Time Series Analysis

Elizabeth Bradley

Department of Computer Science
University of Colorado
Boulder, Colorado, USA 80309-0430 `lizb@cs.colorado.edu`

Chapter 5 of *Intelligent Data Analysis: An Introduction*
D. Hand and M. Berthold, eds.
2nd edition, Springer-Verlag, 2002.

Intelligent data analysis often requires one to extract meaningful conclusions about a complicated system using time-series data from a single sensor. If the system is linear, a wealth of well-established, powerful techniques is available to the analyst. If it is not, the problem is much harder and one must resort to nonlinear dynamics theory in order to infer useful information from the data. Either way, the problem is often complicated by a simultaneous overabundance and lack of data: megabytes of time-series data about the voltage output of a power substation, for instance, but no information about other important quantities, such as the temperatures inside the transformers. Data-mining techniques[16] provide some useful ways to deal successfully with the sheer volume of information that constitutes one part of this problem. The second part of the problem is much harder. If the target system is highly complex—say, an electromechanical device whose *dynamics* is governed by three metal blocks, two springs, a pulley, several magnets, and a battery—but only one of its important properties (e.g., the position of one of the masses) is sensor-accessible, the data analysis procedure would appear to be fundamentally limited.

Fig. 1 shows a simple example of the kind of problem that this chapter addresses: a mechanical spring/mass system and two time-series data sets gathered by sensors that measure the position and velocity of the mass. This system is linear: it responds *in proportion to* changes. Pulling the mass twice as far down, for instance, will elicit an oscillation that is twice as large, not one that is $2^{1.5}$ as large or $\log 2$ times as large. A pendulum, in contrast, reacts *nonlinearly*: if it is hanging straight down, a small change in its angle will have little effect, but if it is balanced at the inverted point, small changes have large effects. This distinction is extremely important to science in general and data analysis in particular. If the system under examination is linear, data analysis is comparatively straightforward and the tools—the topic of section 1 of this chapter—are well developed. One can characterize the data using statistics (mean, standard deviation, etc.), fit curves to them (functional approximation), and plot various kinds of graphs to aid one's understanding of the behavior. If a more-detailed analysis is required, one typically represents the system in an "input + transfer function → output" manner using any of a wide variety of time- or frequency-domain models. This kind of formalism admits a large collection of powerful reasoning techniques, such as superposition and the notion of transforming back and forth between the time and frequency domains. The latter is particularly powerful, as many signal processing operations are much easier in one domain than the other.

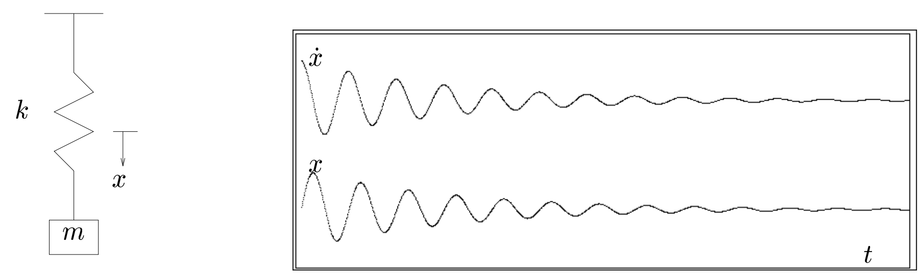


Figure 1: A simple example: A spring/mass system and a time series of the vertical position and velocity of the mass, measured by two sensors

Nonlinear systems pose an important challenge to intelligent data analysis. Not only are they ubiquitous in science and engineering, but their mathematics is also vastly harder, and many standard time-series analysis techniques simply do not apply to nonlinear problems. Chaotic systems, for instance, exhibit broad-band behavior, which makes many traditional signal processing operations useless. One cannot decompose chaotic problems in the standard "input + transfer function $\rightarrow$ output" manner, nor can one simply low-pass filter the data to remove noise, as the high-frequency components are essential elements of the signal. The concept of a discrete set of spectral components does not make sense in many nonlinear problems, so using transforms to move between time and frequency domains—a standard technique that lets one transform differential equations into algebraic ones and vice versa, making the former much easier to work with—does not work. For these and related reasons, nonlinear dynamicists eschew most forms of spectral analysis. Because they are soundly based in nonlinear dynamics theory and rest firmly on the formal definition of invariants, however, the analysis methods described in section 2 of this chapter do not suffer from the kinds of limitations that apply to traditional linear analysis methods.

Another common complication in data analysis is *observability*: whether or not one has access to enough information to fully describe the system. The spring/mass system in Fig. 1, for instance, has two *state variables*—the position and velocity of the mass—and one must measure both of them in order to know the state of the system. (One can, to be sure, reconstruct velocity data from the position time series in the Figure using divided differences[1], but that kind of operation magnifies noise and numerical error, and thus is impractical.) Delay-coordinate embedding is one way to get around this problem; it lets one reconstruct the internal dynamics of a complicated nonlinear system from a *single* time series—e.g. inferring useful information about internal (and unmeasurable) transformer temperatures from their output voltages. The reconstruction produced by delay-coordinate embedding is not, of course, completely equivalent to the internal dynamics in all situations, or embedding would amount to a general solution to control theory's *observer problem*: how to identify all of the internal state variables of a system and infer their values from the signals that *can* be observed. However, a single-sensor reconstruction, if done right, can still be extremely useful because its results are guaranteed to be *topologically* (i.e., qualitatively) identical to the internal dynamics. This means that conclusions drawn about the reconstructed dynamics are also true of the internal dynamics of the system inside the black box. All of this is important for intelligent data analysis because fully *observable* systems are rare in science and engineering practice; as a rule, many—often, most—of a system's state variables either are physically inaccessible or cannot be measured with available sensors. Worse yet, the true state variables may not be known to the user; temperature, for instance, can play an important and often unanticipated role in the behavior of an electronic circuit. The delay-coordinate embedding methods covered in section 3 of this chapter not only yield useful information about the behavior of the unmeasured variables, but also give some indication of how many independent state variables actually exist inside the black box.

Although the vast majority of natural and man-made systems is nonlinear, almost all textbook time-series analysis techniques are limited to linear systems. The objective of this chapter is to present a more broadly useful arsenal of time-series analysis techniques—tools that can be applied to *any* system, linear or nonlinear. The techniques that have been developed by the nonlinear dynamics community over the past decade play a leading role in this presentation, but many other communities have developed different approaches to nonlinear time-series analysis. One of the more famous is Tukey's "exploratory data analysis," a sleuthing approach that emphasizes (and supports) visual examination over blind, brute-force digestion of data into statistics and regression curves[50]. Some of the more-recent developments in this field attempt to aid—or even augment—the analyst's abilities in unconventional ways, ranging from 3D virtual-reality displays to haptics (representing the data as a touch pattern, which has been proposed for reading mammograms[30]) or data sonification.

The sections that follow are organized as follows. Section 1 quickly reviews some of the traditional methods that apply to *linear* systems. Section 2 covers the bare essentials of dynamical systems theory and practice, with a specific emphasis on how those techniques are useful in IDA applications. This material forms the basis of the general theory of dynamics that applies to any system, linear or nonlinear. If all of the important properties of the target system can be identified and measured and the data are basically noise-

[1]e.g., dividing the difference between successive positions by the time interval between the measurements

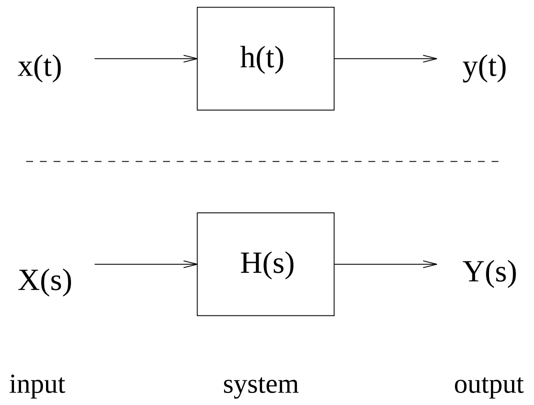


Figure 2: The "input + transfer function → output" framework of traditional signal processing. Top: time domain. Bottom: frequency domain.

free, these techniques, alone, can provide a very good solution to many nonlinear data-analysis problems. If there are fewer sensors than state variables, however, one must call upon the methods described in section 3 in order to reconstruct the dynamics before one can apply the section 2 methods. Noise is a much more difficult problem. There exist techniques that "filter" nonlinear time-series data, turning the nonlinearity to advantage and reducing the noise by a exponentially large factor[15], but the mathematics of this is well beyond the scope of this discussion. This chapter continues with two extended examples that demonstrate both the analysis methods of section 2 and the delay-coordinate reconstruction techniques of section 3, and concludes with some discussion of the utility of these methods in intelligent data analysis.

1 Linear Systems Analysis

The basic framework of traditional signal analysis[43] is schematized in Fig. 2; in it, an input signal is applied to a system to yield an output. One can describe this process in the time domain, using the *impulse response* $h(t)$ to model the system, or in the frequency domain, using the *frequency response* transfer function $H(s)$. The impulse response of a system is its transient response to a quick kick ($x(t_0) = 1$; $x(t) = 0 \,\forall\, t \neq t_0$); the frequency response $H(s)$ describes, for all s, what the system does to a sinusoidal input of frequency s. $H(s)$ is a complex function; it is most frequently written (and plotted) in magnitude ($|H(s)|$) and angle ($\angle H(s)$) form, but sometimes appears as $Re\{H(s)\}$ and $Im\{H(s)\}$.

Decomposing a problem in this "input + transfer function → output" manner is very useful; among other things, it allows one to apply powerful reasoning techniques like superposition[2]. The problem with Fig. 2 is that systems can react very differently to different inputs at different times—that is, $h(t)$ and $H(s)$ may depend on the magnitude of x, or they may have time-dependent coefficients. Either situation negates almost all of the advantages of both parts of the framework shown in the Figure. Nonlinearity (the former case) and nonstationarity (the latter) are treated later in this chapter; in the remainder of this section, we assume linearity and time invariance.

The top paradigm in Fig. 2 is easier to think about, but the bottom is mathematically much easier to work with. In particular, deriving $y(t)$ from $x(t)$ and $h(t)$ involves a convolution:

$$\begin{aligned} y(t) &= x(t) * h(t) \\ &= \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \end{aligned}$$

[2]If the inputs x_1 and x_2 produce the outputs y_1 and y_2, respectively, then the input $x_1 + x_2$ will produce the output $y_1 + y_2$.

whereas the frequency-domain calculation only requires multiplication:

$$Y(s) = X(s)H(s)$$

The frequency domain has a variety of other powerful features. The spectrum is easy to interpret; the peaks of $|H(s)|$ correspond to the natural frequencies ("modes") of the system and hence, loosely speaking, to the number of degrees of freedom. Differential equations become algebraic equations when transformed into the frequency domain, and signal separation is a trivial operation. Because of these advantages, engineers are trained to transform problems into the frequency domain, perform any required manipulations (e.g., filtering) in that domain, and then reverse-transform the results back into the time domain.

Traditional analysis methods characterize a linear system by describing $h(t)$ or $H(s)$. Depending on the demands of the application, this description—the "model"—can range from the highly abstract to the very detailed:

1. descriptive models: e.g., the sentence "as water flows out of a bathtub, the level in the tub decreases"
2. numerical models: a table of the water level in the tub versus time
3. graphical models: the same information, but in pictorial form
4. statistical models: the mean, standard deviation, and/or trend of the water level
5. functional models: a least-squares fit of a line to the water level data
6. analytic models: an equation, algebraic or differential, that relates outflow and water level

The simplicity of the first item on the list is deceptive. Qualitative models like this are quite powerful—indeed, they are the basis for most human reasoning about the physical world. A circuit designer, for instance, reasons about the *gain-bandwidth tradeoff* of a circuit, and understands the system in terms of a balance between these two quantities: "if the gain goes up, the bandwidth, and hence the speed, goes down...". Many traditional analysis methods are also based on qualitative models. One can, for instance, compute the location of the natural frequencies of a system from the ring frequency and decay time of its impulse response $h(t)$ or the shape of its frequency response $H(s)$; the latter also lets one compute the speed (rise time) and stability (gain or phase margin) of the system. *Step* and *ramp* response—how the system reacts to inputs of the form

$$\begin{aligned} x(t) &= 0 \qquad t < 0 \\ x(t) &= 1 \qquad t \geq 0 \end{aligned}$$

and

$$\begin{aligned} x(t) &= 0 \qquad t < 0 \\ x(t) &= t \qquad t \geq 0 \end{aligned}$$

respectively—also yield useful data analysis results; see [41] for details. Though qualitative models are very powerful, they are also very difficult to represent and work with explicitly; doing so effectively is the focus of the qualitative reasoning/qualitative physics community[52].

As noted and discussed by many authors (e.g., [49]), tables of numbers are much more useful to humans when they are presented in graphical form. For this reason, numerical models—item 2 in the list above—are rarely used, and many IDA researchers, among others, have devoted much effort to finding and codifying systematic methods for portraying a data set graphically and highlighting its important features. Another way to make numbers more useful is to digest them into statistical values[53] like means, medians, and standard deviations, or to use the methods of functional approximation (e.g., chapter 10 of [20]) and regression to fit some kind of curve to the data. Statisticians sometimes apply transformations to data sets for the purpose of stabilizing the variance or forcing the distribution into a normal form. These methods—which can be found in any basic text on statistical methods, such as [36]—can make data analysis easier, but one has to remember how the transformed data have been manipulated and be careful not to draw unwarranted conclusions from it. It can also be hard to know what transformation to apply in a given situation; Box and Cox developed a formal solution to this, based on a parametric family of power transforms[5].

Sometimes, none of these abstractions and approximations is adequate for the task at hand and one must use an analytic model. Again, these come in many flavors, ranging from algebraic expressions to partial differential equations. One of the simplest ways to use an algebraic equation to describe a system's behavior is to model its output as a weighted sum of its current and previous inputs. That is, if one has a series of values $\{x_i(t)\}$ of some system input x_i—e.g., the position of a car's throttle, measured once per second—one predicts its output y (the car's speed) using the equation:

$$y(t) = \sum_{l=0}^{L} b_l x_i(t-l) \tag{1}$$

The technical task in fitting such an L^{th}*-order moving average* (MA) model to a data set involves choosing the window size L and finding appropriate values for the b_l. A weighted average of the last L values is a simple smoothing operation, so this equation represents a low-pass filter. The impulse response of such a filter—again, how it responds to a quick kick—is described by the coefficients b_l: as l goes from 0 to L, the impulse first "hits" b_0, then b_1, and so on. Because this response dies out after L timesteps, equation (1) is a member of the class of so-called *finite impulse response* (FIR) filters.

Autoregressive (AR) models are similar to MA models, but they are designed to account for feedback, where the output depends not only on the inputs, but also on the previous output of the system:

$$y(t) = \sum_{m=0}^{M} a_m y(t-m) + x_i(t) \tag{2}$$

Feedback loops are common in both natural and engineered systems; consider, for instance, a cruise control whose task is to stabilize the speed of a car at 100 kph by manipulating the throttle control. Traditional control strategies for this problem measure the difference between the current output and the desired set point, then use that difference to compute the input—e.g., opening the car's throttle x in proportion to the difference between the output y and the desired speed. Feedback also has many important implications for stability, in part because the loop from output to input means that the output y can continue to oscillate indefinitely even if the input is currently zero. (Consider, for example, the AR model $y(t) = -y(t-1)+x(t)$ if $x = 0$.) For this reason, AR models are sometimes called *infinite impulse response* (IIR) filters. The dependence of $y(t)$ on previous values of y also complicates the process of finding coefficients a_m that fit the model to a data set; see, e.g., [6] for more details.

The obvious next step is to combine MA and AR models:

$$y(t) = \sum_{l=0}^{L} b_l x_i(t-l) + \sum_{m=0}^{M} a_m y(t-m) \tag{3}$$

This "ARMA" model is both more general and more difficult to work with than its predecessors; one must choose L and M intelligently and use frequency-transform methods to find the coefficients; see [6] for this methodology. Despite these difficulties, ARMA models and their close relatives have "dominated all areas of time-series analysis and discrete-time signal processing for more than half a century"[51].

Models like those in the ARMA family capture the input/output behavior of a system. For some tasks, such as controller design, input/output models are inadequate and one really needs a model of the internal dynamics: a differential equation that accounts for the system's dependence on present and previous states. As an example, consider the spring/mass system of Fig. 1. If x is the deformation of the spring from its natural length, one can write a force balance at the mass as follows:

$$\begin{aligned} \Sigma F &= ma \\ mg - kx &= ma \end{aligned}$$

Acceleration a is the second derivative of position ($a = x''$) and both are functions of time, so the force-balance equation can be rewritten as:

$$mx(t)'' = mg - kx(t) \tag{4}$$

This linear[3] differential equation expresses a set of constraints among the derivatives of an unknown function $x(t)$ and a set of constants. The mg term is gravity; the kx term is Hooke's law for the force exerted by a simple spring. The signs of mg and kx are opposite because gravity pulls in the direction of positive x and the spring pulls in the direction of negative x. Differential equations capture a system's physics in a general way: not only does their form mirror the physical laws, but their solutions also account for every possible behavior of the system. For any initial conditions for the position and velocity of the mass, for instance, the equation above completely describes where it will be at all times in the future. However, differential equations are much more difficult to work with than the algebraic models described in the previous paragraphs. They are also much more difficult to construct. Using observations of a black-box system's outputs to reverse-engineer its governing equations—i.e., figuring out a differential equation from partial knowledge about its *solutions*—is an extremely difficult task if one does not know what is inside the box. This procedure, which is known as *system identification* in the control-theory literature, is fairly straightforward if the system involved is linear; the textbook approach[28] is to choose a generic ordinary differential equation (ODE) system $\dot{\vec{x}}(t) = B\vec{x}(t)$—with $\vec{x}(t) = x_1(t), x_2(t), \ldots x_n(t)$—fast-Fourier-transform the sensor data, and use the characteristics of the resulting impulse response to determine the coefficients of the matrix B. The natural frequencies, which appear as spikes on the impulse response, yield the system's eigenvalues; the off-diagonal elements can be determined via an analysis of the shape of the impulse response curve between those spikes. See [28] or [33] for a full description of this procedure.

A linear, time-invariant system can be described quite nicely by the kinds of models that are described in this section, but nonstationarity or nonlinearity can throw a large wrench in the works. The standard textbook approach[10] to *nonstationary* data analysis involves special techniques that recognize the exact form of the nonstationarity (e.g., linear trend) and various machinations that transform the time series into stationary form, at which point one can use ARMA methods. *Nonlinearity* is not so easy to get around. It can be shown, for instance, that ARMA coefficients and the power spectrum (i.e., Fourier coefficients) contain the same information. Two very different nonlinear systems, however, may have almost indistinguishable spectra, so methods in the ARMA family break down in these cases[4]. Spectral similarity of dissimilar systems also has important implications for signal separation. In linear systems, it is often safe to assume, and easy to recognize, that the "important" parts of the signal are lower down on the frequency scale and easily separable from the noise (which is assumed to be high frequency), and it is easy to implement digital filters that remove components of a signal above a specified cutoff frequency[37]. In nonlinear systems, as described in more detail in the following section, the important parts of the signal often cover the entire spectrum, making signal separation a difficult proposition. Nonlinearity is even more of a hurdle in system identification: constructing dynamic models of linear systems is relatively tractable, but human practitioners consider nonlinear system identification to be a "black art," and automating the process[7] is quite difficult.

2 Nonlinear Dynamics Basics

A dynamical system is something whose behavior evolves with time: binary stars, transistor radios, predator-prey populations, differential equations, the air stream past the cowl of a jet engine, and myriad other examples of interest to scientists and engineers in general and intelligent data analysts in particular. The bulk of an engineering or science education and the vast majority of the data analysis methods in current use, some of which are outlined in the previous section, are focused on *linear* systems, like a mass on a spring: systems whose governing equations do not include products, powers, transcendental functions,

[3]The right-hand side of a linear differential equations is of the form $ax + b$

[4]One can construct a patchwork of local-linear ARMA models[47] in situations like this, but such tactics contribute little to *global* system analysis and understanding.

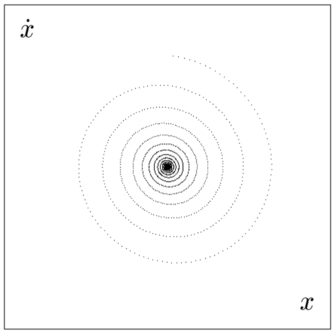


Figure 3: A state-space trajectory representing the oscillation of the spring-mass system of Figure 1.

etc. Very few systems fit this mold, however, and the behavior of nonlinear systems is far richer than that of linear systems. This richness and generality makes nonlinear systems both much more difficult and much more interesting to analyze.

The *state variables* of a dynamical system are the fundamental quantities needed to describe it fully—angular position θ and velocity $\omega = \dot{\theta}$ for a pendulum, for instance, or capacitor voltages and inductor currents in an electronic circuit. The number n of state variables is known as the *dimension* of the system; a pendulum or a mass on a spring is a two-dimensional system, while a three-capacitor circuit has three dimensions. Simple systems like this that have a finite number of state variables can be described by *ordinary differential equation* (ODE) models like Equation (4) for the spring-mass system or

$$\ddot{\theta}(t) = -g \sin \theta(t) \tag{5}$$

for a pendulum moving under the influence of gravity g. Equation (4) is linear and equation (5), because of the sin term, is not; in both systems, $n = 2$. If the number of state variables in the system is infinite—e.g., a moving fluid, whose physics is influenced by the pressure, temperature and velocity at *every point*—the system is called *spatiotemporally extended*, and one must use *partial differential equation* (PDE) models[14] to describe it properly. In this chapter, we will confine our attention to finite-dimensional dynamical systems that admit ODE models. Because so many real-world problems are nonlinear, we will concentrate on methods that are general and powerful enough to handle *all* dynamical systems—not just linear ones. Finally, since most natural and man-made systems are not only nonlinear but also *dissipative*—that is, they lose some energy to processes like friction—we will not cover the methods of conservative or Hamiltonian dynamics[3, 35].

Much of traditional systems analysis, as described in the previous section, focuses on time-series or frequency-domain data. The nonlinear dynamics community, in contrast, relies primarily upon the *state-space* representation, plotting the behavior on the n-dimensional space (R^n) whose axes are the state variables. In this representation, the damped oscillation of a mass bouncing on a spring manifests not as a pair of decaying sinusoidal time-domain signals, as in Fig. 1, but rather as a spiral, as shown in Fig. 3. *State-space trajectories* like this—system behavior (i.e., ODE solutions) for particular initial conditions—only implicitly contain time information; as a result, they make the geometry of the equilibrium behavior easy to recognize and analyze.

Dissipative dynamical systems have *attractors*: invariant state-space structures that remain after transients have died out. A useful way to think about this is to envision the "flow" of the dynamics causing the state to evolve towards a "low point" in the state-space landscape (cf., a raindrop running downhill into an ocean). There are four different kinds of attractors:

- *fixed* or *equilibrium points*
- *periodic orbits* (a.k.a. *limit cycles*)
- *quasiperiodic* attractors
- *chaotic* or *"strange"* attractors

A variety of pictures of these different attractors appear in the later pages of this chapter. Fixed points—states from which the system does not move—can be stable or unstable. In the former case (cf., Fig. 3) perturbations will die out; in the latter, they will grow. A commonplace example of a stable fixed point is a marble at rest in the bottom of a bowl; the same marble balanced precariously on the rim of that bowl is at an unstable fixed point. Limit cycles are signals that are periodic in the time domain and closed curves in state space; an everyday example is the behavior of a healthy human heart. (One of the heart's pathological behaviors, termed *ventricular fibrillation*, is actually chaotic.) Quasiperiodic orbits and chaotic attractors are less familiar and harder to analyze, but no less common or interesting. The latter, in particular, are fascinating. They have a fixed, complicated, and highly characteristic geometry, much like an eddy in a stream, and yet nearby trajectories on a chaotic attractor move apart exponentially fast with time, much as two nearby wood chips will take very different paths through the same eddy. Trajectories cover chaotic attractors *densely*, visiting every point to within arbitrary ϵ, and yet they never quite repeat exactly. These properties translate to the very complex, almost-random, and yet highly structured behavior that has intrigued scientists and engineers for the last twenty years or so. Further discussion of chaotic systems, including a variety of examples, appears in section 4. Parameter changes can cause a nonlinear system's attractor to change drastically. A change in blood chemistry, for instance, can cause the heart's behavior to change from a normal sinus rythym to ventricular fibrillation; a change in temperature from 99.9 to 100.1 degrees Celsius radically alters the dynamical properties of a pot of water. These kinds of *topological* changes in its attractor are termed *bifurcations*.

Attractor type is an important nonlinear data analysis feature, and there are a variety of ways for computer algorithms to recognize it automatically from state-space data. One standard geometric classification approach is *cell dynamics*[26], wherein one divides the state space into uniform boxes. In Fig. 4, for example, the limit cycle trajectory—a sequence of two-vectors of floating-point numbers measured by a finite-precision sensor—can be represented as the *cell sequence*

$$[...(1,0)(2,0)(3,0)(4,0)(4,1)(5,1)(5,2)(4,2)(3,2)(3,3)(4,3)(4,4)...]$$

Because multiple trajectory points are mapped into each cell, this discretized representation of the dy-

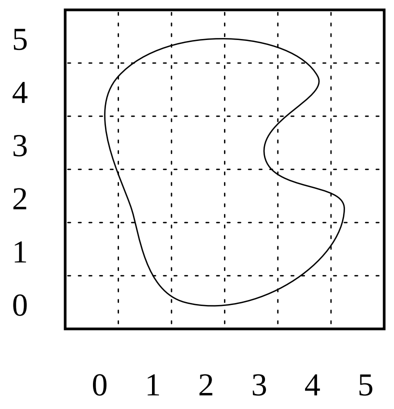


Figure 4: Identifying a limit cycle using simple cell mapping

namics is significantly more compact than the original series of floating-point numbers and therefore much easier to work with. This is particularly important when complex systems are involved, as the number of cells in the grid grows exponentially with the number of dimensions[5]. Though the approximate nature of this representation does abstract away much detailed information about the dynamics, it preserves many of its important invariant properties; see [23] or [32] for more details. This point is critical to the utility of the method; it means that conclusions drawn from the discretized trajectory are also true of the real trajectory—for example, a repeating sequence of cells in the former, as in Fig. 4, implies that the full R^n dynamics is on a limit cycle.

[5]The example of Fig. 4 is two-dimensional, but the cell dynamics formalism generalizes easily to arbitrary dimension.

Much as a bowl can have several low spots or a mountain range can include many drainages, nonlinear systems can have multiple attractors of different types. Each attractor lies in a unique *basin of attraction* (all the points in the bowl or mountain range from which a marble or raindrop will end up at that attractor), and those basins partition[6] the state space. A linear system, on the other hand, can have only one fixed point, and its basin—if it is stable—is all of R^n. Dissipation, the notion of transient behavior that dies out, and the requirement that attractors are *proper* subsets of their basins are linked. Dynamicists think about basin/attractor dynamics using the *state-space contraction* metaphor: initial conditions anywhere inside the boundary of a basin of attraction will converge to the associated attractor, so one envisions a volume of initial conditions spread out across the basin, all eventually converging to the attractor. (*Conservative* systems—those in which energy is conserved—preserve state-space volumes and do not have attractors.) Basins are very important for nonlinear data analysis. Attractors in neighboring basins can be quite different, and so small differences in initial conditions matter; a raindrop a millimeter away from a sharp mountain ridge will take a radically different path if a light breeze comes up. This can be a useful way to approach the analysis of a system that appears to have several behavior modes. Basin boundaries can be computed using the grid-based techniques described in the previous paragraph, as well as a variety of other approaches; see [21] or section 10.3.3 of [39] for more details.

The fixed nature of an attractor of a dynamical system is critically important to the approach to intelligent data analysis that is outlined in this chapter; it implies that the *dynamical invariants* of such attractors—their immutable mathematical properties—do not depend on how these attractors are viewed[7], and therefore that analysis techniques that measure those invariants should yield the same results in the face of transformations like coordinate changes, for instance. Stability is such an invariant: a stable fixed point should not become unstable if one recalibrates a sensor. Topological dimension is another: a fixed point should not appear as a limit cycle when viewed from another angle. The nonlinear dynamics literature defines dozens of other dynamical invariants and proposes hundreds of algorithms for computing them; see [2] for a readable and comprehensive introduction. The two most common invariants in this list are the *Lyapunov exponent* λ, which measures how fast neighboring trajectories diverge, and the family of *fractal dimensions*, so named because they can take on non-integer (fractional $\rightarrow$ "fractal") values, which measure how much of R^n a trajectory actually occupies.

The Lyapunov exponent is defined as:

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

where the $s_i(t)$ are the eigenvalues of the variational system (the matrix-valued linear differential equation that governs the growth of a small variation in the initial condition; see appendix B of [39] for details). A n-dimensional system has n λs, each measuring the expansion rate, in one "direction," of the distance between two neighboring trajectories. λ is the nonlinear generalization of the real part of an eigenvalue; a positive λ implies exponential growth of a perturbation along the *unstable manifold*, the nonlinear generalization of the eigenvector associated with a positive-real-part eigenvalue. A negative λ implies exponential shrinkage of the perturbation along the *stable manifold* that is the nonlinear analog of the stable eigenvector. A system that has all negative λs in some region is said to be "stable in the sense of Lyapunov," and its trajectories relax to some proper subset of that region (the attractor). A system with all *positive* λs is unstable in all directions. A zero λ implies less-than-exponential growth, which generally takes place along the attractor. State-space contraction, part of the formal definition of dissipation, requires that $\Sigma\lambda_i < 0$ for any dissipative system.

The point of retooling the definition of dimension to allow for non-integer values is to be able to accurately characterize objects that are "between" two topological dimensions. A Cantor set, for example—constructed by removing the middle portion of a line segment *ad infinitum*, as shown in Fig. 5—contains an infinite number of zero-dimensional objects (points) but its topological dimension is still zero. Fractal dimensions capture this property; one standard measure of the fractal dimension of the middle-third removed Cantor set, for example, is 0.63. This invariant is common in the nonlinear dynamics community

[6]This is a slight abuse of the technical term "partition;" nonattracting sets—which have no basins of attraction—can exist in dynamical systems, and basins technically do not include their boundaries.

[7]within some limits, of course

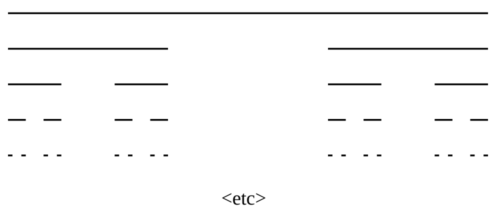


Figure 5: A middle-third-removed Cantor set

because many (not all) chaotic attractors have fractal state-space structure—that is, their attractors have non-integer values of the fractal dimension. The most-common algorithm for computing any fractal dimension of a set A, loosely described, is to discretize state space into ϵ-boxes, count the number of boxes[8] occupied by A, and let $\epsilon \to 0$:

$$d_c = \lim_{\epsilon \to 0} \left\{ \frac{\log(N(A,\epsilon))}{\log(1/\epsilon)} \right\} \tag{7}$$

where $N(A,\epsilon)$ is the number of closed balls of radius $\epsilon > 0$ needed to cover A. (Strictly speaking, one doesn't just *count* the boxes, but rather accumulates the value of some measure on each box; see the discussion of equation (8) in section 3.2.) In reality, floating-point arithmetic and computational complexity place obvious limits on the $\epsilon \to 0$ part of equation (7); in practice, one repeats the dimension calculation for a range of ϵs and finds the power-law asymptote in the middle of the log-log plot of dimension versus ϵ.

Dynamical invariants like λ and d_c can be used to classify attractors. In a n-dimensional system, there are n Lyapunov exponents λ_i and:

- A stable fixed point has n negative λs (since perturbations in any direction will die out) and a fractal dimension of zero.
- An attracting limit cycle has one zero λ and $n-1$ negative λs (since perturbations off the attractor will die out, and a perturbation along the orbit will remain constant) and a fractal dimension of one.
- A chaotic attractor has one zero λ (along the attractor), at least one positive λ and—generally but not always—a non-integer fractal dimension. The positive λ reflects chaos's hallmark "sensitive dependence on initial conditions:" the system's tendency to force neighboring trajectories apart.

Intelligent data analysis tools that target attractor type, basin geometry, dynamical invariants, etc. are harder to implement than the kinds of techniques that one can apply to a linear system, and their implications are generally less wide-ranging. If the system under consideration is linear, as mentioned previously, data analysis is relatively easy and one can make more (and more-powerful) inferences from the results. Where nonlinear systems are concerned, however, traditional methods often do not apply; in these problems, time-series analysis is much harder and the conclusions one can draw from the results are fundamentally limited in range. This stems from the inherent mathematical difficulties of the domain, and it is essentially unavoidable. If one is faced with a fundamentally nonlinear problem, one has no choice but to use the more difficult (and perhaps unfamiliar) methods covered in this chapter. The reader who is interested in delving deeper into this field should consult any of the dozens of good nonlinear dynamics books that are currently in print. An excellent overall starting point is [45], the basic mathematics is covered particularly well in [25], a comprehensive collection of algorithms appears in [39], and an entertaining popular overview may be found in [44].

3 Delay-Coordinate Embedding

Given a time series from a sensor on a single state variable $x_i(t)$ in a n-dimensional dynamical system, delay-coordinate embedding lets one reconstruct a useful version of the internal dynamics[9] of that sys-

[8]Hence the term "box-counting dimension."

[9]That is, the state-space trajectory $\{\vec{x}(t)\}$, where $\vec{x} = \{x_1, x_2, \ldots x_n\}$ is the vector of state variables

$x_i(t)$	t	$x_i(t)$	t
1.6352	0.000	1.6214	0.008
1.6337	0.001	1.6183	0.009
1.6322	0.002	1.6183	0.010
1.6306	0.003	1.6168	0.011
1.6276	0.004	1.6137	0.012
1.6260	0.005	1.6107	0.013
1.6230	0.006	1.6076	0.014
1.6214	0.007	1.6045	0.015

Table 1: An example data set: samples of one state variable x_i, measured every $\Delta t = 0.001$ seconds.

tem. If the embedding is performed correctly, the theorems involved guarantee that the reconstructed dynamics is topologically (i.e., qualitatively) identical to the true dynamics of the system, and therefore that the dynamical invariants are also identical. This is an extremely powerful correspondence; it implies that conclusions drawn from the *embedded* or *reconstruction-space* dynamics are also true of the real—unmeasured—dynamics. This implies, for example, that one can reconstruct the dynamics of the earth's weather simply by setting a thermometer on a windowsill.

There are, of course, some important caveats. Among other things, a correct embedding requires at least twice as many dimensions as the internal dynamics—a requirement that makes reconstruction of the weather thoroughly impractical, as it is a spatially extended system and thus of infinite dimension. Moreover, even if the dynamics of the system under examination is simple, its precise dimension is often very hard to measure and rarely known *a priori*. This is the main source of the hard problems of delay-coordinate embedding, which are discussed in more detail—together with some solutions—in the following sections.

3.1 Embedding: the basic ideas

Consider a data set comprised of samples $x_i(t)$ of a single state variable x_i in a n-dimensional system, measured once every Δt seconds, such as the example sensor time series shown in Table 1. To embed such a data set, one constructs d_E-dimensional *reconstruction-space* vectors $\vec{r}(t)$ from d_E time-delayed samples of the $x_i(t)$, such that

$$\vec{r}(t) = [x_i(t),\ x_i(t-\tau),\ x_i(t-2\tau),\ \ldots\ ,\ x_i(t-(m-1)\tau)]$$

or

$$\vec{r}(t) = [x_i(t),\ x_i(t+\tau),\ x_i(t+2\tau),\ \ldots\ ,\ x_i(t+(m-1)\tau)]$$

For example, if the time series in Table 1 is embedded in two dimensions ($d_E = 2$) with a delay $\tau = 0.005$, the first few points in the reconstruction-space trajectory are:

```
(1.6352 1.6260)
(1.6337 1.6230)
(1.6322 1.6214)
(1.6306 1.6214)
(1.6276 1.6183)
(1.6260 1.6183)
...
```

If $d_E = 5$ and $\tau = 0.003$, the first few points of the trajectory are:

```
(1.6352 1.6306 1.6230 1.6183 1.6137)
```

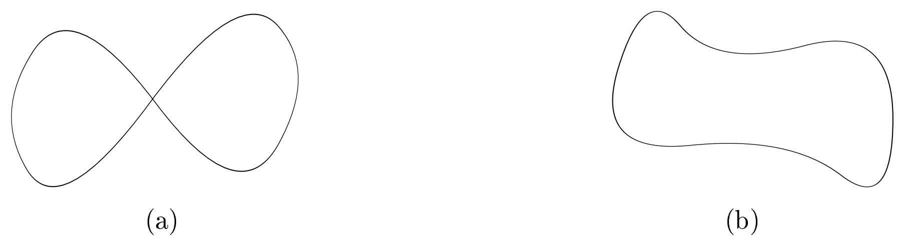


Figure 6: A closed curve in 3D, viewed from (a) the top and (b) the side. The latter projection is is topologically conjugate to a circle; because of the self-intersection, the projection in (a) is not.

```
(1.6337 1.6276 1.6214 1.6183 1.6107)
(1.6322 1.6260 1.6214 1.6168 1.6076)
(1.6306 1.6230 1.6183 1.6137 1.6045)
...
```

The act of sampling a single system state variable $x_i(t)$ is equivalent to projecting an n-dimensional state-space dynamics down onto a single axis; the embedding process demonstrated above is akin to "unfolding" or "reinflating" such a projection, albeit on different axes: the d_E delay coordinates $x_i(t)$, $x_i(t-\tau)$, $x_i(t-2\tau)$, etc. instead of the n true state variables $x_1(t)$, $x_2(t)$, $\ldots$, $x_n(t)$. The central theorem[46] relating such embeddings to the true internal dynamics, which is generally attributed to Takens, was proved in [38] and made practical in [42]; informally, it states that given enough dimensions (d_E) and the right delay (τ), the reconstruction-space dynamics and the true, unobserved state-space dynamics are topologically identical. More formally, the reconstruction-space and state-space trajectories are guaranteed to be diffeomorphic if $d_E = 2n+1$, where n is the true dimension of the system[10].

Diffeomorphisms—transformations that are invertible, differentiable, and that possess differentiable inverses—preserve topology but not necessarily geometry. This means that an attractor reconstructed using delay-coordinate embedding may look very different from the true attractor, but the former can be stretched and bent into the shape of the latter without "crossing over" itself. The $2n+1$ requirement of the theorem is really a brute-force worst-case limit for eliminating projection-induced crossings. The self-intersection point in Fig. 6(a), for example, makes the 2D projection of that curve not diffeomorphic to a circle; viewed from another angle, however, as in part (b), the curve is indeed smoothly deformable into a circle. $2n+1$ is simply the minimum number of dimensions required to eliminate all such crossings, so lower-dimension embeddings may well be correct. This can, in fact, be exploited in deriving a tighter and easy-to-compute bound on d_E that is valid in "almost every" situation[42].

The topological equivalence guaranteed by the Takens theorem is a powerful concept: it lets one draw sensible, justifiable conclusions about the full dynamics of an n-dimensional system using only the output of a single sensor. In particular, many properties of the dynamics are preserved by diffeomorphisms; if one computes them from a correct embedding, the answer will hold for the true internal dynamics as well. There are, of course, some important conditions on the theorem, and the difficulties that they pose are the source of most of the effort and subtlety in these types of methods. Specifically, in order to embed a data set, one needs d_E and τ, and neither of these parameters can be measured or derived from the data set, either directly or indirectly, so algorithms like those described in the following section rely on numeric and geometric heuristics to estimate them.

From a qualitative standpoint, embedding is not as outlandish as it may initially appear. The state variables in a nonlinear system are generally coupled to one another temporally by the dynamics, so using quantities that resemble forward differences as the axes of a reconstruction space makes some sense. (As mentioned before, techniques like divided differences can, in theory, be used to derive velocities from position data; in practice, however, these methods often fail because the associated arithmetic magnifies

[10] τ is missing from these requirements because the theoretical conditions upon it are far less stringent and limiting, as described in the second paragraph of the next section.

sensor error.) One can think of the $x_i(t)$, $x_i(t-\tau)$, etc., as independent coordinates that are nonlinearly related to the true state variables. The specifics of that relationship may not—and *need* not—be obvious; the important point is that the form of that relationship ensures that the reconstructed dynamics $\vec{r}(t) \in R^{d_E}$ is diffeomorphic to the true dynamics $\vec{x}(t) \in R^n$.

3.2 Finding appropriate embedding parameters

The time-series analysis literature contains scores of methods that use a variety of heuristics to solve the central problem of delay-coordinate reconstruction: given a scalar time series from a dynamical system of unknown dimension, estimate values for the dimension d_E and delay τ that will guarantee a correct embedding. Many of these algorithms are somewhat *ad hoc*; almost all are computationally expensive and highly sensitive to sensor and algorithm parameters, and different ones produce surprisingly different results, even on the same data set. See [2] for a recent summary and the FAQ for the newsgroup `sci.nonlinear`[1] for a list of public-domain software implementations of many of these algorithms. This chapter covers only a few of the most widely accepted and/or interesting representatives of this body of work.

The delay τ governs whether or not the coordinates $x(t-j\tau)$ are indeed independent. If τ is small, the reconstruction-space trajectory will lie very near the main diagonal. As long as the structure is not infinitely thin, this type of embedding is *theoretically* correct; in practice, however, finite-precision arithmetic on fixed-length (and possibly noisy) trajectories can easily generate apparent crossings in situations like this. If τ is too large, on the other hand, successive points $\vec{r}(t)$ and $\vec{r}(t+\Delta t)$, where Δt is the sampling interval, will be uncorrelated and the larger spacing of the points in $\vec{r}(t)$ again interferes numerically with topological equivalence. Ideally, then, one wants a time window for τ that is long enough for the system state to evolve to a visible (with respect to floating-point arithmetic) but not excessive extent.

One way to compute such an estimate is to perform some sort of averaged autocorrelation of successive points in the time series $x_i(t)$ or in the embedded trajectory $\vec{r}(t)$—e.g., average mutual information[17]—as a function of τ. For very small τ, these statistics will be close to 1.0, since successive reconstruction-space trajectory points are very close to one another[11]. For larger τ, successive points become increasingly uncorrelated. The first minimum in the distribution is a sensible choice for τ: qualitatively, it corresponds to the smallest τ for which the dynamics has caused nearby trajectory points to become *somewhat* uncorrelated (i.e., new information has been introduced between samples). This choice was originally proposed[17] by Fraser; other authors suggest using other features of the autocorrelation curve to choose good values for τ—e.g., the first *maximum*, with the rationale that these "close returns" correspond to natural periods of the system. Note that since one can compute average mutual information (AMI) from one- and two-embeddings (that is, $d_E = 1$ and $d_E = 2$), this kind of procedure does *not* require one to first find a correct value for d_E.

The Pineda-Sommerer (P-S) algorithm[40], which solves both halves of the embedding parameter problem at once, is more esoteric and complicated. Its input is a time series; its outputs are a delay τ and a variety of different estimates of the dimension d_E. The procedure has three major steps: it estimates τ using the mutual information function, uses that estimated value τ_0 to compute a temporary estimate E of the embedding dimension, and uses E and τ_0 to compute the *generalized dimensions* D_q, members of a parametrized family of fractal dimensions. Generalized dimensions are defined as

$$D_q = \frac{1}{q-1} \limsup_{\epsilon \to 0} \frac{\log \sum_i p_i^q}{\log \epsilon} \tag{8}$$

where p_i is some measure of the trajectory on box i. D_0, D_1, and D_2 are known, respectively, as the capacity, information, and correlation dimensions. The actual details of the P-S algorithm are quite involved; we will only give a qualitative description:

- Construct one- and two-embeddings of the data for a range of τs and compute the saturation dimension D_1^* of each; the first minimum in this function is τ_0. The D_1^* computation entails:

[11]Note that $\vec{r}(t) = x_i(t)$ if $d_E = 1$.

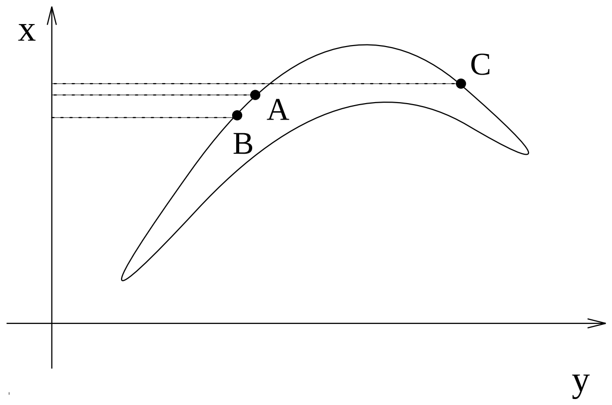


Figure 7: The geometric basis of the FNN algorithm. If this curve is projected onto the x axis, the points A, B, and C appear to be near neighbors, even though C is quite distant in the 2D view. Differences between one- and two-embeddings of these data will expose *false near neighbors* like the [A,C] pair.

- Computing the information dimension D_1 for a range of embedding dimensions E and identifying the saturation point of this curve, which occurs at embedding dimension D_1^*. The D_1 computation entails:
  - Embedding the data in E-dimensional space, dividing that space into E-cubes that are ϵ on a side, and computing D_1 using equation (8) with $q = 1$.

P-S incorporates an ingenious complexity-reduction technique in the fractal dimension calculation: the ϵs (see equation (7)) are chosen to be of the form 2^{-k} for integers k and the data are integerized, allowing most of the mathematical operations to proceed at the bit level and vastly accelerating the algorithm.

The false near neighbor (FNN) algorithm[29], which takes a τ and a time series and produces a lower bound on d_E, is far simpler than P-S. (As mentioned above, *upper* bounds for d_E are often chosen to be the smallest integer greater than twice the capacity dimension, D_0, of the data, in accordance with [42].) FNN is based on the observation that neighboring points may in reality be projections of points that are very far apart, as shown in Fig. 7. The algorithm starts with $d_E = 1$, finds each point's nearest neighbor, and then embeds the data with $d_E = 2$. If the point separations change abruptly between the one- and two-embeddings, then the points were *false* neighbors (like A and C in the x-projection of Fig. 7). The FNN algorithm continues adding dimensions and re-embedding until an acceptably small[12] number of false near neighbors remains, and returns the last d_E-value as the estimated dimension. This algorithm is computationally quite complex; finding the nearest neighbors of m points requires $O(m^2)$ distance calculations and comparisons. This can be reduced to $O(m \log m)$ using a K-D tree implementation[18].

As should be obvious from the content and tone of this introduction, estimating τ and d_E is algorithmically *ad hoc*, computationally complex, and numerically sensitive. For this reason, among others, nonlinear time-series analysis techniques that do not require embedding are extremely attractive. Recent evidence[27] suggests that the *recurrence plot*—a two-dimensional representation of a single trajectory wherein the time series spans both ordinate and abscissa and each point (i, j) on the plane is shaded according to the distance between the two corresponding trajectory points y_i and y_j—may be such a technique. Among their other advantages, recurrence plots also work well on nonstationary data; see the following section for an example (Fig. 11) and more discussion.

[12]An algorithm that removes *all* false near neighbors can be unduly sensitive to noise.

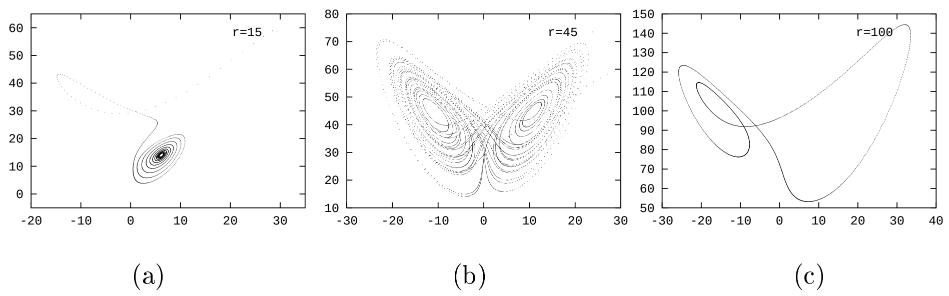


Figure 8: State-space plots of Lorenz system behavior with $a = 10$ and $b = 8/3$: (a) a stable fixed point for $r = 15$ (b) a chaotic attractor for $r = 45$ (c) a periodic orbit for $r = 100$. All three plots are two-dimensional $(x - z)$ projections of three-dimensional attractors.

4 Examples

In this section, we demonstrate some of the concepts and algorithms described in the previous two sections using two examples, one simulated and one real.

4.1 The Lorenz system

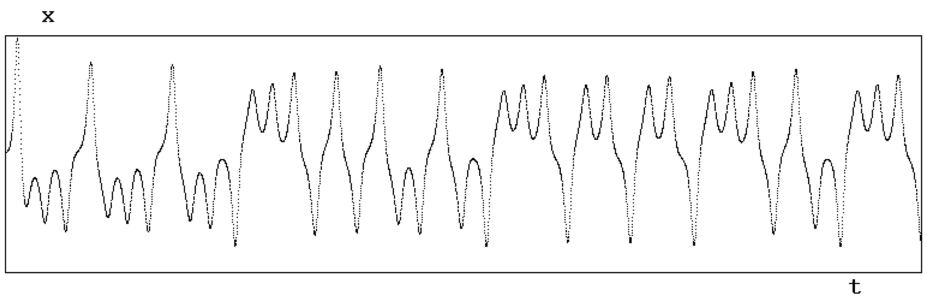
In the early 1960s[34], Edward Lorenz derived a simple model of the physics of a fluid that is being heated from below:

$$\dot{\vec{x}}(t) = \frac{d}{dt}\vec{x}(t) = \left[\begin{array}{c} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{array}\right] = \left[\begin{array}{c} a(y(t) - x(t)) \\ rx(t) - y(t) - x(t)z(t) \\ x(t)y(t) - bz(t) \end{array}\right] \tag{9}$$

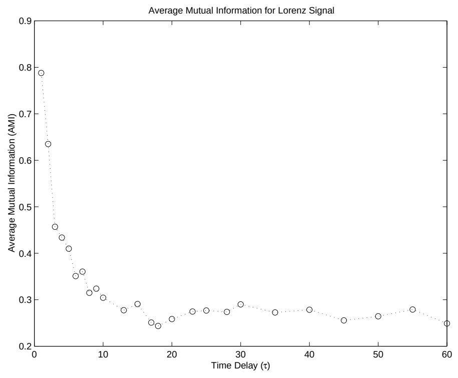
This 3^{rd}-order ($n = 3$) ODE system is a rough approximation of a much more complex model: the Navier-Stokes PDEs for fluid flow. The state variables x, y, and z are convective intensity, temperature variation, and the amount of deviation from linearity in the vertical convection profile, respectively; the coefficients a and r are physical parameters of the fluid—the Prandtl and Rayleigh numbers—and b is the aspect ratio. This set of equations is one of the most common examples in the nonlinear dynamics literature. At low r values, its solutions exhibit damped oscillations to simple fixed-point equilibria, the first category on the list of attractor types on page 7, as shown in Fig. 8(a). For higher r—which translates to a higher heat input—the convection rolls in the modeled fluid persist, in a complicated, highly structured, and nonperiodic way; see part (b) of Fig. 8 for an example. This behavior, reported in a 1963 paper entitled "Deterministic Nonperiodic Flow," led Lorenz to recognize the classic "sensitive dependence on initial conditions" in the context of a fixed attractor geometry that is now a well-known hallmark of chaos. (The term "chaos" was coined twelve years later[31].) If r is raised further, the convection rolls become periodic—the second category in the list on page 7. See part (c) of the Figure for an example.

The trajectories plotted in Fig. 8 include complete information about all three of the state variables. In the analysis of a real system, this may be an overly optimistic scenario; while temperature is not hard to measure, the other state variables are not so easy, so a full state-space picture of the dynamics—information that is amenable to the techniques of section 2—may well be unavailable. Using the theory and techniques described in section 3, however, one can reconstruct the internal dynamics of this system from a time-series sampling of *one* of its state variables—say, the x coordinate of the chaotic attractor in part (b) of Fig. 8, which is plotted in time-domain form in Fig. 9(a). After embedding those data in delay coordinates, one can apply the nonlinear state-space analysis methods of section 2 to the results. The first step in the embedding process is to decide upon a delay, τ. The first minimum in the AMI results shown in Fig. 9 falls at roughly $\tau = 0.09$ seconds[13]. Using this τ, the false-near neighbor results (part (b) of Fig. 9) suggest an

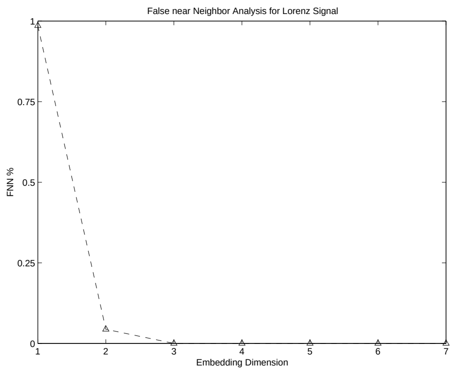
[13]The x-axis of the plot is measured in multiples of the sample interval of 0.002 second.



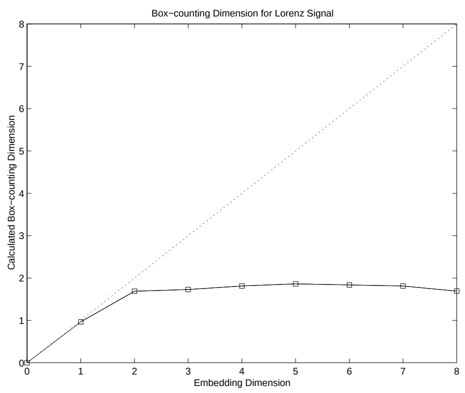
(a)



(b)



(c)



(d)

Figure 9: The x coordinate of the chaotic Lorenz signal from part (b) of Fig. 8 and the corresponding embedding parameter analysis: (a) time series (b) average mutual information (AMI) as a function of the delay τ (c) false-near neighbor (FNN) percentage as a function of embedding dimension d_E (d) box-counting dimension (D_0) as a function of d_E. AMI, FNN and D_0 results courtesy of Joe Iwanski.

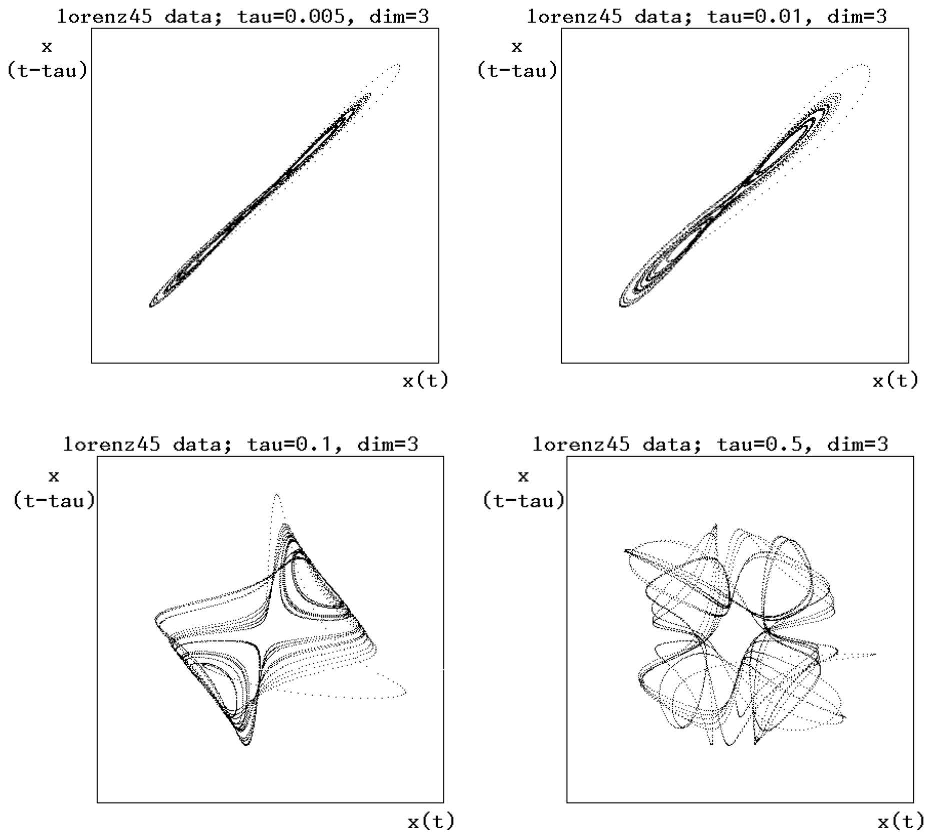


Figure 10: Embeddings of the chaotic Lorenz signal from Fig. 9(a) with $d_E = 3$ and various delays, plotted in 2D projection. The formal requirements of the embedding process—which these attractors meet—guarantees that they are topologically identical to the true attractor in Fig. 8(b).

embedding dimension of two or three, depending on one's interpretation of the heuristic "acceptably small percentage" threshold in the algorithm. The box-counting dimension of this data set levels off at roughly 1.8 for $d_E = 2$ and above, as can be seen in part (c) of the Figure. Following [42], this would imply an upper-bound embedding dimension of four.

It can be difficult to keep this menagerie of dimensions straight. In this example, the true dimension is known: $n = 3$. The time series $x(t)$ in Fig. 9(a) is a one-dimensional projection of the R^3 trajectory in Fig. 8(b) onto the x axis. In the worst case, the Takens theorem tells us that an accurate reconstruction may require as many as $d_E = 2n + 1 = 7$ embedding dimensions in order to assure topological conjugacy to the true dynamics. Recall that this is a very pessimistic upper bound; in practice, slightly more opportunistic algorithms like the one proposed in [42] are able to make better bounds estimates—values for d_E that are lower than $2n+1$ *and*, at the same time, that avoid projection-induced topological inequivalencies between the true and reconstructed dynamics. In making such estimates, many of these algorithms make use of the fact that attractors do not occupy *all* of R^n. The fractal dimension of the $a = 10$, $r = 45$, $b = 8/3$ Lorenz attractor, for instance, is somewhere between 1 and 2, depending upon which algorithm one uses; the calculated capacity dimension D_0 of the trajectory in Fig. 8(b), in particular, is 1.8, implying an upper bound of $d_E = 4$. Even this estimate is somewhat pessimistic. Fractal dimension is a highly digested piece of information: a lumped parameter that compresses all the geometric information of an attractor into a single number. Because the FNN algorithm is based upon a more-detailed examination of the geometry, its results ($d_E = 3$, in this case) are a better *lower* bound.

Fig. 10 shows embeddings of the Lorenz time series of Fig. 9 with $d_E = 3$ and various τs. Note how this reconstructed attractor starts out as a thin band near the main diagonal and "inflates" with increasing τ.

b)

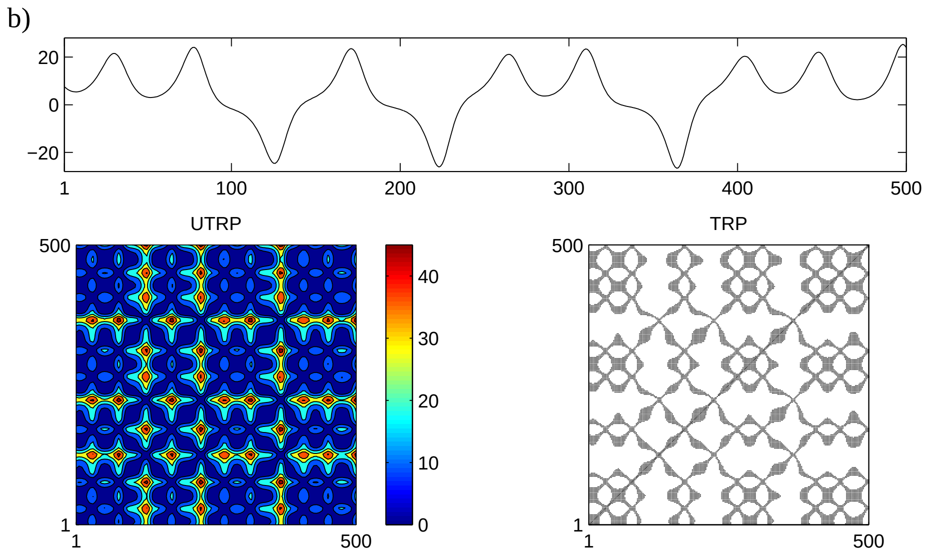


Figure 11: Recurrence plots (RPs) of a short segment (top) of the Lorenz data from part (a) of Fig. 9. The pixel at i, j is shaded to reflect the distance between the ith and jth point in the time series. On the unthresholded recurrence plot (UTRP) on the bottom left, each pixel is coded according to the color bar shown to the right of the UTRP; in the thresholded RP to the bottom right, pixels are black if the distance falls within some prescribed *threshold corridor* and white otherwise. Results courtesy of Joe Iwanski.

The sample interval in this data set was not much smaller than the τ returned by the AMI algorithm, so the thinnest reconstruction is fairly wide. Note, too, the resemblance of these reconstructed attractors to the true state-space trajectory in Fig. 8(b) and how that resemblance changes with τ. The whole point of doing an embedding is that the former can be deformed smoothly into the latter—even the $\tau = 0.5$ reconstruction, where the similarity (let alone the diffeomorphism!) is hard to visualize—and that the dynamical invariants of true (Fig. 8(b)) and reconstructed (Fig. 10) attractors are identical. That is, a fixed point in the reconstructed dynamics implies that there is a fixed point in the true dynamics, and so on. As noted before, this is the power of delay-coordinate embedding: one can use nonlinear dynamics analysis techniques on its results and safely extend those conclusions to the hidden internal dynamics of the system under examination.

It would, of course, be ideal if one could avoid all of these embedding machinations and analyze the scalar time series directly. As mentioned at the end of section 3, recurrence plots (RPs) are relatively new and potentially quite powerful nonlinear time-series analysis tools whose results appear to be independent of embedding dimension in some cases[27]. An RP is a two-dimensional representation of a single trajectory; the time series is spread out along both x and y axes of the plot, and each pixel is shaded according to the distance between the corresponding points—that is, if the 117th point on the trajectory is 14 distance units away from the 9435th point and the distance range 13–15 corresponds to the color red, the pixel lying at (117, 9435) on the RP will be shaded red. Fig. 11 shows a recurrence plot (RP) of a short segment of the the Lorenz signal in part (a) of Fig. 9. Different types of attractors leave clear and suggestive signatures in RPs; it is easy to recognize a periodic signal, for instance, and chaotic attractors exhibit the type of intricate patterns that are visible in Fig. 11. Formalized classification of these signatures, however, is a difficult problem—and a current research topic. There are well-developed statistical approaches[27, 48], but structural/metric analysis (e.g., via pattern recognition) is still an open problem, although some recent progress has been made[8, 19].

4.2 The driven pendulum

A time-series plot of a data set from an angle sensor on a parametrically forced pendulum—a solid aluminum arm that rotates freely on a standard bearing, driven vertically by a motor through a simple linkage—is shown in part (a) of Fig. 12. An actuator controls the drive frequency and a sensor (an optical encoder) measures its angular position. The behavior of this apparently simple device is really quite complicated and interesting: for low drive frequencies, it has a single stable fixed point, but as the drive frequency is raised, the attractor undergoes a series of bifurcations. In the sensor data, this manifests as interleaved chaotic and periodic regimes[13]. The driven pendulum is also interesting from a modeling standpoint; at high resolutions, the backlash in the bearings invalidates the standard textbook model. Modeling these effects is critical, for instance, to the accurate control of robot arms.

The test run plotted in Fig. 12 was chosen for this example because the pendulum is oscillating in a chaotic manner, which rules out many traditional time-series analysis methods. The chaos manifests as seemingly structured, almost-periodic patterns in the time-series signal: oscillations that are quite similar but not identical and that almost (but not quite) repeat. Though these patterns are highly suggestive, they are very difficult to describe or classify in the time domain; in a state-space view, however, the characteristic structure of the pendulum's chaotic attractor becomes patently obvious. Unfortunately, direct state-space analysis of this system is impossible. Only angle data are available; there is no angular velocity sensor and attempts to compute angular velocity via divided differences from the angle data yield numerically obscured results because the associated arithmetic magnifies the discretization error in angle (from the sensor resolution) and time (from timebase variation in the data channel).

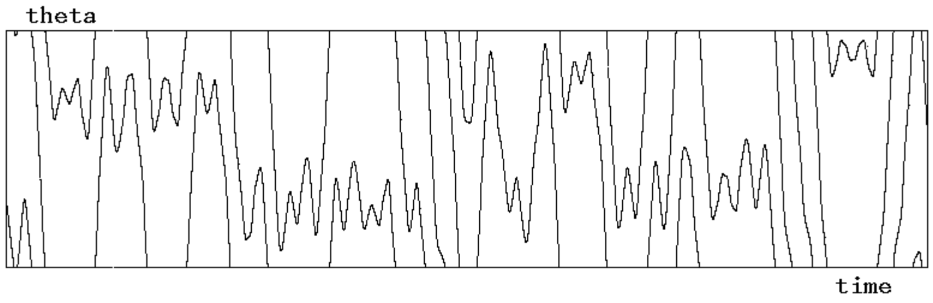
Delay-coordinate embedding, however, produces a clean, easily analyzable picture of the dynamics that is guaranteed to be diffeomorphic to the system's true dynamics. As in the Lorenz example, the embedding procedure begins with an estimation of τ. AMI results on the chaotic pendulum data set, shown in part (b) of Fig. 12, suggest a delay of 0.022 seconds (roughly 11 clicks at a sample interval of 0.002 seconds). FNN results constructed using this τ, shown in Fig. 11(c), suggest an embedding dimension of $d_E = 3$. The capacity dimension D_0—part (d)—varies between 1.7 and 2.1, implying an upper bound of $d_E = 5$, following [42].

In the Lorenz example of the previous section, the true dimension n was known. In the experimental pendulum setup, this is not the case. Presumably, three of the state variables are the bob angle θ, the angular velocity ω, and the time[14] t; if, however, the device is shaking the lab bench or contracting and expanding with ambient temperature, other forces may come into play and other state variables may have important roles in the dynamics. The results described in the previous paragraph, which suggest that the dynamical behavior of the pendulum is low-dimensional ($d_E = 3 - 5$, specifically), imply that the system is probably not influenced by variables like lab bench position or temperature. Higher d_E values from the estimation algorithms would suggest otherwise. This kind of high-level information, a natural result of delay-coordinate reconstruction and nonlinear dynamics analysis, is extremely useful for intelligent data analysis.

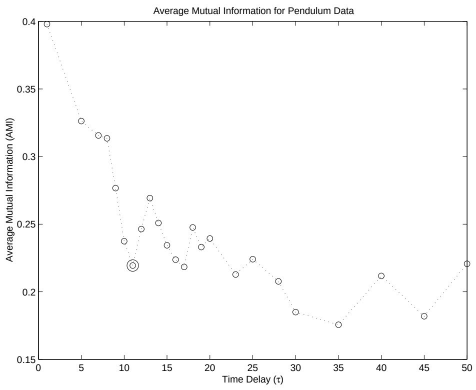
Fig. 13 shows embeddings for various τs; note how a small τ, as in the Lorenz example, creates a reconstruction that hugs the main diagonal, and how that reconstructed attractor unfolds as τ grows. The pendulum data were greatly oversampled, so it is possible to create a thinner embedding than in the Lorenz example, as shown in part (a) of this Figure. This is the type of reconstruction whose topologically conjugacy to the true dynamics is effectively destroyed by noise and numerical problems; note the apparent overlap of trajectories and sprinkling of noisy points just outside the true attractor in the $\tau = 0.01$ and $\tau = 0.02$ embeddings.

As before, once one has a successful reconstruction of the dynamics, all of the analysis tools described in section 2 can be brought to bear upon it, and their conclusions can be assumed to hold for the system's full underlying behavior.

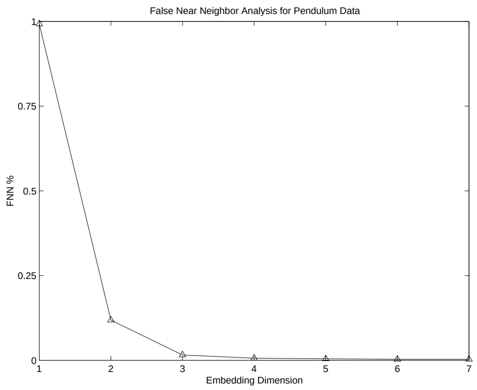
[14]In a driven or *nonautonomous* system, time is an exogenous variable.



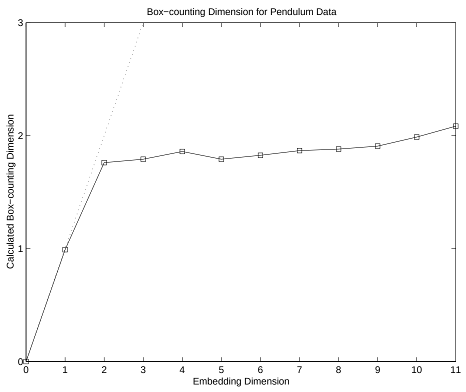
(a)



(b)



(c)



(d)

Figure 12: A chaotic sensor data set from a parametrically forced pendulum: (a) time-domain plot of the bob angle, measured modulo 2π (b) AMI (c) FNN and (d) D_0 results, all courtesy of Joe Iwanski.

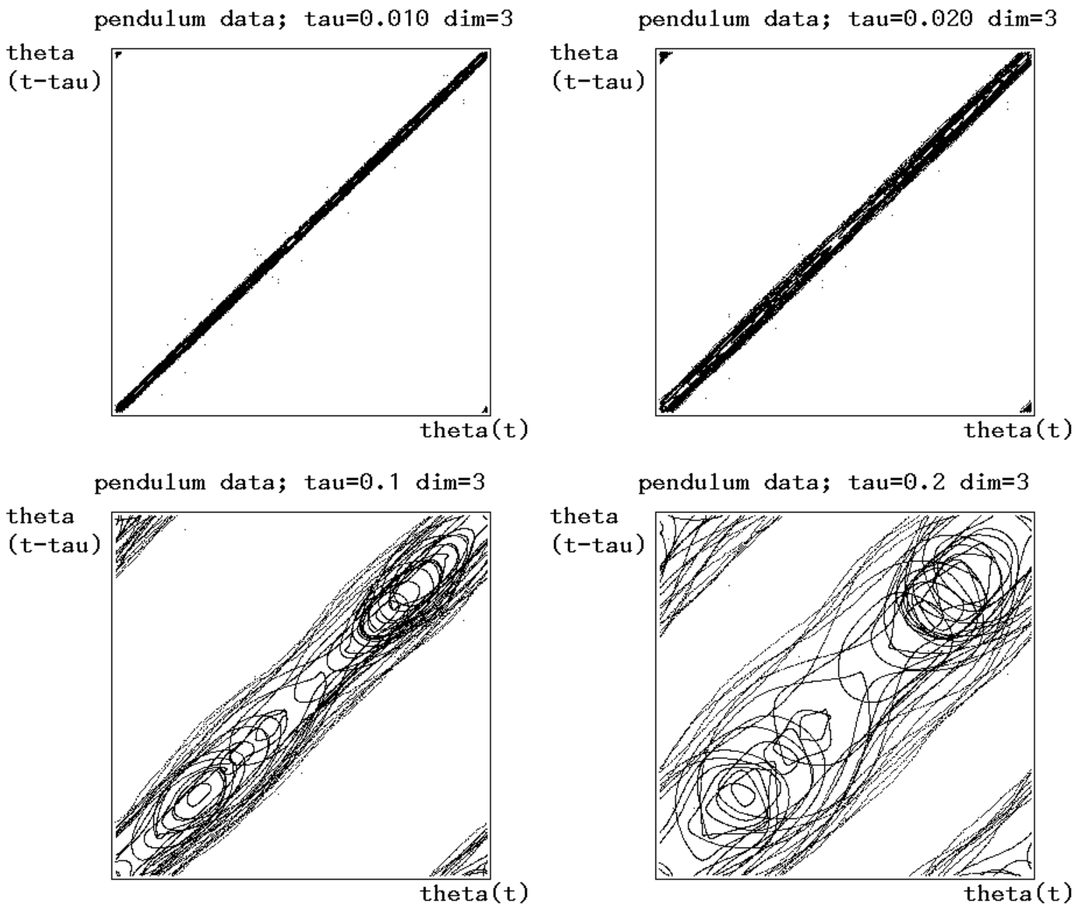


Figure 13: Embeddings of the pendulum data set from part (a) of Fig. 12.

5 Why Nonlinear Dynamics and Embedding are Useful for Intelligent Data Analysis

One of the more common—and more difficult—problems faced by an engineer or scientist is to analyze the dynamics of a complicated nonlinear system, given only measurements of one state variable. The techniques described in section 3 of this chapter, coupled with the theory covered in section 2, make significant inroads on this problem, allowing one to draw useful, justifiable, and sensible conclusions about a nonlinear system from the output of a single sensor. Specifically, a correct embedding of a data set from a single sensor on a black-box system is guaranteed to have the same dynamical invariants as the n-dimensional dynamics of the system inside the black box, and those invariants are useful tools for intelligent data analysis. Time-series analysis tools for *linear* systems are much easier to understand, implement, and use, but the universe is by and large nonlinear, so the application range of those kinds of tools is severely limited. Filtering out noise, for example, is fairly straightforward when one is working with data from a linear system: one simply transforms the data into the frequency domain and uses a low-pass filter. In nonlinear systems, separating signal from noise is problematic, as the former is often broad band and thus the two are intermingled. (Noise, incidentally, is infinite-dimensional, so its implications for embedding dimension calculations are dire; recall the $2n + 1$ requirement in the embedding theorems.) There has been some recent work on nonlinear "filtering" algorithms[22], including *filtered delay-coordinate embedding*[42] and an intriguing technique that exploits the stable and unstable manifold structure of a chaotic attractor to compress the noise ball. The latter method requires complete knowledge of the dynamics—the ODEs that govern the system. Since reverse-engineering ODEs from time-series samples of their solutions is an open problem for nonlinear systems, this filtering approach is hard to put into practice. One can, however, *approximate* the ODEs with local-linear models and get some reasonable results; see [15] for more details. In some cases, noise can actually be turned to advantage; its presence in a time series can allow the modeler to "explore" more of the state space[9].

One popular technique that may be conspicuous by its absence from this chapter is the neural net. Neural nets[24], which are discussed in Chapter 7 of this volume, are essentially nonlinear regression networks that model the input/output behavior of a system. They are very good at learning the patterns in a data set, and hence are very effective at predicting what a system will do next. However, they do *not* model the underlying physics in a human-comprehensible form. It is very difficult to learn anything useful about a system by examining a neural net that has been "trained" on that system, so this technique has been omitted from this discussion. Their ability to predict, however, makes neural nets potentially useful to intelligent data analysis in a somewhat counterintuitive fashion: if one needs more data, one can train a neural net on the time series and then use it to augment that data set, generating new points that are consistent with the dynamics[11].

Nonlinear dynamics techniques like the ones described in this chapter may be more difficult to understand and use than the more-familiar linear ones, but they are more broadly applicable—indeed, the latter can be viewed as a subset of the former. This family of theory and technique is valuable not only for time-series analysis, but also for many other tasks, such as modeling and prediction[12]. The kinds of models mentioned in the first paragraph of this section, for instance, have been successfully used to predict the behavior of systems ranging from roulette wheels[4] to physiological disease patterns, currencies markets, and Bach fugues[51].

Acknowledgements: Matt Easley, Joe Iwanski, and Vanessa Robins contributed code, data analysis results, and/or ideas to this paper.

References

[1] `http://amath-www.colorado.edu/appm/faculty/jdm/faq.html`.

[2] H. D. I. Abarbanel. *Analysis of Observed Chaotic Data.* Springer, 1995.

[3] V. I. Arnold. *Mathematical Methods of Classical Mechanics.* Springer, 1989. Second edition.

[4] T. Bass. *The Eudaemonic Pie.* Penguin, New York, 1992.

[5] G. Box and D. Cox. An analysis of transformations. *J. R. Statist. Soc. B*, 26:211–252, 1964.

[6] G. E. P. Box and F. M. Jenkins. *Time Series Analysis: Forecasting and Control.* Holden Day, 1976. Second edition.

[7] E. Bradley, M. Easley, and R. Stolle. Reasoning about nonlinear system identification. *Artificial Intelligence*, 133:139–188, December 2001.

[8] E. Bradley and R. Mantilla. Recurrence plots and unstable periodic orbits. *Physical Review Letters.* In review; also available as University of Colorado Department of Computer Science Technical Report CU-CS-919-01.

[9] J. L. Breeden, F. Dinkelacker, and A. Hübler. Noise in the modeling and control of dynamical systems. *Physical Review A*, 42(10):5827–5836, 1990.

[10] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods.* Springer Verlag, 1991. Second edition.

[11] R. Brown, P. Bryant, and H. D. I. Abarbanel. Computing the Lyapunov spectrum of a dynamical system from an observed time series. *Physical Review A*, 43:2787–2806, 1991.

[12] M. Casdagli and S. Eubank, editors. *Nonlinear Modeling and Forecasting.* Addison Wesley, 1992.

[13] D. D'Humieres, M. R. Beasley, B. Huberman, and A. Libchaber. Chaotic states and routes to chaos in the forced pendulum. *Physical Review A*, 26:3483–3496, 1982.

[14] B. Epstein. *Partial Differential Equations: An Introduction.* McGraw-Hill, 1962.

[15] J. Farmer and J. Sidorowich. Exploiting chaos to predict the future and reduce noise. In *Evolution, Learning and Cognition.* World Scientific, 1988.

[16] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.

[17] A. M. Fraser and H. L. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33(2):1134–1140, 1986.

[18] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

[19] J. Gao and H. Cai. On the structures and quantification of recurrence plots. *Physical Letters A*, 270:75–87, 2000.

[20] C. F. Gerald and P. O. Wheatley. *Applied Numerical Analysis.* Addison-Wesley, 1994.

[21] C. Grebogi, E. Ott, and J. A. Yorke. Chaos, strange attractors and fractal basin boundaries in nonlinear dynamics. *Science*, 238:632–638, 1987.

[22] J. Guckenheimer. Noise in chaotic systems. *Nature*, 298:358–361, 1982.

[23] B.-L. Hao. Symbolic dynamics and characterization of complexity. *Physica D*, 51:161–176, 1991.

[24] R. Hecht-Neilsen. *Neurocomputing.* Addison Wesley, 1990.

[25] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, San Diego CA, 1974.

[26] C. S. Hsu. *Cell-to-Cell Mapping.* Springer-Verlag, New York, 1987.

[27] J. Iwanski and E. Bradley. Recurrence plots of experimental data: To embed or not to embed? *Chaos*, 8(4), 1998.

[28] J.-N. Juang. *Applied System Identification.* Prentice Hall, Englewood Cliffs, N.J., 1994.

[29] M. B. Kennel, R. Brown, and H. D. I. Abarbanel. Determining minimum embedding dimension using a geometrical construction. *Physical Review A*, 45:3403–3411, 1992.

[30] D. Lawrence and L. Pao. `http://ece-www.colorado.edu/~pao/research.html`.

[31] T.-Y. Li and J. A. Yorke. Period three implies chaos. *American Mathematical Monthly*, 82:985–992, 1975.

[32] D. Lind and B. Marcus. *An Introduction to Symbolic Dynamics and Coding.* Cambridge University Press, 1995.

[33] L. Ljung, editor. *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, N.J., 1987.

[34] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[35] R. S. MacKay and J. D. Meiss, editors. *Hamiltonian Dynamical Systems.* Adam Hilger, 1987.

[36] D. Moore. *The Basic Practice of Statistics.* W. H. Freeman, 2000.

[37] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall, 1989.

[38] N. Packard, J. Crutchfield, J. Farmer, and R. Shaw. Geometry from a time series. *Physical Review Letters*, 45:712, 1980.

[39] T. S. Parker and L. O. Chua. *Practical Numerical Algorithms for Chaotic Systems.* Springer-Verlag, New York, 1989.

[40] F. J. Pineda and J. C. Sommerer. Estimating generalized dimensions and choosing time delays: A fast algorithm. In *Time Series Prediction: Forecasting the Future and Understanding the Past.* Santa Fe Institute Studies in the Sciences of Complexity, Santa Fe, NM, 1993.

[41] J. K. Roberge. *Operational Amplifiers: Theory and Practice.* Wiley, New York, 1975.

[42] T. Sauer, J. A. Yorke, and M. Casdagli. Embedology. *Journal of Statistical Physics*, 65:579–616, 1991.

[43] W. M. Siebert. *Circuits, Signals, and Systems.* M.I.T. Press, 1986.

[44] I. Stewart. *Does God Play Dice?: The Mathematics of Chaos.* Blackwell, Cambridge MA, 1989.

[45] S. H. Strogatz. *Nonlinear Dynamics and Chaos.* Addison-Wesley, Reading, MA, 1994.

[46] F. Takens. Detecting strange attractors in fluid turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence*, pages 366–381. Springer, Berlin, 1981.

[47] H. Tong. *Nonlinear Time Series Analysis: A Dynamical Systems Approach.* Oxford University Press, 1990.

[48] L. Trulla, A. Giuliani, J. Zbilut, and C. Webber. Recurrence quantification analysis of the logistic equation with transients. *Physics Letters A*, 223:255–260, 1996.

[49] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[50] J. W. Tukey. *Exploratory Data Analysis.* Addison Wesley, 1977.

[51] A. S. Weigend and N. S. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past.* Santa Fe Institute Studies in the Sciences of Complexity, Santa Fe, NM, 1993.

[52] D. S. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning About Physical Systems.* Morgan Kaufmann, San Mateo CA, 1990.

[53] T. Yamane. *Statistics, An Introductory Analysis.* Harper & Row, 1967.