. Other similar metrics can also be defined and may be more appropriate in different applications. For example, we will also define and use vector-valued *characteristic position* as

$$\mathbf{P}(\mathbf{y}) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{f}_p(\mathcal{T}^i(\mathbf{x})) \,, \tag{16.8}$$

where again $\mathbf{y} \in \mathbb{R}^m$, and $\mathbf{f}_p : X \to \mathbb{R}^m$ is defined as

$$\mathbf{f}_p(\mathbf{x}) = \frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|_2} \,. \tag{16.9}$$

16.2 Reconstructing Slowly Drifting Variables

We consider a coupled dynamical system where a slow-time state variable causes drifts in the parameters of a fast-time subsystem:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\mu}(\boldsymbol{\phi}), t), \quad \dot{\boldsymbol{\phi}} = \epsilon \mathbf{g}(\boldsymbol{\phi}, t), \tag{16.10}$$

where $\mathbf{x} \in X \subset \mathbb{R}^n$ is a fast-time dynamic variable, $\boldsymbol{\phi} \in P \subset \mathbb{R}^m$ is a hidden slow-time dynamic variable, which alters the parameter vector $\boldsymbol{\mu} \in \mathbb{R}^p$. $t \in \mathbb{T} \subset \mathbb{R}$ is time, overdots denote time differentiation, $0 < \epsilon \ll 1$ is a small rate constant defining the time scale separation. This formulation is relevant to tracking and identifying damage processes as discussed in [20]. It is assumed that the variable $\boldsymbol{\phi}$ changes slowly. Thus, $\boldsymbol{\phi}$ is considered as approximately constant for a fast-time data set collected over an intermediate time interval.

A general solution to the fast subsystem of Eq. (16.10) can be written as $\mathbf{x} = \mathbf{X}(t, \mathbf{x}_0, \boldsymbol{\phi})$, where $\mathbf{x}_0$ is the initial condition. Hence, the continuous form of the characteristic distance for a fixed point $\mathbf{y}$ can be expressed as

$$D(\mathbf{y}) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \|\mathbf{y} - \mathbf{X}(t, \mathbf{x}_0, \boldsymbol{\phi})\|_2 \, dt \,. \tag{16.11}$$

This limit is approximated by the integral

$$D(\mathbf{y}) \approx \frac{1}{T_n} \int_0^{T_n} \|\mathbf{y} - \mathbf{X}(t, \mathbf{x}_0, \boldsymbol{\phi})\|_2 \, dt \,, \tag{16.12}$$

where T_n is a large number. By the Mean Value Theorem for integrals, there is a $t_i \in [0, T_n]$ such that

$$\frac{1}{T_n} \int_0^{T_n} \|\mathbf{y} - \mathbf{X}(t, \mathbf{x}_0, \boldsymbol{\phi})\|_2 \, dt = \|\mathbf{y} - \mathbf{X}(t_i, \mathbf{x}_0, \boldsymbol{\phi})\|_2 \,. \tag{16.13}$$

Therefore, an estimated value of $D(\mathbf{y})$ is a direct function of $\boldsymbol{\phi}$.[1] Because of bifurcations, the position of the attractor may change dramatically, therefore the characteristic distance $D(\mathbf{y})$ is not expected to be a continuous function of $\boldsymbol{\phi}$. However, the estimated characteristic distances may still provide valuable information about the parameter drifts in the system. In what follows, $D(\mathbf{y})$ for different points $\mathbf{y}$ will be used to reconstruct slow state variable $\boldsymbol{\phi}$.

It is assumed that scalar time series measured from a fast-time subsystem are recorded into m consecutive data sets. Each of the ith data set ($i = 1, \ldots, m$) is recorded over the intermediate time interval over which the variations in the slow variable are assumed negligible $\boldsymbol{\phi} \approx \boldsymbol{\phi}_i = \langle \boldsymbol{\phi} \rangle_i$, where $\langle \boldsymbol{\phi} \rangle_i$ is the average value of $\boldsymbol{\phi}$ within the ith data set. However, $\boldsymbol{\phi}_i$ itself changes gradually from one data set to another.

The characteristic distances are calculated for each data set and are assembled together in a feature vector

$$\mathbf{Y}^i = [D(\boldsymbol{\phi}_i, \mathbf{y}_1); D(\boldsymbol{\phi}_i, \mathbf{y}_2); \ldots; D(\boldsymbol{\phi}_i, \mathbf{y}_n)] \,, \tag{16.14}$$

where $\{\mathbf{y}_i\}_{i=1}^n$ are randomly chosen fixed points in the phase space. These feature vectors describe the gradual evolution of slow state variable $\boldsymbol{\phi}$ and are concatenated in time sequence into tracking matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$ in time sequence. This matrix $\mathbf{Y}$ embeds the changes in the parameters of the fast-time subsystem.

Previously, *smooth orthogonal decomposition* SOD [19, 21] has been used for identifying smooth trends in multivariate time series. Here, we also use SOD to extract deterministic slow trends from $\mathbf{Y}$. SOD analysis is performed using generalized singular value decomposition of matrix $\mathbf{Y}$ and its time derivative $\mathbf{DY}$, where $\mathbf{D}$ is a discrete differential operator. These matrices are decomposed as:

$$\mathbf{Y} = \mathbf{UCX}^T \quad \text{and} \quad \mathbf{DY} = \mathbf{VSX}^T \,, \tag{16.15}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, $\mathbf{C}$ and $\mathbf{S}$ are diagonal matrices, and $\mathbf{X}$ is a square matrix. The *smooth orthogonal coordinates* (SOCs) are given by the columns of $\mathbf{UC}$, *smooth projective modes* (SPMs) are provided by columns of $\mathbf{X}^{-T}$, and *smooth orthogonal values* (SOVs) are $\boldsymbol{\sigma} = \text{diag}(\mathbf{C}^T\mathbf{C})./\text{diag}(\mathbf{S}^T\mathbf{S})$ ('./' indicates term by term division). The greater the magnitude of the

[1] The same derivation can also be repeated for the characteristic position metric to show that $\mathbf{P}(\mathbf{y})$ is also a direct function of $\boldsymbol{\phi}$.

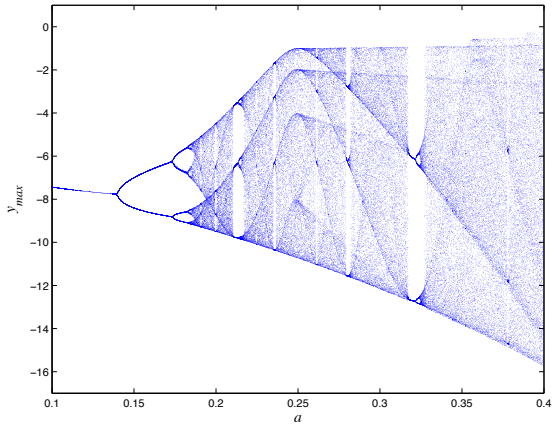

Figure 16.1: Bifurcation diagram for Rössler equation

SOV the smoother in time is its corresponding SOC. Since the slow-time state variable is a product of a smooth deterministic process, we hypothesize that this variable is embedded in the smoothest SOCs.

16.3 Numerical Examples

In the following, the time series derived from a Rössler and a Duffing equations are used to validate our slow variable tracking algorithm. In the simulations, a slow-time variable is introduced by slowly varying a parameter in the equations.

Rössler Equation is given by

$$\begin{aligned} \dot{x} &= -y - z\,, \\ \dot{y} &= x + ay\,, \\ \dot{z} &= b + z(x - c)\,. \end{aligned} \tag{16.16}$$

In our simulations, b is fixed at 0.6, c is fixed at 6.0, and a is varied sinusoidally from 0.1 to 0.4. Here, a is considered a slow-time variable. For each particular value of a, $100,000$ steady state trajectory points are generated with time step $t_s = 0.06$. The bifurcation diagram is shown in Fig. 16.1. Filled-in regions of the plot indicate chaotic regions.

Then characteristic distances from 50 arbitrarily chosen fixed points are estimated. Fig. 16.2 shows the plot of characteristic distance as a function of the bifurcation parameter a. Finally, the 786×50 tracking matrix $\mathbf{Y}$ is assembled. The tracking results after applying SOD to this matrix are shown in Fig. 16.3. Fig. 16.3(a) shows the first SOV is much larger than the rest, which is consistent with the fact that there is only one slowly varying parameter in the system. The SOC

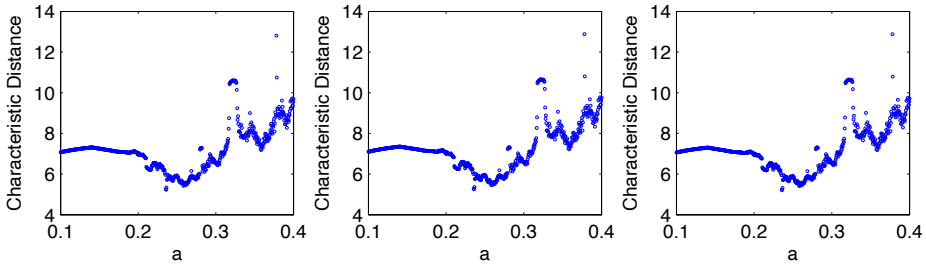

Figure 16.2: The characteristic distances versus a for three randomly chosen fixed points in the phase space

corresponding to the largest SOV is depicted in Fig. 16.3(c). To indicate the strength of the linear relationship between the real slow state variable, as shown in Fig. 16.3(b), and the first SOC, the correlation coefficient r is calculated. $r \approx 1$ shows that there is a strong linear relationship between the two variables as seen in Fig. 16.3(d).

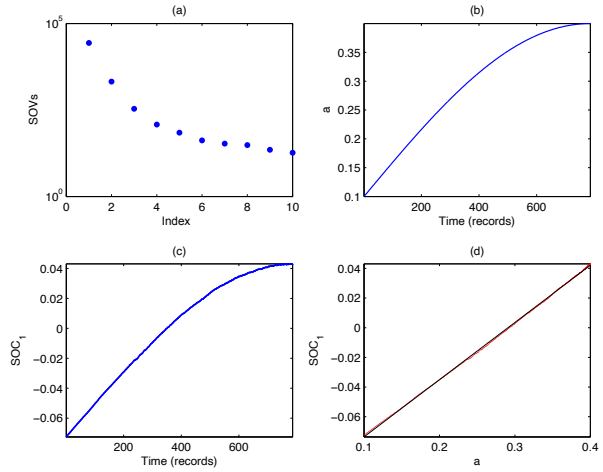

Figure 16.3: First ten SOVs (a), parameter a versus time (b), dominant SOC_1 corresponding to the largest SOV (c), plot of SOC_1 versus parameter a (red dots) with least square linear fit (black line) (d). The correlation coefficient between a and SOC_1 is $r = 0.9999$.

Duffing Equation In the previous example, the real phase space of Rössler equation is used to calculate the characteristic distances. In this example, only a time series x from a two-well Duffing equation is considered:

$$\ddot{x} + \gamma\dot{x} + \alpha x + \beta x^3 = f\cos\omega t \tag{16.17}$$

where the system's parameters and forcing frequency are fixed to $\gamma = .25$, $\alpha = -1$, $\beta = 1$, and $\omega = 1$. f is used as a slowly drifting variable or a bifurcation parameter. The bifurcation diagram

is shown in the Fig. 16.4. We consider the case when f is changed slowly and linearly from 0.35 to 0.4. In this range of f, the response of the system is in the chaotic region.

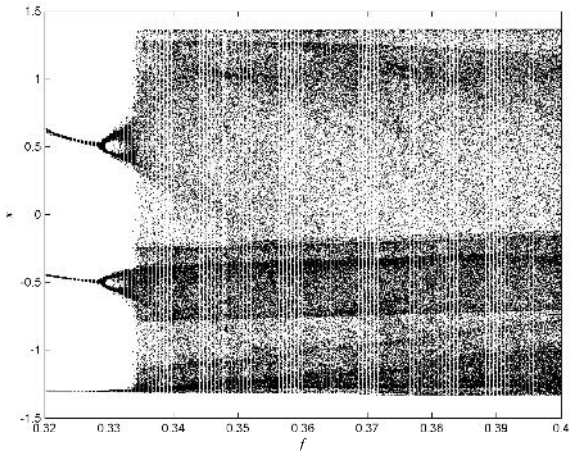

Figure 16.4: Bifurcation diagram for Duffing oscillator

In the calculations, the first 50 cycles of data are dropped, and 100,000 steady state points are recorded for each forcing amplitude using a sampling time $t = \pi/36$. The phase space is then reconstructed using a time delay $\tau = 53$ and embedding dimension $d = 4$ [3, 54]. Fig. 16.5 shows the characteristic distances to the reconstructed phase space from three randomly chosen points.

400 randomly chosen points in the phase space are used to assemble 1001×400 tracking matrix $\mathbf{Y}$. Then the SOD-based tracking result is shown in Fig. 16.6. Again, we obtain a linear relationship between the smoothest SOC_1 and the real slowly drifting variable f as seen in Fig. 16.6(d). Please note that these tracking results are still possible even when each component of matrix $\mathbf{Y}$ is discontinuous as shown in Figs. 16.2 and 16.5.

16.4 Experimental Data Example

Data generated from a modified version [24] of the well known two-well magneto-elastic oscillator [71] is used to illustrate the slow-parameter tracking. Description of this experiment is given in Ref. [21]. The basic schematic of the experiment is shown in Fig. 16.7. In this experiment, a couple of electromagnets powered by a computer-controlled power supply are used to cause perturbations in the magnetic potential at the free end of a vibrating cantilever beam. The vibration of the beam is measured by two laser vibrometers (CH1, CH2) mounted near the clamped end of the stiffened beam. The position of the beam is calculated by the difference between CH1 and CH2. The beam is excited by a 10 Hz harmonic load, and its amplitude is set so the beam undergoes nominally chaotic cross-well oscillations for fully loaded (10 V) electromagnets. Vibration data is low-pass filtered with

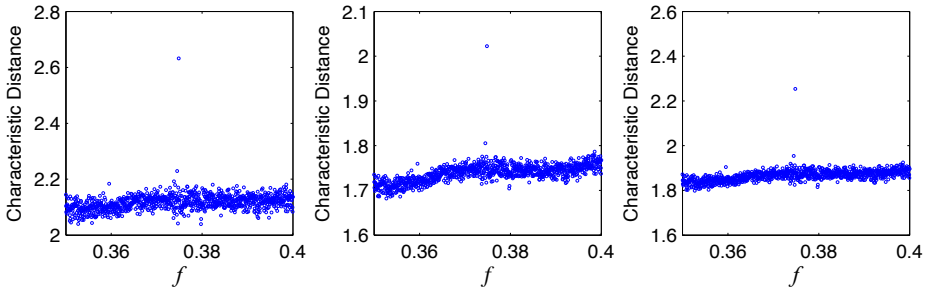

Figure 16.5: Characteristic distances from the three randomly chosen points in the phase space of the Duffing oscillator

50 Hz cut off frequency and collected at a 160 Hz sampling frequency.

The voltages supplied to the electromagnets ($v_1(t)$ and $v_2(t)$) are altered harmonically and independently as shown in Fig. 16.9(b). These voltages play a role of slow state variables in the system.

The experiment lasts about 12 hours and 6.6 million data points are recorded [21]. The 5-dimensional fast-time phase space is reconstructed using a delay time of six time samples ($t_s = 1/160$

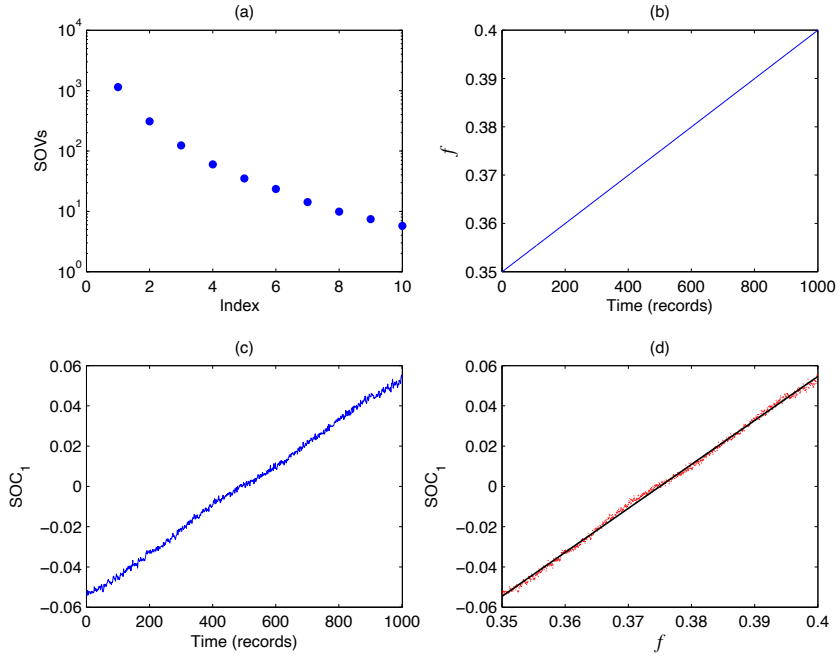

Figure 16.6: First ten SOVs (a), parameter f versus time (b), dominant SOC_1 corresponding to the largest SOV (c), plot of SOC_1 versus parameter f (red dots) with least square linear fit (black line) (d). The correlation coefficient between f and SOC_1 is $r = 0.9991$.

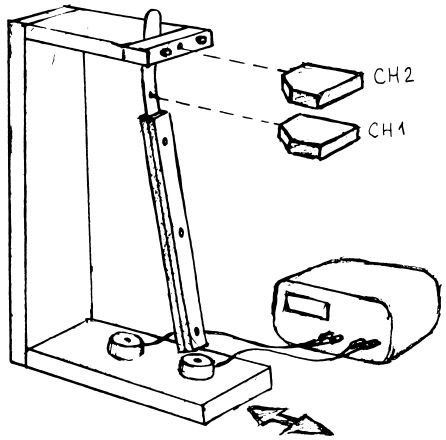

Figure 16.7: Schematic of the experimental apparatus

s). The time series are split into 800 data records, and each record contains 2^{13} points. Here, the characteristic distances are calculated from 300 fixed points randomly chosen in the reconstructed phase space. Then the 800×300 tracking matrix $\mathbf{Y}$ is assembled. Three randomly chosen columns of the tracking matrix are shown in Fig. 16.8. The SOD-based tracking result is shown in Fig. 16.9(a) and (c). The two largest SOVs are several orders-of-magnitude larger than the rest, which is consistent with two-dimensional slowly varying variable in the system. The SOCs corresponding to these two largest SOVs are depicted in Fig. 16.9(c). The phase portraits of the actual power supply terminal voltages and two smoothest SOCs are shown in Fig. 16.10.

The scaled SOCs are compared with $v_1 + v_2$ and $v_1 - v_2$ as shown in Fig. 16.9(d). The correlation coefficients between $v_1 + v_2$, $v_1 - v_2$ and SOC_1, SOC_2 are 0.9951 and 0.9834, respectively. The result confirms that the slow-time state variable is embedded (or reconstructed) in the smoothest SOCs as indicated by topological similarity of phase portraits in Fig. 16.10.

16.5 Noise Effects

To illustrate the robustness of characteristic distances to noise, normally distributed random noise was added to Rössler's attractor at 1/5, 3/5 and 5/5 RMS amplitude ratios, which corresponds to 20%, 60% and 100% noise in the signal. Fig. 16.11 shows that the characteristic distances for a particular fixed point drift under the presence of noise. However, the shape of the plot of characteristic distance versus bifurcation parameter a does not change noticeably. The tracking results of slow state variable are shown in Fig. 16.12. Even for 100% noise, our method still recovers the real slow state variable.

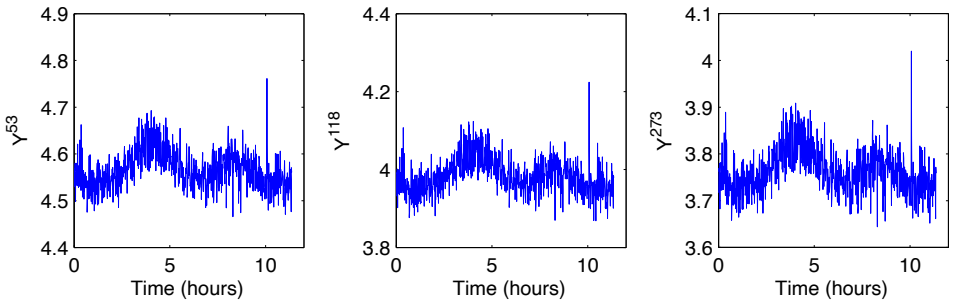

Figure 16.8: Three randomly chosen column of the tracking matrix

16.6 Discussion

The application of the SOD was warranted by the assumption that feature vectors were in smooth functional relationship with the slow-time state variables. In our simulation examples, the systems exhibited complicated bifurcations while one of the parameters was varied slowly. Therefore, individually, the estimated characteristic distances were not smooth function of the drifting parameters. However, when we combined the characteristic distances from different fixed points and applied SOD to the tracking matrix, we recovered the corresponding smooth deterministic trends, which can be used for one-to-one tracking even in noisy environments.

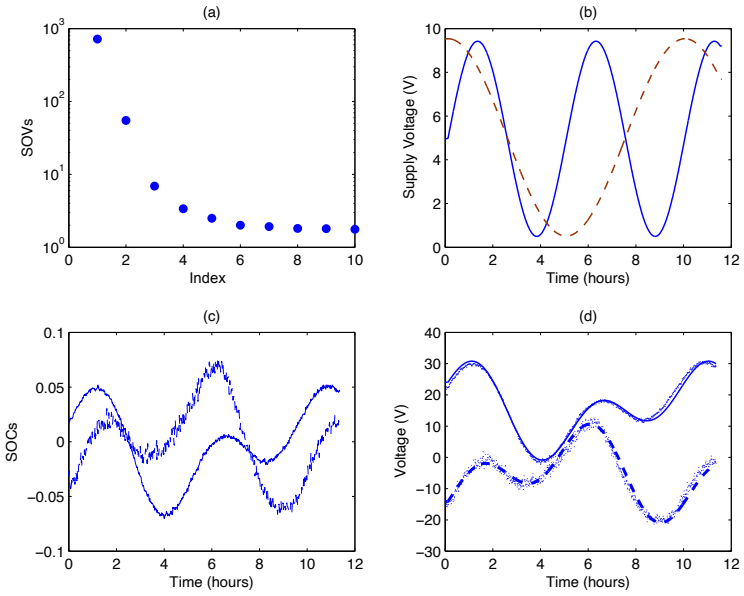

Figure 16.9: Plot of ten largest SOVs (a); actual electromagnet supply voltages v_1 (—) & v_2 (- -) versus time plot (b); plot of the first (—) and second (- -) SOCs corresponding to the two largest SOVs (c); plots of $v_1 + v_2$ (—), $v_1 - v_2$ (- -), and the scaled first two SOCs (.) (d).

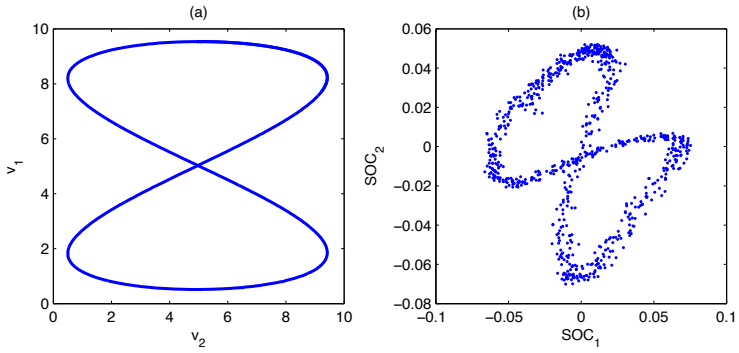

Figure 16.10: Phase portrait of the power supply terminal voltages (a) and two smoothest SOCs (b)

We also studied the quality of tracking results versus the number of fixed points and their locations. In numerical simulation, 50 fixed points were sufficient. However, in the experiment, we need 300 feature vectors to obtain robust results. To reduce the required number of fixed points and complexity of the tracking matrix, instead of using characteristic distances, we also used characteristic positions as defined in Eq. (16.9). For example, in the Rössler simulation, instead of using 50 fixed points, we used characteristic positions of only 20 points for tracking (see Fig. 16.13(a) for the tracking result). For experimental data, characteristic positions of 60 points are enough to obtain reliable result in comparison with characteristic distances of 300 points as shown in Fig. 16.13(b).

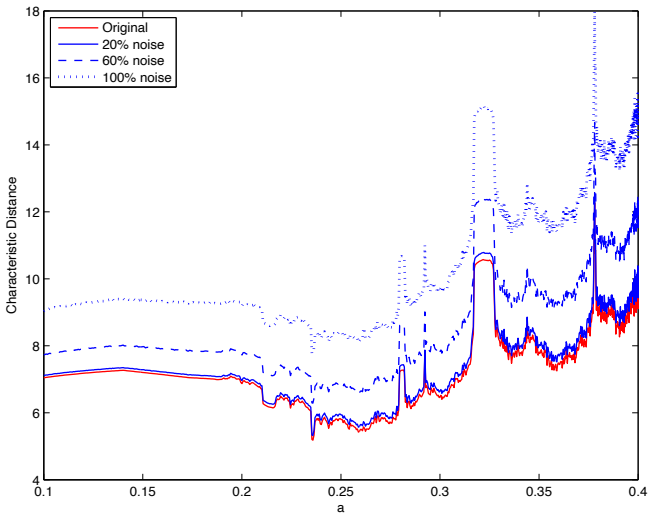

Figure 16.11: Characteristic distances of one fixed point.

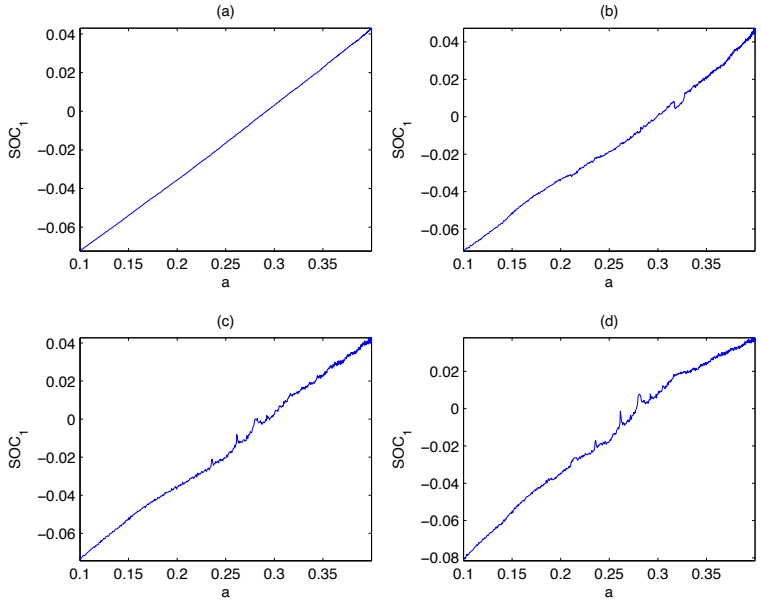

Figure 16.12: Tracking results. (a) no noise; (b) 20% noise; (c) 60% noise; (d) 100% noise.

16.7 Summary

Birkhoff Ergodic theorem for measure preserving transformation was used to develop a new class of nonlinear metrics for dynamical systems. In particular, a characteristic distance metric was defined by the average distance from a fixed point in a phase space to all points on the attractor, and the characteristic position was given by the normalized average location of the attractor with respect to a fixed point. These metrics were shown to be sensitive to small parameter changes in a system, and were used to track and identify these changes in conjunction with smooth orthogonal decomposition. The main advantage of these metrics was that they were simple and easy to calculate, while also being robust to noise. These properties make them suitable for online, real-time applications. Numerical simulations, synthetic data, and experiments were used to show how the slowly evolving parameters can be continuously tracked and identified using the considered metrics.

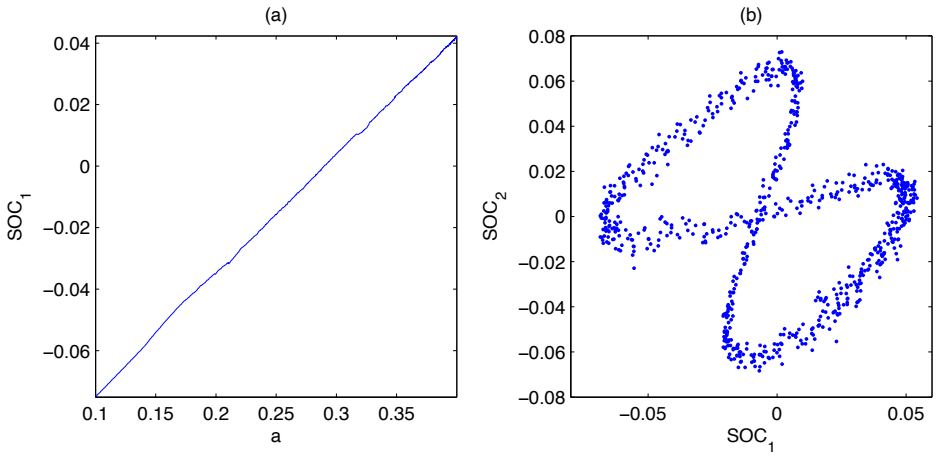

Figure 16.13: (a). Tracking a parameter in Rössler equation; (b). Phase portrait of two smoothest SOCs in the experimental data.

Appendices

Appendix A

MATLAB Functions

A.1 Average Mutual Information Algorithm

```
function [Ixx] = ami( x, n, m, nb )
% AMI   calculates average mutual information for the input scalar time
%       series. if no output is requested it plots the AMI curve.
%
% CALL: Ixx = ami( x, n, m, nb );
%
% x  - input scalar time series
% n  - number of points to use in the calculation
% m  - maximum delay parameter to consider
% nb - number of bins to use in calculation (default=16)
% p  - length of continuously sampled data
% Ixx - average mutual information in bits
%
% Copyright and Revised by David Chelidze 2-11-2009 & 12/12/17
% chelidze@uri.edu

if nargin < 3
   error('Requires at least three input arguments!')
elseif nargin == 3
   nb = 16; % set the number of bins to default
end
```

```
% check data and make a row vector of scalar data
if min(size(x)) ~= 1
   error('first input should be a vector of scalar data points.')
else % normalize the input data
    x = (x(:) - mean(x))/std(x);
end

% get the matrix of delayed versions of x

nmax = size(x,1) - m; % <- thats the maximum size possible!!!

% get the maximum possible matrix size!!!
if (n == 0) || (n > nmax)
    warning(['Specified data size is zero or exceeds maximum possible'...
        ' value!\n Continuing using maximum possible size!\n'])
    n = nmax;
end

y = zeros(n,m+1); % initialize array for embedding full subsets
for i = 0:m
    y(:,i+1) = x(1+i:n+i);
end

% obtain Px
x1 = y(:,1);
Px = hist(x1, nb); Px = Px/n;
[~, ~, NPx] = find(Px);

% initialize Ixx
Ixx = zeros(m+1,1);

for ii = 1:m+1 % calculate the average mutual information
   x2 = y(:,ii);
   [Pxy, ~, ~] = joint_prob(x1, x2, nb);
   [~, ~, NPxy] = find(Pxy);
   Ixx(ii) = sum(NPxy.*log2(NPxy)) - 2*sum(NPx.*log2(NPx));
```

```
end

if nargout == 0
    plot(0:m,Ixx,'o-','markerfacecolor','w'); grid on
    xlabel('Delay (sampling time)');
    ylabel('Average Mutual Information (bits)');
end

%%-------------------------------------------------------------------------
function [Pxy, xx, yy] = joint_prob( x, y, nbx, nby )
%
% Determines the joint Probability P(x,y) based on
% distributed sets of scalar variables x and y
%
% call [Pxy, xx, yy] = joint_prob( x, y, nbx, nby );
% Inputs:
% x & y - vectors (time series)
% nbx   - number of bins for x
% nby   - number of bins for y
% Output: Pxy - (nbx,nby) matrix
% It plots results if no output is requested
%
% Copyright and Revised by David Chelidze 7-31-2000
% >>>> Needs cond_prob.m <<<<

if (nargin == 0 || nargin == 1)
   error('Requires at least two input arguments')
end
if length(x) ~= length(y)
   error('Both arrays should be the same size')
end
if nargin == 2
   nbx =16; nby = 16;
elseif nargin == 3
   nby = nbx;
end
```

```
% put in column form
y = y(:); x = x(:);
[Px_y,xx,yy] = cond_prob(x,y,nbx,nby);
Py = histc(y,yy); Py = Py(:)/length(y); Py = Py(1:end-1);
Pxy = Px_y.*(ones(nbx,1)*Py');
end

%%--------------------------------------------------------------------------
function [Px_y, xx, yy] = cond_prob( x, y, nbx, nby )
% COND_PROB     Determines the conditional Probability P(x|y) based on
%               distributed sets of scalar variables x and y
%
% CALL: [Px_y, xx, yy] = cond_prob(x, y, nbx, nby);
% INPUTS:
% x & y - vectors (scalar time series)
% nbx   - number of bins for x
% nby   - number of bins for y
% Output: Px_y - (nbx,nby) matrix
% It plots results if no output is requested
%
% Copyright and Revised by David Chelidze 7-31-2000

if (nargin == 0) || (nargin == 1)
   error('Requires at least two input arguments')
end
if length(x) ~= length(y) % this can be changed for other applications
   error('Both arrays should be the same size')
end
if nargin == 2
   nbx = 16; nby = 16;
elseif nargin == 3
   nby = nbx;
end
% initialize the conditional probability density matrix
Px_y = zeros(nbx, nby);
% put inputs into column form
```

```
y = y(:); x = x(:);
% get bin sizes
minx = min(x); maxx = max(x);
binx = (maxx - minx)/nbx;
miny = min(y); maxy = max(y);
biny = (maxy - miny)/nby;
% get bin boundaries (edges)
xx = minx + binx*(0:nbx);
xx(1) = -inf; xx(end) = inf;
yy = miny + biny*(0:nby);
yy(1) = -inf; yy(end) = inf;
for j=1:nby % calculate!
   iy = find((y > yy(j)) & (y <= yy(j+1))); % # points in bin # j
   if ~isempty(iy)
      Hx_y = histc(x(iy), xx);
      Hx_y = Hx_y(:);
      Px_y(:,j) = Hx_y(1:end-1)/length(iy);
   end
end
end
end
```

A.2 Correlation Dimension Algorithm

```
function [r, c] = c2r( x, d, m, n, r, p, q )
% C2R function calculates the correlation sum from a scalar time
%  series after reconstructing its phase space using delay coordinates
%  It uses Euclidean distances!
%
% CALL: [r, c] = c2r( x, d, m, n, r, p, q );
%       If no output is requested it will just plot the results
%
% INPUT:
%   x -- scalar time series
%   d -- embedding dimension
%   m -- delay time
```

```
%   n -- number of points to use in the reconstructed phase space
%   r -- number of bins (ball length-scales) to use if it's just a number,
%        or the array of ball sizes (bins) to consider if it's a vector
%   p -- (optional) supply 0 to NOT remove temporal correlations
%        or number of points to remove. The default is p = d*m.
%        NOTE: only need to remove these correlations for the continuous
%        flow data!
%   q -- (optional) only supply if instead of removing temporal
%        correlations you wish to randomly populate attractor with q sample
%        points (thus, this will ignore p input if provided, & use max
%        number of points possible in the phase space reconstruction)
%
% OUTPUT:
%   r -- ln of ball sizes
%   c -- ln of correlation sum
%
% copyright by David Chelidze 3/27/2002
% revised by DC 5/31/2002 & 6/12/2002 & 10/2/2002 & 10/30/2002
% & 11/04/2005 & 3/3/3009 & 11/4-21/2013 & 12/12/2013 & 10/20/17 & 12/12/17

if nargin < 5 % not enough input variables
    error('Error: Needs at least five input variables.')
end
if min(size(x)) ~= 1
   error('Error: The first input should be a vector of scalars.')
else % normalize the input time series
    x = (x(:)' - mean(x))/std(x); % make it zero mean and unit variance
end
nm = length(x) - (d-1)*m; % maximum number of reconstructed points possible
if isempty(n) % use max
    n = nm;
elseif n > nm % use max possible and notify user too
    fprintf(['WARNING: Max number of embedded vectors exceeded.\n'...
        'ACTION:  Continuing with max number of vectors possible.\n'])
    n = nm;
end
```

```
if nargin < 6 || isempty(p) % remove temporal correlations of this length
    p = d*m; % still open for debate, may want to use different length
end

%% reconstruct the phase space to analyze interpoint distance distributions
if nargin == 7 && isempty(q) < 1 % randomly populate attractor w/ points
    indx = randperm(n, q); % pick q random points from the reconstruction
    n = q; % new number of phase space points to use
    p = 0; % do not need to remove temporal correlations
else % use the whole phase space
    indx = 1:n;
end
y = zeros(d,n); % initialize
for i = 0:d-1 % reconstructed phase space
    y(i+1,:) = x(indx+i*m);
end

%% calculate the correlation sum
np = n - p;
if length(r) == 1 % determine the needed length scales or bins to use
    c = zeros(1,r);
else % use the provided length scale or bins
    r = r.^2;
    c = zeros(size(r));
end
for i = 1:np-1 % for every point in the reconstructed phase space
    indx = i+p+1:n; % index of points to calculate distance to
    dist = 0; % initialize distances
    for k = 1:d % calculate distances squared to all other points
        dist = dist + (y(k, indx) - y(k, i)).^2;
    end
    if i == 1 && length(r) == 1 % find the suitable bins or length scales
        r = exp( linspace( ...
            log(min(dist)/np),log(max(dist) + min(dist) ), r) );
    end
    c = c + cumsum( histcounts(dist, [0, r]) ); % correlation sum
```

```
end
c = log(2*c/np/(np-1)); % ln of correlation sum
r = log(r)/2; % ln of the corresponding ball sizes

if nargout == 0 % plot the results
    figure
    set(gcf,'PaperSize',[10.5 3.5],'PaperPosition', [-.75 -.25 12 4]);
    subplot(121), plot(r,c,'-','linewidth',1)
    set(gca,'fontsize',12)
    grid on, pbaspect([1.618, 1, 1])
    xlabel('$\ln(\epsilon)$','Interpreter','Latex','FontSize',14)
    ylabel('$\ln \hat C(\epsilon)$','Interpreter','Latex','FontSize',14)
    subplot(122), plot(r(1:end-1),diff(c)./diff(r),'.-','linewidth',1)
    set(gca,'fontsize',12)
    grid on, pbaspect([1.618, 1, 1])
    xlabel('$\ln(\epsilon)$','Interpreter','Latex','FontSize',14)
    ylabel(['$D_2 \approx \frac{\mathrm{d}\ln \hat C(\epsilon)}',...
        '{\mathrm{d}\ln(\epsilon)}$'],'Interpreter','Latex','FontSize',14)
end
```

A.3 Pointwise Dimension Algorithm

```
 function [lnD, lnN] = pwdim( x, m, d, k, N, w )
% PWDIM: estimates average pointwise dimension using fixed distance and
%        fixed mass methods
% NOTE: York claims it is equivalent to information dimension,
%       Moon claims it is like correlation dimension
%
%   CALL [lnD, lnN] = pwdim( x, m, d, k, N, w );
%
% INPUT:
%   x - scalar time series
%   m - delay parameter (time delay for embedding)
%   d - embedding dimensions to use (it can be an array!)
%   k - number of distance partitions (for fixed distance) or points inside
%       a ball (for fixed mass)
```

```
%   N - maximum number of averages to do (optional -- 2^12 is default)
%   w - (optional) Theiler window to remove the correlations, by default
%       not temporal correlations are removed. This does not seem to alter
%       the results significantly, but considerably increase the
%       computational time.
%
% OUTPUT: The first two columns are fixed distance and the second are fixed
%         mass. If no output is requested, then it plots the results.
%   lnN(:,1)  - <ln(fraction of points inside the eps-ball)>
%   lnD(:,1)  - ln(eps)--i.e., log of the epsilon-ball
%   lnN(:,2)  - ln(n/N)--i.e., faction of points or mass
%   lnD(:,2)  - <ln(ball size containing n points)>
%
%     copyright by David Chelidze, 1998-2013
%     modified 10/26/17 by DC

%% Initialization and preprocessing
if nargin < 3
   error('Requires at least three input arguments!')
else
    x = x(:)'; % put 'em in correct form and get parameters
    n = length(x)- m*(d-1); % <- that's the maximum size possible!!!;
    if nargin < 6 % do not remove temporal correlations
        w = 0;
    end
    if nargin < 5 || N == 0 || isempty(N) % set the default
        N = min(2^12, n);
    else % use the provided or max possible points
        N = min(N, n);
    end
    if nargin < 4 || k == 0 || isempty(k) % use the default # of NNs
        k = 500;
    end
end
lnN = zeros(N,k); % initialize the log of measure array
y = zeros(d, n); % initialize array for embedded vectors
```

```
for i = 1:d % reconstruct the phase space
    y(i,:) = x((1:n) + (i-1)*m);
end % embedding

%% create the kd-partition search for nearest distances
ns = createns(y',’NSMethod’,’kdtree’,’Distance’,’euclidean’);
% search for the k points nearest to the test points (+ 2*w if removing
% temporal correlations)
np = k + 1 + 2*w; % total nearest neighbors to look up
[idx, dis] = knnsearch(ns,y(:,randperm(n,N))’,...
    ’Distance’,’euclidean’,’k’,np);
if w == 0 % do not remove temporarily correlated nearest neighbor distances
    dis = dis(:,2:end); % only drop the first zero distance
else % remove temporal correlations
    dis = cell2mat( cellfun( @removecorrs,mat2cell(dis,ones(N,1),np), ...
          mat2cell(idx,ones(N,1),np),’UniformOutput’,false ) );
end

%% the fixed mass based estimates:
lnDm = mean(log(dis)); % average log of the eps-ball
% lnNm = log(1:k) - log(N); % measure inside the eps-ball or use:
lnNm = psi(1:k) - log(N); % Grassberger (1985) measure inside the ball

%% the fixed distance based estimates
dm = max(dis(:)); % maximum distance
edgs = [0 exp(linspace(lnDm(1), mean([lnDm(end) log(dm)]), k))];
for ii = 1:N
    ln = log( cumsum(histcounts(dis(ii,:), edgs)) );
    ln( ln==-Inf ) = 0; % if count is zero assign zero
    lnN(ii,:) = ln;
end
lnN = [sum(lnN)./sum(lnN~=0) - log(N); lnNm]’;
lnD = [log(edgs(2:end)); lnDm]’;

%% plotting the results
if nargout == 0
```

```
    figure
    set(gcf,'PaperSize',[10.5 3.5],'PaperPosition', [-.75 -.25 12 4]);
    subplot(121)
    set(gca,'fontsize',12)
    plot(lnD,lnN,'linewidth',1), axis tight
    ylabel('$\langle \ln \hat \mu(\epsilon)\rangle,\; \ln \left(n/N\right)$',...
        'Interpreter','Latex','FontSize',14)
    xlabel('$\ln(\epsilon),\; \langle \ln(\epsilon) \rangle$','Interpreter',...
        'Latex','FontSize',14)
    legend('fixed distance', 'fixed mass','location','southeast')
    grid on, pbaspect([1.618, 1, 1])
    subplot(122)
    plot(lnD(1:end-1,:),diff(lnD).\diff(lnN),'linewidth',1)
    set(gca,'fontsize',12)
    grid on, pbaspect([1.618, 1, 1])
    xlabel('$\ln(\epsilon),\; \langle \ln(\epsilon) \rangle$','Interpreter',...
        'Latex','FontSize',14)
    ylabel('$\hat D_P, \bar D_P$','Interpreter','Latex','FontSize',14)
    legend('fixed distance', 'fixed mass','location','south')
    axis([min(lnD(1,2)) max(lnD(end,2)) 0 ...
        ceil(max(max(diff(lnD).\diff(lnN))))])
end

%% this removes distances to temporally correlated nearest neighbors
function d = removecorrs(d, id)
    d = d( abs(id - id(1)) > w );
    d = d(1:k);
end

end
```

A.4 Projective Noise Reduction Algorithm

This algorithm should work on both flow and map generated data.

```
function [ xf ] = podfilt( x, l, m, d, q, f, u )
% PODFILT: Projective nonlinear noise reduction using SVD.
```

```
% CALL:
%   [ xf ] = podfilt( x, l, m, d, q, f, u );
% REQUIRED INPUT:
%   x -- scalar time series (data) in a column or row form
%   l -- number of points in a cloud of nearest neighbors or strand length
%   m -- delay time for the phase space reconstruction (see ami.m)
%   d -- embedding dimension (see fnns.m) (d < l)
%   q -- local embedding dimension (q < d)
% OPTIONAL INPUT:
%   f -- f = 1 (default) for cloud based or f = 0 strand based POD
%   u -- u = 1 (default) do not adjust for curvature, and u = 0 adjust
% OUTPUT:
%   xf -- filtered time series in matrix form
%
% This function uses proper orthogonal decomposition to filter noisy
% nonlinear time series. The input time series is used to reconstruct phase
% space using delay coordinate embedding (generating a multivariate time
% series). To avoid edge effects, the reconstruction is continued on both
% ends by finding the nearest neighbor (nn) to the end points in the
% reconstruction and following their trajectories. Point cloud based
% projective noise reduction is based on standard published procedure (see
% Kantz/Schreiber). For strand based projection, an l-length segment of
% this vector-valued time series is called a strand. Each strand is
% decomposed using SVD and only the first q dominant modes are used for
% noise reduction. The filtering is done considering all the corrections to
% each individual sample.
%
% copyright and written by David Chelidze 03/16/2012-04/06/2012, 11/16/17
% contact email: chelidze@uri.edu

%% input check and initialization
if nargin < 5
   error('Requires at least five input arguments!\n')
elseif min(size(x)) ~= 1
   error('Error: data should be a vector of scalar data points.\n')
elseif d <= q
```

```
    error('make subspace dimension q smaller than embedding dimension d\n')
else % normalize the input time series
    x = x(:)';
    nx = m*(d-1); % shrinkage in reconstruction
    if nargin < 6 || isempty(f) % do cloud based SOD
        f = 0;
    end
    if nargin < 7 || isempty(u) % do not adjust for curvature
        u = 0;
    end
end
%% Embedding and Partitioning for the Nearest Neighbor Search
n = length(x); % original length of time series
nrp = n - nx; % number of reconstructed points
%% embedding and partitioning for nearest neighbor (nn) search
y = zeros(d, nrp); % initialize array for embedded vectors
for i = 1:d % reconstruct the phase space
    y(i,:) = x( (1:nrp) + (i-1)*m );
end % embedding
tree = createns(y','NSMethod','kdtree','Distance','euclidean');
R = eye(d,d); R(1,1) = 1000; R(d,d) = 1000; % weighting function
%% Padding (extending the reconstruction to deal with edge effects)
if f == 1 % Cloud Based Projection
    np = 1 + nx; % padding needed for the cloud based scheme
elseif l <= d
        error('Error: make string length (l) larger (l > d!)')
else
    l = floor(l/2) + 1; % strand middle point
    sl = 2*l - 1; % strand length
    np = l + nx; % padding needed for the strand based scheme
end
[pqi, ~] = knnsearch(tree,y(:,1)','k',np); % find analogue to the first pnt
pqi = pqi( pqi >  np ); % choose the acceptable nns
ya = y(:, pqi(1) - np : pqi(1) - 1); % padding from the left
[pqi, ~] = knnsearch(tree,y(:,end)','k',np); % find analogue to the last p
pqi = pqi( pqi <  nrp - np ); % choose the acceptable nns
```

```
yb = y(:, pqi(1) + 1 : pqi(1) + np); % padding from the right
yp = [ ya y yb ]; % padded phase space
npr = nrp + 2*np; % new size of the phase space
%% Projecting the Noise Out
Yh = zeros(d, npr); % stores adjusted phase space points
if f == 1 % do nearest neighbor projection
    b = zeros(d, npr); % stores the bias
    [pqi, ~] = knnsearch(tree,yp','Distance','euclidean','k',l);
    for k = 1:npr % for each reconstructed point
        Y = y(:, pqi(k,:)); % the nearest neighbors
        b(:, k) = mean(Y, 2); % bias mean vector (crude mean filter)
        [U, S, V] = svd( ( Y - b(:,k)*ones(1,l) )'*R, 0 ); % POD
        C = V(:,1:q)*S(1:q,1:q)*U(:,1:q)'; % projection
        Yh(:,k) = R\C(:,1) + b(:,k); % adjustment
    end
    if u == 0
        Yh = Yh(:,2:npr-1) + b(:,2:end-1) - (b(:,1:end-2) + b(:,3:end))/2;
    else
        Yh = Yh(:,2:npr-1);
    end
else % do the strand projection
    b = zeros(d,npr-sl+1); % stores bias
    W = [0:l-1, l-2:-1:0]; % use tent weighting for adjusting neighbors
    W = diag(W/sum(W)); % normalize
    for k = 1:npr-sl+1 % for each reconstructed point
        Y = yp(:,k+(0:sl-1)); % get the corresponding strand
        b(:,k) = mean(Y,2); % bias mean vector (crude mean filter)
        [U,S,V] = svd((Y-b(:,k)*ones(1,sl))'*R,0); % POD
        C = V(:,1:q)*S(1:q,1:q)*U(:,1:q)'; % projection
        Yh(:, k+(0:sl-1) ) = Yh(:, k+(0:sl-1) )...
                + ( R\C + b(:,k)*ones(1,sl) )*W; % adjustment with weight
    end % filtered phase space, now adjust for curvature
    if u == 0
        Yh = Yh(:,l+1:npr-l) + b(:,2:end-1) - (b(:,1:end-2)+b(:,3:end))/2;
    else
        Yh = Yh(:,l+1:npr-l);
```

```
    end
end
%% Shifting coordinates so they align in columns
xf = zeros(d,n);
for k = 1:d
    xf(k,:) = Yh(k,(m*(d-k)+1):end-m*(k-1));
end
xf = mean(xf, 1);
```

A.5 Smooth Projective Noise Reduction Algorithm

This algorithm is only intended to work on flow generated time series and will fail if applied to map generated data.

```
function [ xf ] = sodfilt( x, l, m, d, q, w )
% SODFILTER: SOD based noise reduction in nonlinear time series.
% CALL:
%   [ xf ] = sodfilt( x, l, m, d, q, w );
% REQUIRED INPUT:
%   x -- scalar time series (data) in a column or row form
%   l -- number of points in each strand (use odd number, or it will add 1)
%   m -- delay time for the phase space reconstruction (see ami.m/cds.m)
%   d -- embedding dimension (see fnns.m) (d < l!)
%   q -- local embedding dimension (q < d!!)
%  OPTIONAL INPUT PARAMETERS:
%   w -- weighting used in filtering (0--comb def, 1--tent, 2--exponential)
% OUTPUT:
%   xf -- filtered time series
%
% This function uses smooth orthogonal decomposition to filter noisy
% nonlinear time series. The input time series x is used to reconstruct
% phase space via delay coordinate embedding (generating a multivariate
% time series). To avoid edge effects, the reconstruction is continued on
% both ends by finding the nearest neighbor to the end points in the
% reconstruction and following their trajectories. An l-length segment of
% this vector-valued time series is called a strand. Each strand is
% decomposed using SOD and only the first q smooth modes are used for noise
```

```
% reduction. The filtering is done considering all the corrections to each
% individual sample.
%
% copyright and written by David Chelidze 03/16/2012 04/06/2012 11/16/17
% contact email: chelidze@uri.edu

%% input check and initialization
if nargin < 5
   error('Requires at least five input arguments!\n')
elseif min(size(x)) ~= 1
   error('Error: data should be a vector of scalar data points.\n')
else % normalize the input time series
    x = x(:)';
    nx = m*(d-1); % shrinkage in reconstruction
    n = length(x)- nx; % <- the maximum size possible!
    if nargin < 6
        w = 0;
    end
end
y = zeros(d, n); % initialize array for embedded vectors
%% check if input variables are feasible
if l <= d || isempty(l)
    error('make string length l larger than the embedding dimension d\n')
end
if d <= q
    error('make subspace dimension q smaller than embedding dimension d\n')
end
l = floor(l/2); % distance from first point to mid-point
sl = 2*l + 1; % strand length
%% embedding and partitioning for nearest neighbor (nn) search
for i = 1:d % reconstruct the phase space
    y(i,:) = x((1:n) + (i-1)*m);
end % embedding
tree = createns(y','NSMethod','kdtree','Distance','euclidean');
%% padding (extending edges for filtering and shrinkage mitigation)
np = l + nx; % padding required to deal with edge effects
```

```
[pqi, ~] = knnsearch(tree,y(:,1)','Distance','euclidean','k',np);
pqi = pqi( pqi >  np ); % find suitable nearest neighbor
ya = y(:,pqi(1)-np:pqi(1)-1); % pad the left side
[pqi, ~] = knnsearch(tree,y(:,end)','Distance','euclidean','k',np);
pqi = pqi( pqi <  n - np ); % find suitable neighbor
yb = y(:,pqi(1)+1:pqi(1)+np); % pad the right side
y = [ ya y yb ]; % padded phase space
npr = n+2*np; % new number of padded reconstructed points
%% weighting function for filtering
if w == 1 % use tent weighting function W (good results)
    W = [0:l, l-1:-1:0];
    W = diag(W/sum(W));
elseif w == 2 % use exponential weighting function W (try if you want)
    W = exp(-([l:-1:0, 1:l]));
    W = diag(W/sum(W));
else % use comb weighting function W (no averaging)
    W = diag([zeros(1,l) 1 zeros(1,l)]);
end
%% SOD filtering part
Yh = zeros(d, npr); % stores adjusted trajectory points
for k = 1:npr-sl+1 % each reconstructed point
    Y = y(:,k:k+sl-1); % get the reference strand for smoothing
    b = mean(Y, 2); % get the reference strand bias vector
    Y = Y - b*ones(1,sl); % get the unbiased reference strand
    [U,~,X,C,~] = gsvd( Y', diff(Y'), 0 ); % SOD of the original strand
    X = fliplr(X); % SOMs
    U = fliplr(U); % SOCs
    C = flipud(diag(C)); % SOVs
    Ya = X(:,1:q)*diag(C(1:q))*U(:,1:q)'; % smooth projection
    % project original strand to the smooth subspace and add back the bias
    Yh(:, k:k+sl-1) = Yh(:, k:k+sl-1)...
                    + ( Ya + b*ones(1,sl) )*W;
end
%% final averaging to get filtered time series
% trimming padded edges
Yh = Yh(:,l+1:npr-l);
```

```
%% shifting coordinates so they align in columns
xf = zeros(d,n+nx);
for k = 1:d
    xf(k,:) = Yh(k,m*(d-k)+1:end-m*(k-1));
end
xf = mean(xf, 1); % average for filtered time series
```

A.6 Largest Short-Time Lyapunov Exponent Algorithm

```
function [ ns, lnd ] = lyapunov( x, d, m, N, r, n, w )
% This function calculates metrics for the largest short-time Lyapunov
% exponent from a scalar time series after reconstructing its phase space
% using delay coordinates
%
% CALL: [ ns, lnd ] = lyapunov( x, d, m, N, r, n, w );
%   If no output is requested it will just plot the results
%
% INPUT:
%   x -- scalar time series
%   d -- embedding dimension
%   m -- delay parameter
%   N -- number of initial (fiducial) points to use (should be large)
%   r -- number of nearest neighbors to use (r = 1 should work)
%   n -- maximum number of steps ahead to proceed
%   w -- (optional) supply 0 to not remove temporal correlations
%        or number of points to remove. The default is p=d*m.
%        NOTE: only need to remove these correlations for flow data!!
%
% OUTPUT:
%   ns -- the time steps used
%   lnd -- ln of averaged distances
%   If not output is requested it plots the results. rescale results using
%   appropriate sampling time interval to get actual short-time maximal LE.
%
% (c) by David Chelidze 11/04/2005
%     modified by DC on 11/16/2005, 4/11/2011, 11/26/2012, 11/11/2013,
```

```
%     11/2/17, 12/11/17, 1/5/18

if nargin < 6
    error('Error: Needs at least six input variables!')
end
if min(size(x)) ~= 1 % check data and make a raw vector of scalar data
   error('Error: First input should be a vector of scalar data points.')
else % make row vector with zero mean and unit variance
    x = (x(:)' - mean(x))/std(x);
end
nm = length(x) - (d-1)*m; % max number of points possible in phase space
if nargin < 7 || isempty(w)
    w = d*m; % remove temporal correlations of this length, open for debate
end
if N == 0 || isempty(N) % set the default
    N = min(2^12, nm);
else % use the provided or max possible points
    N = min(N, nm);
end
% reconstruct the phase space
y = zeros(d, nm);
for i = 0:d-1
    y(i+1,:) = x( (1:nm) + i*m );
end

% expected nearest neighbor distance for noise in d-dimensions (used in w)
a = sqrt(2)*gamma((d+1)/2)*(sqrt(d)/nm)^(1/d)/gamma(d/2);

% define N random initial conditions for 'fiducial' points
indx = randperm(nm-n,N);
lnd = zeros(1,n); % initialize log of average distance array

% partition all suitable phase space points for fast searching
ns = createns(y(:,1:end-n)','NSMethod','kdtree','Distance','euclidean');
% search for the k points nearest to the test points (+ 2*w if removing
% temporal correlations)
```

```
np = r + 2*w + 1; % total nearest neighbors to look up
[idx, dis] = knnsearch(ns,y(:,indx)','Distance','euclidean','k',np);
if w == 0 % do not remove temporarily correlated nearest neighbor distances
    dis = dis(:,2:end); % only drop the first zero distance
    idx = idx(:,2:end);
else % remove temporal correlations
    [dis, idx] = cellfun(@removecorrs, mat2cell(dis,ones(N,1),np),...
                 mat2cell(idx,ones(N,1),np),'UniformOutput',false);
    dis = cell2mat(dis);
    idx = cell2mat(idx);
end

for k = 1:N % for each fiducial trajectory
    W = a ./ ( (dis(k,:)-dis(1,:)).^4 + a ); % weighting function
    dist = zeros(1, n); % initialize local distances
    for inn = 1:r % all nearest neighbor trajectories
        % find the difference between nn and fiducial trajectories
        tmp = y(:, idx(k,inn)+(1:n)) - y(:, indx(k)+(1:n));
        % normalized distances between trajectories:
        dist = dist + W(inn)*sqrt(sum(tmp.*tmp));
    end
    lnd = lnd + log(dist/sum(W)); % log of average distance
end % k loop
lnd = lnd/N; % normalize by the number of reference/fiducial points
ns = 1:n; % just how far we went

if nargout == 0 % plot the results
    figure
    set(gcf,'PaperSize',[10.5 3.5],'PaperPosition', [-.75 -.25 12 4]);
    subplot(121)
    set(gca,'fontsize',12)
    plot(ns,lnd,'linewidth',1), axis tight
    grid on, pbaspect([1.618, 1, 1])
    xlabel('$n$','Interpreter','Latex','FontSize',14)
    ylabel('$\Delta_n = \langle \ln \delta_n - \ln \delta_0 \rangle$',...
        'Interpreter','Latex','FontSize',14)
```

```
    subplot(122)
    set(gca,'fontsize',12)
    plot(diff(ns).\diff(lnd),'linewidth',1), axis tight
    grid on, pbaspect([1.618, 1, 1])
    xlabel('$n$','Interpreter','Latex','FontSize',14)
    ylabel('$\Delta t \hat\lambda_1=\mathrm{d}\Delta_n/\mathrm{d}n$',...
        'Interpreter','Latex','FontSize',14)
end
%% this removes temporally correlated nearest neighbors
function [d, id] = removecorrs(d, id)
    cond =  abs(id - id(1)) > w;
    d = d(cond);
    id = id(cond);
    d = d(1:r);
    id = id(1:r);
end
end
```

A.7 Sample MATLAB Function for Local Linear Model Prediction

```
function yp = loclin( ym, yq, n, N, w )
% LOCLIN local linear modeling for prediction
%
% INPUT:
%   ym -- reconstructed phase space points (trajectory) used for modeling
%         should be d times M, where M is the number of points
%   yq -- initial point to start the prediction from
%         should be d times 1
%   n -- number of nearest neighbors to use for prediction
%   N -- how far ahead to do the prediction (number of time steps)
%   w -- optional, to use weighting w = 1, otherwise use no weighting
% OUTPUT:
%   yp -- predicted phase space trajectory starting from yq point
%
% EXAMPLE:
```

```
%   x = x(:)'; % experimental time series or generated from some model
%   np = size(x,2) - (d-1)*m; % maximum number of reconstructed points
%   y = zeros(d,np); % reconstructed phase space trajectory
%   for k = 1:d
%       y(k,:) = x((1:np) + (k-1)*m);
%   end
%   % use half points for modeling and other half for testing:
%   yp = loclin( y(:,1:np/2), y(:,np/2 + 10), 8, 1000 );
%   plot(yp(1,:)), hold on, plot(y(1,n/2+10:n/2+1000))
%
% written and copyrighted by David Chelidze on 12/4/17

[d, nm] = size(ym);
ns = createns(ym(:,1:end-1)','NSMethod','kdtree','Distance','euclidean');
yp = [yq zeros(d,N)];
if nargin < 5 || w ~= 1 % use no weighting
    w = 0;
elseif w == 1 % expected nearest neighbor distance for noise in d-dims:
    a = sqrt(2)*gamma((d+1)/2)*(sqrt(d)/nm)^(1/d)/gamma(d/2);
end

for k = 1:N
    [indx, dis] = knnsearch(ns,yq','Distance','euclidean','k',n);
    Yn = [ym(:,indx); ones(1,n)];
    Ynp1 = ym(:,indx+1);
    if w == 1 % use weighting
        W = diag(a ./ ( dis.^4 + a )); % weighting fun (can make your own)
        yp(:,k+1) = ((Ynp1*W*Yn')/(Yn*W*Yn'))*[yq; 1];
    else
        yp(:,k+1) = ((Ynp1*Yn')/(Yn*Yn'))*[yq; 1];
    end
    yq = yp(:,k+1);
end
```

A.8 Iteratively Refined Surrogates

```
function [rb, sb] = irsurr(s, n)
% IRSURR generates the iteratively refined surrogates of the original time
% series.
%
% USE: >> [rb, sb] = irsurr(s, n);
%
% INPUT:
%   x -- original scalar time series
%   n -- number of iterations to do
%
% OUTPUT:
%   rb -- surrogates that exactly match in histogram (PDF)
%   sb -- surrogates that exactly match in FFT magnitudes (PSD)
%
% Written by David Chelidze on 7/12/18 according to the paper:
%
%   Schreiber, T., Schmitz, A., 2000. Surrogate time series.
%   Physica D 142 (3-4), 346-382.

s = s(:); % make a column vector
N = size(s,1); % get the length of the time series

c = sort(s); % sort and store the magnitudes in ascending order
S = abs( fft(s) ); % store the FFT magnitudes

% initialize the iterations by phase randomizing the original signal
rb = real(ifft(fft(s).*exp(1i*2*pi*rand(N,1))));

for k = 1:n
    sb = real(ifft( S.*exp(1i*angle(fft(rb))) )); % first match the PSD
    [~, indx] = sort(sb);  % now match the PDF
    rb(indx) = c;          % by substituting the original magnitudes
end
```

Bibliography

[1] Henry Abarbanel". Local false nearest neighbors and dynamical dimensions from observed chaotic data. *Physical Review E*, 47(5):3057–3068, 1993.

[2] Henry Abarbanel. *Analysis of observed chaotic data*. Springer, New York, 1996.

[3] Henry Abarbanel. *Analysis of observed chaotic data*. Institute for nonlinear science. Springer, 1996.

[4] Jonathan B.Dingwell. *Lyapunov Exponents*. John Wiley and Sons, Inc., 2006.

[5] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[6] J. Brocker, U. Parlitz, and M. Ogorzalek. Nonlinear noise reduction. *Proceedings of the IEEE*, 90(5):898–918, 2002.

[7] David S Broomhead and Gregory P King. Extracting qualitative dynamics from experimental data. *Physica D: Nonlinear Phenomena*, 20(2):217–236, 1986.

[8] Liangyue Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena*, 110(1):43–50, 1997.

[9] Liangyue Cao, Alistair Mees, and Kevin Judd. Dynamics from multivariate time series. *Physica D: Nonlinear Phenomena*, 121(1):75–88, 1998.

[10] James R Carey. Demography and population dynamics of the mediterranean fruit fly. *Ecological Modelling*, 16(2-4):125–150, 1982.

[11] Martin Casdagli, Tim Sauer, and James A Yorke. Embedology. *Journal of Statistical Physics*, 65(3/4):579–616, 1991.

[12] A. Chatterjee, J. P. Cusumano, and D. Chelidze. Optimal tracking of parameter drift in a chaotic system: Experiment and theory. *Journal of Sound and Vibration*, 250(5):877–901, 2002.

[13] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current Science: Computational Science*, 78(7):808–817, 2000.

[14] C Chatzinakos and C Tsouros. Estimation of the dimension of chaotic dynamical systems using neural networks and robust location estimate. *Simulation Modelling Practice and Theory*, 51:149–156, 2015.

[15] D. Chelidze and M. Liu. Reconstructing slow-time dynamics from fast-time measurements. *Philosophical Transaction of the Royal Society A*, 366:729–3087, 2008.

[16] D. Chelidze and W. Zhou. Smooth orthogonal decomposition-based vibration mode identification. *Journal of Sound and Vibration*, 292(3):461–473, 2006.

[17] David Chelidze. Statistical analysis of nearest neighbors leads to robust and reliable estimates for minimum embedding dimension. *Journal of Computational & Nonlinear Dynamics*, (CND-16-1488), Submitted 2016.

[18] David Chelidze and Joseph P. Cusumano. Phase space warping: Nonlinear time series analysis for slowly drifting systems. *Philosophical Transactions of the Royal Society A*, 364, 2006.

[19] David Chelidze, Joseph P. Cusumano, and Anindya Chatterjee. Dynamical Systems Approach to Damage Evalution Tracking, Part I: Desccription and Experimental Application. *Journal of Vibration and Acoustics*, 124(2):250–257, 2002.

[20] David Chelidze and Ming Liu. Dynamical systems approach to fatigue damage identification. *Journal of Sound and Vibration*, 281:887–904, 2004.

[21] David Chelidze and Ming Liu. Multidimensional damage identification based on phase space warping: An experimental study. *Nonlinear Dynamics*, 46(1–2):887–904, 2006.

[22] Antoine Clement and Stephane Laurens. An alternative to the Lyapunov exponent as a damage sensitive feature. *Smart Materials and Structures*, 20(2), 2011.

[23] P Craven and G. Wahba. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:377–403, 1979.

[24] JP Cusumano and BW Kimble. A stochastic interrogation method for experimental measurements of global dynamics and basin evolution: Application to a two-well oscillator. *Nonlinear Dynamics*, 8(2):213–235, 1995.

[25] JP Cusumano and D-C Lin. Bifurcation and modal interaction in a simplified model of bending-torsion vibrations of the thin elastica. *Journal of vibration and acoustics*, 117(1):30–42, 1995.

[26] Reik V. Donner and Susana M. Barbosa, editors. *Nonlinear Time Series Analysis in the Geosciences - Applications in Climatology, Geodynamics and Solar-Terrestrial Physics*. Berlin: Springer, 2008.

[27] James Eells, Hassler Whitney, and Domingo Toledo. *Collected papers of Hassler Whitney*. Nelson Thornes, 1992.

[28] M. EtehadTavakol, E Y K Ng, C. Lucas, S. Sadri, and M. Ataei. Nonlinear analysis using Lyapunov exponents in breast thermograms to identify abnormal lesions. *Infrared Physics & Technology*, 55(4):345 – 352, 2012.

[29] J Doyne Farmer, Edward Ott, and James A Yorke. The dimension of chaotic attractors. *Physica D: Nonlinear Phenomena*, 7(1-3):153–180, 1983.

[30] U. Farooq and BF Feeny. Smooth orthogonal decomposition for modal analysis of randomly excited systems. *Journal of Sound and Vibration*, 316(1):137–146, 2008.

[31] Andrew M Fraser. Reconstructing attractors from scalar time series: a comparison of singular system and redundancy criteria. *Physica D: Nonlinear Phenomena*, 34(3):391–404, 1989.

[32] Andrew M Fraser and Harry L Swinney. Independent coordinates for strange attractors from mutual information. *Physical review A*, 33(2):1134, 1986.

[33] J. Gao, H. Sultan, J. Hu, and W.W. Tung. Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: a comparison. *Signal Processing Letters, IEEE*, 17(3):237–240, 2010.

[34] S. Ghafari, F. Golnaraghi, and F. Ismail. Effect of localized faults on chaotic vibration of rolling element bearings. *Nonlinear Dynamics*, 53:287–301, 2008.

[35] John F Gibson, J Doyne Farmer, Martin Casdagli, and Stephen Eubank. An analytic approach to practical state space reconstruction. *Physica D: Nonlinear Phenomena*, 57(1):1–30, 1992.

[36] Gene H Golub and Charles F Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[37] Peter Grassberger and Itamar Procaccia. Characterization of strange attractors. *Physical review letters*, 50(5):346, 1983.

[38] Peter Grassberger, Thomas Schreiber, and Carsten Schaffrath. Nonlinear time sequence analysis. *International Journal of Bifurcation and Chaos*, 1(03):521–547, 1991.

[39] Allan Gut. *An intermediate course in probability*. Springer Texts in Statistics. Springer, New York, NY, 2009.

[40] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[41] Karen M Hampson and Edward AH Mallen. Chaos in ocular aberration dynamics of the human eye. *Biomedical optics express*, 3(5):863–877, 2012.

[42] Arnold Hanslmeier, Roman Brajša, Jaša Čalogović, Bojan Vršnak, Domagoj Ruždjak, Friedhelm Steinhilber, CL MacLeod, Željko Ivezić, and Ivica Skokić. The chaotic solar cycle-ii. analysis of cosmogenic 10be data. *Astronomy & Astrophysics*, 550:A6, 2013.

[43] Rainer Hegger and Holger Kantz. Improved false nearest neighbor method to detect determinism in time series data. *Physical Review E*, 60(4):4970, 1999.

[44] Rainer Hegger, Holger Kantz, and Thomas Schreiber. Practical implementation of nonlinear time series methods: The tisean package. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(2):413–435, 1999.

[45] Michel Hénon. A two-dimensional mapping with a strange attractor. In *The Theory of Chaotic Attractors*, pages 94–102. Springer, 1976.

[46] Behshad Hosseinifard, Mohammad Hassan Moradi, and Reza Rostami. Classifying depression patients and normal subjects using machine learning techniques and nonlinear features from {EEG} signal. *Computer Methods and Programs in Biomedicine*, 109(3):339 – 345, 2013.

[47] Tijana Ivancevic, Lakhmi Jain, John Pattison, and Alex Hariz. Nonlinear dynamics and chaos methods in neurodynamics and complex data analysis. *Nonlinear Dynamics*, 56(1-2):23–44, 2009.

[48] N Jevtic, JS Schweitzer, and P Stine. Optimizing nonlinear projective noise reduction for the detection of planets in mean-motion resonances in transit light curves. *CHAOS2010*, page 191, 2011.

[49] Kevin Judd and Alistair Mees. Embedding as a modeling problem. *Physica D: Nonlinear Phenomena*, 120(3):273–286, 1998.

[50] S Kacimi and S Laurens. The correlation dimension: A robust chaotic feature for classifying acoustic emission signals generated in construction materials. *Journal of Applied Physics*, 106(2), 2009.

[51] H. Kantz and T. Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge University Press, 2003.

[52] H. Kantz, T. Schreiber, I. Hoffmann, T. Buzug, G. Pfister, L.G. Flepp, J. Simonet, R. Badii, and E. Brun. Nonlinear noise reduction: A case study on experimental data. *Physical Review E*, 48(2):1529, 1993.

[53] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*. Cambridge University Press, Cambridge, UK, 2 edition, 2004.

[54] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 2004.

[55] Ahmad Kazem, Ebrahim Sharifi, Farookh Khadeer Hussain, Morteza Saberi, and Omar Khadeer Hussain. Support vector regression with chaos-based firefly algorithm for stock market price forecasting. *Applied Soft Computing*, 13(2):947–958, 2013.

[56] G Kember and AC Fowler. A correlation function for choosing time delays in phase portrait reconstructions. *Physics Letters A*, 179(2):72–80, 1993.

[57] Matthew B Kennel and Henry DI Abarbanel. False neighbors and false strands: A reliable minimum embedding dimension algorithm. *Physical review E*, 66(2):026209, 2002.

[58] Matthew B Kennel, Reggie Brown, and Henry DI Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical review A*, 45(6):3403, 1992.

[59] Gaetan Kerschen, Jean-Claude Golinval, Alexander F. Vakakis, and Lawrence A. Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview. *Nonlinear Dynamics*, 41:147–169, 2005.

[60] H_S Kim, R Eykholt, and JD Salas. Nonlinear dynamics, delay times, and embedding windows. *Physica D: Nonlinear Phenomena*, 127(1):48–60, 1999.

[61] E.J. Kostelich and T. Schreiber. Noise reduction in chaotic time-series data: a survey of common methods. *Physical Review E*, 48(3):1752, 1993.

[62] Ivana Kovacic and Michael J Brennan. *The duffing equation: Nonlinear oscillators and their behaviour*. Wiley, 2011.

[63] Dimitris Kugiumtzis. State space reconstruction parameters in the analysis of chaotic time seriesâthe role of the time window length. *Physica D: Nonlinear Phenomena*, 95(1):13–28, 1996.

[64] W. Liebert, K. Pawelzik, and HG Schuster. Optimal embeddings of chaotic attractors from topological considerations. *EPL (Europhysics Letters)*, 14(6):521, 2007.

[65] Henrik Madsen. *Time series analysis*. CRC Press, 2007.

[66] T Matsumoto, Leon O Chua, and S Tanaka. Simplest chaotic nonautonomous circuit. *Physical Review A*, 30(2):1155, 1984.

[67] Takashi Matsumoto. A chaotic attractor from chua's circuit. *IEEE Transactions on Circuits and Systems*, 31(12):1055–1058, 1984.

[68] Takashi Matsumoto, Leon Chua, and Motomasa Komuro. The double scroll. *IEEE Transactions on Circuits and Systems*, 32(8):797–818, 1985.

[69] A Maus and JC Sprott. Neural network method for determining embedding dimension of a time series. *Communications in Nonlinear Science and Numerical Simulation*, 16(8):3294–3302, 2011.

[70] FC Moon and Philip J Holmes. A magnetoelastic strange attractor. *Journal of Sound and Vibration*, 65(2):275–296, 1979.

[71] FC Moon and Philip J Holmes. A magnetoelastic strange attractor. *Journal of Sound and Vibration*, 65(2):275–296, 1979.

[72] Francis C Moon and FC Moon. *Chaotic vibrations: an introduction for applied scientists and engineers*. Wiley Online Library, 1987.

[73] Jonathan Nichols and Michael Todd. *Nonlinear Features for SHM Applications*. John Wiley & Sons, Ltd, 2009.

[74] Muruhan Rathinam and Linda R. Petzold. A new look at proper orthogonal decomposition. *SIAM J. Numer. Anal.*, 41(5):1893–1925, 2003.

[75] C.H. Reinsch. Smoothing by spline fiunctions. *Numer. Math.*, 10:177–183, 1967.

[76] C.H. Reinsch. Smoothing by spline fiunctions ii. *Numer. Math.*, 16:451–454, 1971.

[77] Carl Rhodes and M Morari. False-nearest-neighbors algorithm and noise-corrupted time series. *Physical Review E*, 55(5):6162–6170, 1997.

[78] Michael T Rosenstein, James J Collins, and Carlo J De Luca. Reconstruction expansion as a geometry-based framework for choosing proper delay times. *Physica D: Nonlinear Phenomena*, 73(1):82–98, 1994.

[79] Tim Sauer, James A Yorke, and Martin Casdagli. Embedology. *Journal of statistical Physics*, 65(3-4):579–616, 1991.

[80] Jeffrey M Schiffman, David Chelidze, Albert Adams, David B Segala, and Leif Hasselquist. Nonlinear analysis of gait kinematics to track changes in oxygen consumption in prolonged load carriage walking: A pilot study. *Journal of biomechanics*, 42(13):2196–2199, 2009.

[81] I.J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math.*, 4:45–99 and 112–141, 1946.

[82] Thomas Schreiber and Andreas Schmitz. Improved surrogate data for nonlinearity tests. *Physical Review Letters*, 77(4):635, 1996.

[83] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[84] Michael Small and Chi K Tse. Optimal embedding parameters: a modelling paradigm. *Physica D: Nonlinear Phenomena*, 194(3):283–296, 2004.

[85] Miao Song, David B Segala, Jonathan B Dingwell, and David Chelidze. Slow-time changes in human emg muscle fatigue states are fully represented in movement kinematics. *Journal of biomechanical engineering*, 131(2):021004, 2009.

[86] Julien Clinton Sprott and Julien C Sprott. *Chaos and time-series analysis*, volume 69. Oxford University Press Oxford, 2003.

[87] Ian Stewart. Mathematics: The lorenz attractor exists. *Nature*, 406(6799):948–949, 2000.

[88] James Theiler. Statistical precision of dimension estimators. *Physical Review A*, 41(6):3038, 1990.

[89] James Theiler. Some comments on the correlation dimension of $1/f^{\alpha}$ noise. *Physics Letters A*, 155(8):480–493, 1991.

[90] PE Tikkanen. Nonlinear wavelet and wavelet packet denoising of electrocardiogram signal. *Biological cybernetics*, 80(4):259–267, 1999.

[91] M D Todd, J M Nichols, L M Pecora, and L N Virgin. Vibration-based damage assessment utilizing state space geometry changes: local attractor variance ratio. *Smart Materials and Structures*, 10(5):1000–1008, 2001.

[92] W. J. Wang, Z. T. Wu, and J. Chen. Fault identification in rotating machinery using the correlation dimension and bispectra. *Nonlinear Dynamics*, 25(4):383–393, 2001.

[93] W.J. Wang, J. Chen, X.K. Wu, and Z.T. Wu. The application of some non-linear methods in rotating machinery fault diagnosis. *Mechanical Systems and Signal Processing*, 15(4):697 – 705, 2001.

[94] BA Wernitz and NP Hoffmann. Recurrence analysis and phase space reconstruction of irregular vibration in friction brakes: Signatures of chaos in steady sliding. *Journal of Sound and Vibration*, 331(16):3887–3896, 2012.

[95] Alan Wolf. *Quantifying chaos with Lyapuonov exponents*, pages 273–290. Princeton University Press, 1986.

[96] M. Pollicott Yuri. *Dynamical Systems and Ergodic Theory*. Cambridge University Press, 1998.