Quantifying Time-Series Predictability through Structural Complexity

Joshua Garland,[1, *] Ryan James,[1, †] and Elizabeth Bradley[1, 2, ‡]

[1]*Department of Computer Science, University of Colorado at Boulder. Colorado, USA*
[2]*Santa Fe Institute, New Mexico, USA*

(Dated: June 17, 2014)

This paper provides insight into when, why, and how forecast strategies fail when they are applied to complicated time series. We conjecture that the inherent complexity of a time series — which results from the dimension, nonlinearity, and non-stationarity of the generating process, as well as from measurement issues like noise, aggregation, and finite data length — is both empirically quantifiable and directly correlated with predictability. In particular, we argue that *redundancy* is an effective way to measure complexity and predictive structure in a time series and that *weighted permutation entropy* is an effective way to estimate that complexity. To validate these conjectures, we study 120 different time-series data sets. For each time series, we construct predictions using a wide variety of forecast models, then compare the accuracy of the predictions with the permutation entropy of that time series. We use the results to develop a heuristic that can help practitioners recognize when a particular prediction method is not well matched to the task at hand: that is, when the time series has more predictive structure than that method can capture and exploit.

Keywords: entropy, permutation entropy, weighted permutation entropy, time-series analysis, predictability

PACS numbers: 05.45.Tp 89.70.-a 89.75.Kd 95.75.Pq

I. INTRODUCTION

Complicated time-series data are ubiquitous in modern scientific research. The complexity of these data spans a wide range. On the low end of this spectrum are time series that exhibit perfect predictive structure, i.e, signals whose future values can be successfully predicted from past values. Signals like this can be viewed as the product of an underlying process that generates information and/or transmits it from the past to the future in a perfectly predictable fashion. Constant or periodic signals, for example, fall in this class. On the opposite end of this spectrum are signals that are what we call *fully complex*, where the underlying generating process transmits no information at all from the past to the future. White noise processes fall in this class. In fully complex signals, knowledge of the past gives no insight into the future, regardless of what model one chooses to use. Signals in the midrange of this spectrum—e.g., deterministic chaos—pose interesting challenges from a modeling perspective. In these signals, enough information is being transmitted from the past to the future that an *ideal* model—one that captures the generating process—can forecast the future behavior of the observed system with high accuracy.

As a corollary of the undecidability of the halting problem [1], however, no single forecasting schema is ideal for all noise-free deterministic signals [2]—let alone all real-world time-series data sets. This leads naturally to an important and challenging question: given a complicated real-valued time series, does there exist a forecast model that can leverage the information (if any) that is being transmitted from past to future by the underlying generating process? A first step in answering this question is to reliably quantify where on the complexity spectrum a given time series falls; a second step is to determine how complexity and predictability are related. These are the goals of this paper.

In particular, we wish to determine—without

* joshua.garland@colorado.edu
† ryan.james@colorado.edu
‡ lizb@colorado.edu

knowing anything about the generating process—whether a time series is too complex to successfully predict. An important practical corollary to this is a strategy for assessing appropriateness of forecast methods. If the forecast produced by a particular method is poor but the time series contains a significant amount of predictive structure, one can reasonably conclude that that method is inadequate to the task and that one should seek another method.

The information in an observation can be partitioned into two pieces: redundancy and entropy generation [3]. Our approach exploits this decomposition in order to assess how much predictive structure is present in a signal—i.e., where it falls on the complexity spectrum mentioned above. We define *complexity* as a particular approximation of Kolmogorov-Sinai entropy [4]. That is, we view a random-walk time series (which exhibits high entropy) as purely complex, whereas a low-entropy periodic signal is on the low end of the complexity spectrum. We argue that an extension of *permutation entropy* [5]—a method for approximating the entropy through ordinal analysis—is an effective way to assess the complexity of a given time series. Permutation entropy is ideal for our purposes because it works with real-valued data and is known to converge to the true entropy value. Other existing techniques either require specific knowledge of the generating process or produce biased values of the entropy [6].

We focus on real-valued, scalar, time-series data. We make no assumptions about the properties of the generating process: whether it is linear, nonlinear, deterministic, stochastic, stationary, non-stationary, etc. To explore the relationship between complexity, predictive structure, and predictability, we generate forecasts for a variety of time-series datasets using four different prediction methods, then compare the accuracy of those predictions to the permutation entropy of the associated signals. This results in two primary findings:

1. The complexity of a noisy real-valued time series is *(i)* quantifiable by permutation entropy and *(ii)* correlated with prediction accuracy of an appropriate predictor.
2. The way information is generated and processed internally by a system plays a crucial role in the success of different forecasting schema—and in the choice of which one is appropriate for a given time series.

The forecast methods used in this study were chosen to be a representative sampling of standard prediction strategies, but they do not, of course, cover that space exhaustively. Our goal here is an empirical assessment of the relationship between predictability and complexity, not formal results about a "best" predictor for a given time series. From a practitioner's standpoint, it would be useful to know *a priori* if a time series contains enough predictive structure to make it worth spending the time and effort searching for a good forecast method. It would also be useful to know if a given method is inadequate—that is, if an alternative method could do better and so one should continue searching.

For the purposes of this study, we require a broad array of time-series datasets from across the complexity spectrum. Since forecasting is a real-world problem, we chose to study sensor data from a computer-performance experiment. While this is not a common laboratory experiment, it is a highly appropriate choice here. Computers are extremely complicated systems. Modern microprocessor chips contain multiple processing units and multi-layer memories, for instance, and they use complicated hardware/-software strategies to maximize performance by moving data and threads of computation across those resources. As a result, the processor and memory loads during the execution of even a very simple program can evolve in a very complicated fashion. Figure 1 shows a performance trace of a short program that repeatedly initializes the upper triangle of a matrix in column-major order. Although this time series appears to be periodic, it is actually chaotic [7], which places fundamental limits on predictability. Changes in the code can radically change this behavior; indeed, many performance traces exhibit interesting regime changes as a program moves through the different phases of its operation. The resulting traces span the whole range of the complexity spectrum, from completely predictable to completely

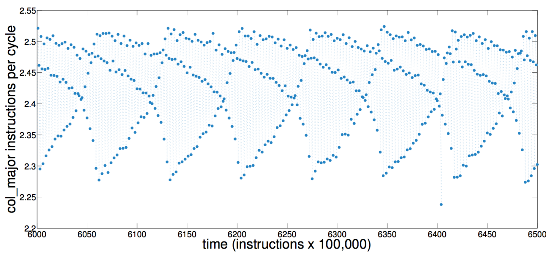


FIG. 1: A computer performance trace: the processor load of `col_major`, a simple C program that repeatedly initializes a matrix in column-major order, running on an Intel i7®-based machine.

unstructured, making them an ideal testbed for this study[1].

The rest of the paper is organized as follows. Section II discusses previous results on generating partitions, local modeling, and error distribution analysis, and situates this work in that context. Section III covers the experimental setup and methods used to collect each time series. Section IV describes the prediction models used in this study. In Section V, we review permutation entropy, the technique that we use to measure complexity. In Section VI, we estimate the complexity of each time series and compare that complexity to the accuracy of predictions produced by the methods of Section IV, operating on that time series. In Section VII, we discuss these results and their implications, and consider future areas of research.

II. RELATED WORK

Modeling time-series data for the purposes of prediction dates back at least to Yule's 1927 invention of autoregression [9]. Since then, hundreds, if not thousands, of strategies have been developed for a wide variety of prediction tasks. The purpose of this paper is not to add a new weapon to this arsenal, nor to assess or compare the effectiveness of existing methods. Our goals are more general: *(i)* to empirically quantify the predictive structure that is present in a real-valued scalar time series and *(ii)* to explore how the performance of prediction methods is related to that inherent complexity. It would, of course, be neither practical nor interesting to report results for every existing forecast method; instead, we choose a representative set, as described in Section IV.

Quantifying predictability, which is sometimes called "predicting predictability," is not a new problem. Most of the corresponding solutions fall into two categories that we call model-based error analysis and model-free information analysis. The first class focuses on errors produced by a fixed forecasting schema. This analysis can proceed locally or globally. The local version approximates error distributions for different regions of a time-series model using local ensemble in-sample forecasting[2]. These distributions are then used as estimates of out-of-sample forecast errors in those regions. For example, Smith *et al.* make in-sample forecasts using ensembles around selected points in order to predict the local predictability of that time series [10]. This approach can be used to show that different portions of a time series can exhibit varying levels of local predictive uncertainty. We expand on this finding later in this paper with a time series that exhibits interesting regime shifts.

Local model-based error analysis works quite well, but it only approximates the *local* predictive uncertainty *in relation to a fixed model.* It cannot quantify the inherent predictability of

[1] Predicting the *state* of a computer, of course, would amount to solving the halting problem. What we are doing here is predicting computer *performance*, which does not violate the Rice-Shapiro theorem [8].

[2] The terms "in sample" and "out of sample" are used in different ways in the forecasting community. Here, we distinguish those terms by the part of the time series that is the focus of the prediction: the observed data for the former and the unknown future for the latter. In-sample forecasts—comparisons of predictions generated from *part* of the observed time series—are useful for assessing model error and prediction horizons, among other things.

a time series and thus cannot be used to draw conclusions about predictive structure that may be usable by other forecast methods. Global model-based error analysis moves in this direction. It uses out-of-sample error distributions, computed *post facto* from a class of models, to determine which of those models was best. After building an autoregressive model, for example, it is common to calculate forecast errors and verify that they are normally distributed. If they are not, that suggests that there is structure in the time series that the model-building process was unable to capture and use. The problem with this approach is lack of generality. Normally distributed errors indicate that a model has captured the structure in the data insofar as is possible, *given the formulation of that particular model* (viz., the best possible linear fit to a nonlinear dataset). This gives no indication as to whether another modeling strategy might do better.

A practice known as deterministic vs. stochastic modeling [2, 11] bridges the gap between local and global approaches to model-based error analysis. The basic idea is to construct a series of local linear fits, beginning with a few points and working up to a global linear fit that includes all known points, and then analyze how the average out-of-sample forecast error changes as a function of number of points in the fit. The shape of such a "DVS" graph indicates the amounts of determinism and stochasticity present in a time series.

The model-based error analysis methods described in the previous three paragraphs are based on specific assumptions about the underlying generating process and knowledge about what will happen to the error if those assumptions hold or fail. Model-free information analysis moves away from those restrictions. Our approach falls into this class. Our goal is to empirically measure the inherent complexity of a time series, then correlate that complexity with the predictive accuracy of forecasts made using a number of different methods.

We build on the notion of *redundancy* that was introduced on page 2, which formally quantifies how information propagates forward through a time series: i.e., the mutual information between the past n observations and the current one. The redundancy of i.i.d. random processes, for instance, is zero, since all observations in such a process are independent of one another. On the other hand, deterministic systems—including chaotic ones—have high redundancy that is maximal in the infinite limit, and thus they can be perfectly predicted if observed for long enough [2]. In practice, it is quite difficult to estimate the redundancy of an arbitrary, real-valued time series. Doing so requires knowing either the Kolmogorov-Sinai entropy or the values of all positive Lyapunov exponents of the system. Both of these calculations are difficult—the latter particularly so if the data are very noisy or the generating system is stochastic.

Using entropy and redundancy to quantify the inherent predictability of a time series is not a new idea. Past methods for this, however, (e.g., [12, 13]) have hinged on knowledge of the *generating partition* of the underlying process, which lets one transform real-valued observations into symbols in a way that preserves the underlying dynamics [4]. Using a projection that is not a generating partition—e.g., simply binning the data—can introduce spurious complexity into the resulting symbolic sequence and thus misrepresent the entropy of the underlying system [6]. Generating partitions are luxuries that are rarely, if ever, afforded to an analyst, since one needs to know the underlying dynamics in order to construct one. And even if the dynamics are known, these partitions are difficult to compute and often have fractal boundaries [14].

We sidestep these issues by using a variant of the *permutation entropy* of Bandt and Pompe [5] to estimate the value of the Kolmogorov-Sinai entropy of a real-valued time series—and thus the redundancy in that data, which we show to be an effective proxy for predictability. This differs from existing approaches in a number of ways. It does not rely on generating partitions—and thus does not introduce bias into the results if one does not know the dynamics or cannot compute the partition. It makes no assumptions about, and requires no knowledge of, the underlying generating process: linear, nonlinear, the Lyapunov spectrum, etc. These features make our approach applicable to noisy real-valued time series from all classes of systems.

III. EXPERIMENTAL METHODS

The time-series data sets for these experiments were collected on an Intel Core® i7-2600-based machine running the 2.6.38-8 Linux kernel. This particular microprocessor chip has eight processing units, a clock rate of 3.40 GHz, and a cache size of 8192 KB. We gathered performance traces during the execution of three different programs—the simple `col_major` loop whose performance is depicted in Figure 1, as well as two more-complex programs: one from the SPEC 2006CPU benchmark suite (`403.gcc`), and one from the LAPACK linear algebra package (`dgesdd`). In these experiments, the scalar observation x_i was a measurement of the processor performance at time i during the execution of each program. To record these measurements, we used the `libpfm4` library, via PAPI 5.2 [15], to stop program execution at 100,000-instruction intervals—the unit of time in this paper—and read the contents of the CPU's onboard hardware performance monitors[3], which we had programmed to count how many instructions were executed in each clock cycle (IPC). For an in-depth description of this custom-measurement infrastructure, including discussion of the implications of the sampling interval, please see [7, 16, 17]. For statistical validation, we collected 15 performance traces from each of the three programs. These traces, and the processes that generated them, are described in more depth in the rest of this section.

`col_major` is a simple C program that repeatedly initializes the upper triangle of a 2048 × 2048 matrix in column-major order by looping over the following three lines of code:

```
for(i=0;i<2048;i++)
    for(j=i;j<2048;j++)
        data[j][i] = 0;
```

As mentioned in Section I, this simple program exhibits surprisingly complicated behavior [7]. A time series of the processor performance during the execution of this program is shown in Figure 2.

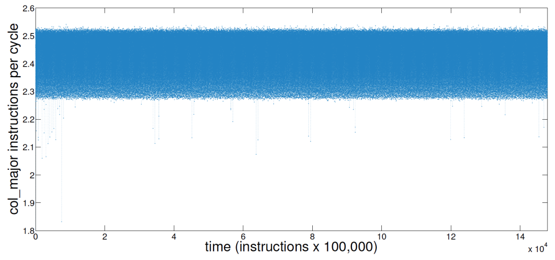


FIG. 2: The instructions executed per CPU clock cycle (IPC) as the `col_major` program runs. Each point is the average IPC in a 100,000 instruction period. Figure 1 is a closeup of the first part of this signal.

The SPEC CPU2006 benchmark suite [18] is a collection of complicated programs that are used in the computer-science community to assess and compare the performance of different computers. `403.gcc` is a member of that suite. It is a *compiler*: a program that translates code written in a high-level language (C, in the case of `403.gcc`) into a lower-level format that can be executed by the processor chip. Its behavior is far more complicated than that of `col_major`, as is clear from Figure 3. Unlike `col_major`, where the processor utilization is quite structured, `403.gcc`'s performance appears almost random.

`dgesdd` is a Fortran program from the LAPACK linear algebra package [19]. It calculates the singular value decomposition of a rectangular M by N matrix with real-valued entries. For our experiments, we choose $M = 750$ and $N = 1000$ and generate the matrix entries randomly. The behavior of this program as it computes the sin-

[3] These specialty registers are built into modern microprocessors for the purpose of monitoring and storage of performance data. They can be programmed to measure a wide array of processor and memory metrics besides the one used here.

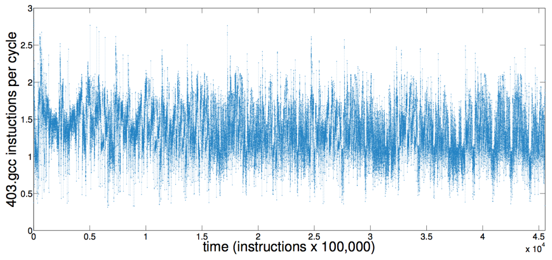


FIG. 3: The instructions per CPU clock cycle (IPC) during the execution of `403.gcc`. Each point is the average IPC in a 100,000 instruction period.

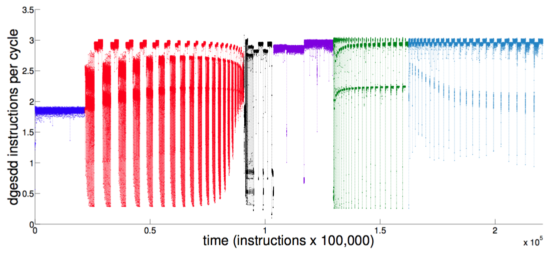


FIG. 4: The instructions per CPU clock cycle (IPC) during the execution of `dgesdd`. Each point is the average IPC in a 100,000 instruction period. The colors identify the different *segments* of the signal that are discussed in the text.

gular values of this matrix is very interesting, as shown in Figure 4. As the code moves though its different phases—diagonalizing the matrix, computing its transpose, multiplying, etc.—the processor utilization patterns change quite radically, as shown in Figure 4. For the first ∼21,000 measurements (21,000 × 100,000 instructions), roughly 1.8 instructions are executed per cycle, on the average, by the eight processing units on this chip. After that, the IPC moves through a number of different oscillatory regimes, which we have color-coded in the figure in order to make textual cross-references easy to track.

This wide range of behaviors provides a unique advantage, for the purposes of this paper, in that a number of different generating processes—with a wide range of complexities—are at work in different phases of a single time series. The complexity of `col_major` and `403.gcc` in Figures 2 and 3 appears to be far more consistent over time—probably the result of a single generating process. `dgesdd`, in contrast, has multiple regimes, each the result of a different generating process. To take advantage of this, we split the signal into six different segments, thereby obtaining an array of examples for the analyses in the following sections. For notational convenience, we refer to these 90 time-series data sets[4] as $\texttt{dgesdd}_i$, with $i \in \{1 \ldots 6\}$ where i corresponds to one of the six segments of the signal, ordered from left to right. These segments, which were determined visually, are shown in different colors in Figure 4. Visual decomposition is subjective, of course, particularly since the regimes exhibit some fractal structure. Thus, it may be the case that more than one generating process is at work in each of our segments. This is a factor in the discussion of Section VI.

IV. MODELING

In this section, we describe the four different forecasting methods used in this study, as well as the error metric used to evaluate their predictive accuracy. These methods include:

- The *random-walk* method, which uses the previous value in the observed signal as the forecast,
- The *naïve* method, which uses the mean of the observed signal as the forecast,
- The *ARIMA* (auto-regressive integrated moving average) method, a common linear forecast strategy, and
- The *LMA* (Lorenz method of analogues) method, which uses a near-neighbor forecast strategy on a dynamical reconstruction of the signal.

[4] 15 runs, each with six regimes

ARIMA is based on standard autoregressive techniques. LMA is designed to capture and exploit the deterministic structure of a signal from a nonlinear dynamical system. The naïve and random-walk methods, somewhat surprisingly, often outperform these more-sophisticated prediction strategies in the case of highly complex signals, as discussed below.

A. Two Simple Prediction Strategies

A random-walk predictor simply uses the last observed measurement as the forecast: that is, the predicted value p_i at time i is calculated using the following relation:

$$p_i = x_{i-1}$$

The prediction strategy that we refer to using the term "naïve" averages the prior observations to generate the forecast:

$$p_i = \sum_{j=1}^{i-1} \frac{x_j}{i-1}$$

While both of these methods are simplistic, they are not without merit. For a time series near the high end of the complexity spectrum—i.e., one that possesses very little predictive structure—these two methods can actually be the best choice. In forecasting currency exchange rates, for instance, sophisticated econometrics-based prediction models fail to consistently outperform the random-walk method [20, 21]. These signals are constantly changing, noisy, and possess very little predictive structure, but their variations are not—on the average—very large, so the random-walk method's strategy of simply guessing the last known value is not a bad choice. If a signal has a unimodal distribution with low variance, the naïve prediction strategy will perform quite well—even if the signal is highly complex—simply because the mean is a good approximation of the future behavior. Moreover, the naïve prediction strategy's temporal average effects a low-pass filtering operation, which can mitigate the complexity in signals with very little predictive structure.

Both of these methods have significant weaknesses, however. Because they do not model the temporal patterns in the data, or even the distribution of its values, they cannot track changes in that structure. This causes them to fail in a number of important situations. Random-walk strategies are a particularly bad choice for time series that change significantly at every time step. In the worst case—a large-amplitude square wave whose period is equivalent to twice the sample time—a random-walk prediction would be exactly 180 degrees out of phase with the true continuation. The naïve method would be a better choice in this situation, since it would always split the difference. It would, however, perform poorly when a signal has a number of long-lived regimes that have significantly different means. In this situation, the inertia of the naïve method's accumulating mean is a liability and the agility of the random-walk method is an advantage, since it can respond quickly to regime shifts.

Of course, methods that could capture and exploit the geometry of the data, or its temporal patterns, would be far more effective in the situations described in the previous paragraph. The ARIMA and LMA methods introduced in Sections IV B and IV C are designed to do exactly that. However, if a signal contains little predictive structure, forecast strategies like ARIMA and LMA have nothing to work with and thus will often be outperformed by the two simple strategies described in this section. This effect is explored further in Sections IV D and VI.

B. A Regression-Based Prediction Strategy

A simple and yet powerful way to capture and exploit the structure of data is to fit a hyperplane to the dataset and then use it to make predictions. The roots of this approach date back to the original autoregressive schema [22], which forecasts the next time step through a weighted average of past observations:

$$p_i = \sum_{j=1}^{i-1} a_j x_j$$

The weighting coefficients a_j are generally computed using either an ordinary least squares approach, or with the method of moments using the Yule-Walker equations. To account for noise in the data, one can add a so-called "moving average" term to the model; to remove nonstationarities, one can detrend the data using a differencing operation. A strategy that incorporates all three of these features is called a *nonseasonal ARIMA model*. If evidence of periodic structure is present in the data, a *seasonal ARIMA model*, which adds a sampling operation that filters out periodicities, can be a good choice.

There is a vast amount of theory and literature regarding the construction and use of models of this type; we refer the reader to [23] for an in-depth exploration. For the purposes of this paper, where the goal is to explore the relationship between predictability and complexity across a broad array of forecast strategies, seasonal ARIMA models are a good exemplar of the class of linear predictors. Fitting such a model to a dataset involves choosing values for the various free parameters in the autoregressive, detrending, moving average, and filtering terms. We employ the automated fitting techniques described in [24] to accomplish this. This procedure uses sophisticated methods—KPSS unit-root tests [25], a customization of the Canova-Hansen test [26], and the Akaike information criterion [27], conditioned on the maximum likelihood of the model fitted to the detrended data—to select good values for the free parameters of the ARIMA model.

ARIMA forecasting is a common and time-tested procedure. Its adjustments for seasonality, nonstationarity, and noise make it an appropriate choice for short-term predictions of time-series data generated by a wide range of processes. If information is being generated and/or transmitted in a nonlinear way, however, a global linear fit is inappropriate and ARIMA forecasts can be inaccurate. Another weakness of this method is prediction horizon: an ARIMA forecast is guaranteed to converge to the mean after some number of predictions, depending on model order. To sidestep this issue, we build forecasts in a stepwise fashion: i.e., fit the model to the existing data, use that model to perform a one-step prediction, rebuild it using the latest observations, and iterate until the desired prediction horizon is reached. (For consistency, we take the same approach with the other three models in this study as well, even though doing so amounts to artificially hobbling LMA.)

C. A Nonlinear Prediction Strategy

When the temporal progressions in a time series are produced by a deterministic nonlinear process, one can use a technique called delay-coordinate embedding to model the structure of the information generation and transmission occurring in the underlying process, then use that reconstruction to generate forecasts. This section discusses the theory and implementation of a prediction strategy that is based on this idea.

Delay-coordinate embedding [28–30] allows one to reconstruct a dynamical system's full state-space dynamics from a scalar time-series measurement—provided that some conditions hold regarding those data. Specifically, if the underlying dynamics and the measurement function—the mapping from the unknown state vector $\vec{X}$ to the observed value x_i—are both smooth and generic, Takens [30] formally proves that the delay-coordinate map

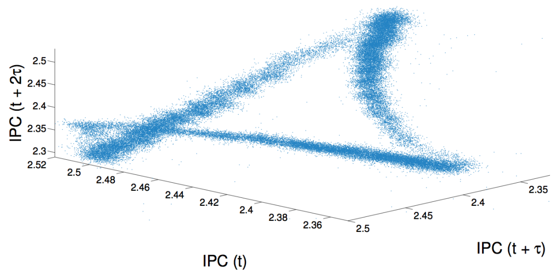
$$F(\tau, m)(\vec{X}) = ([x_i \; x_{i+\tau} \; \ldots \; x_{i+m\tau}])$$

from a d-dimensional smooth compact manifold M to $\mathbb{R}^{2d+1}$ is a diffeomorphism on M—in other words, that the reconstructed dynamics and the true (hidden) dynamics have the same topology. This is an extremely powerful result: among other things, it means that one can model the full system dynamics, up to diffeomorphism, without measuring—or even knowing—every one of its state variables.

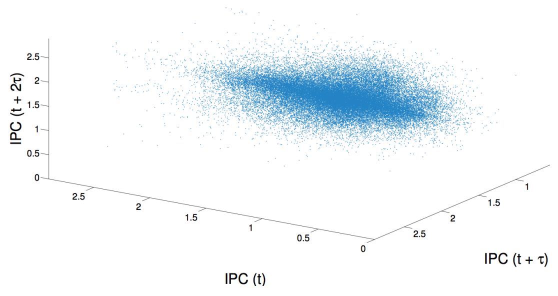
The first step in the delay-coordinate embedding process is to estimate values for the two free parameters in the map: the delay τ and the dimension m. We follow standard procedures for this, choosing the first minimum in the time-delayed mutual information as an estimate of τ [31] and using the false-near(est)-neighbor(s) method of [32] to estimate m. Some example plots of data from Figures 2-4, embedded following this procedure, are shown in Figure 5.

Geometric structure in these kinds of plots is an indication of structure in the information generation/transmission process that produced the time series. The dynamical systems community has developed a number of methods that leverage this structure to generate predictions (e.g., [2, 10, 33]). One of the most straightforward of these is the *Lorenz method of analogues* (LMA), which is essentially nearest-neighbor prediction in the embedded[5] space [36]. Even this simple algorithm—which builds predictions by finding the nearest neighbor in the embedded space of the given point, then taking that neighbor's path as the prediction—provides accurate forecasts when the generating process is a deterministic dynamical system.

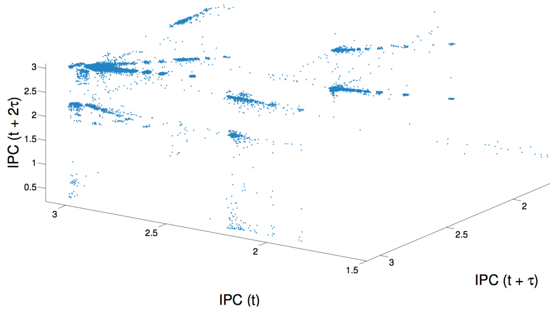
Since LMA does not rest on an assumption of linearity (as ARIMA does), it can handle both linear and nonlinear processes. If the underlying generating process is nondeterministic, however, it can perform poorly. In Figure 5b, for instance, little structure is visible, so LMA may not work well. More structure appears to be present in Figure 5c, but this reconstruction also appears to contain some noise. The question as to how much structure is present in a reconstruction—and how much of that structure can be captured and used by LMA—is apropos of the central question treated in this paper. It may be that `403.gcc` has some redundancy that LMA cannot exploit, or that the structure in $\texttt{dgesdd}_5$ is effectively obfuscated, from the standpoint of the LMA method, by noise. By quantifying the balance between redundancy (that is, predictive structure) and entropy for these real-valued time series, as shown in Section VI, we can begin to answer these questions.



(a) `col_major`



(b) `403.gcc`



(c) $\texttt{dgesdd}_5$

FIG. 5: 3D projections of delay-coordinate embeddings of the traces from (a) Figure 2 (b) Figure 3 and (c) the fifth (green) segment of Figure 4.

[5] Lorenz's original formulation used the full system state space; this method was first extended to embedded dynamics by Pikovsky [34], but is also related to the prediction work of Sugihara & May [35]

D. Assessing Prediction Accuracy

To study the relationship between predictability and complexity, we use the four methods outlined above to generate predictions of all 360 traces described in Section III, then calculate the error of the predictions with respect to the true continuations. Specifically, we split each time series into two pieces: the first 90%, referred to as the "initial training" signal and denoted $\{x_i\}_{i=1}^{n}$, and the last 10%, known as the "test" signal $\{c_j\}_{j=n+1}^{k+n+1}$. The initial training signal is used to build the model, following the procedures described in the previous section; that model is used to generate a prediction of the value of x_{n+1}, which is then compared to the true continuation, c_{n+1}. The model is then rebuilt using $\{x_i\}_{i=1}^{n+1}$ and the process repeats k times, out to the end of the observed time series. This "one step prediction" process is not technically necessary in the LMA method, whose ability to generate accurate predictions is limited only by the positive Lyapunov exponents of the system. However, the performance of the other three methods used here will degrade severely if the associated models are not periodically rebuilt. In order to make the comparison fair, we used an iterative one-step prediction schema *for all four methods*. This has the slightly confusing effect of causing the "test" signal to be used both to assess the accuracy of each model and for periodic refitting.

Figure I shows example forecasts made using all four methods for the `col_major`, `403.gcc`, and `dgesdd`$_5$ time series.

In these images, the vertical axis is the prediction p_j and the horizontal axis is the true continuation c_j. On such a plot, a perfect prediction would lie on the diagonal. LMA, for instance, generates a very accurate prediction of the `col_major` data, while ARIMA does not. Horizontal lines result when a constant predictor (e.g., naïve) is used on a non-constant signal. Point clouds reflect the structure of the distribution of the errors—roughly normally distributed, for example, in the case of LMA on `403.gcc`. Note that the shapes of Figures 6a and 6g are reminiscent of the projected embedding in Figure 5a. For a random-walk predictor, a p_j vs. c_j plot is equivalent to a two-dimensional embedding with $\tau = 1$. For ARIMA, the correspondence is not quite as simple, since the p_j values are linear combinations of a number of past values of the c_j, but the effect is largely the same[6].

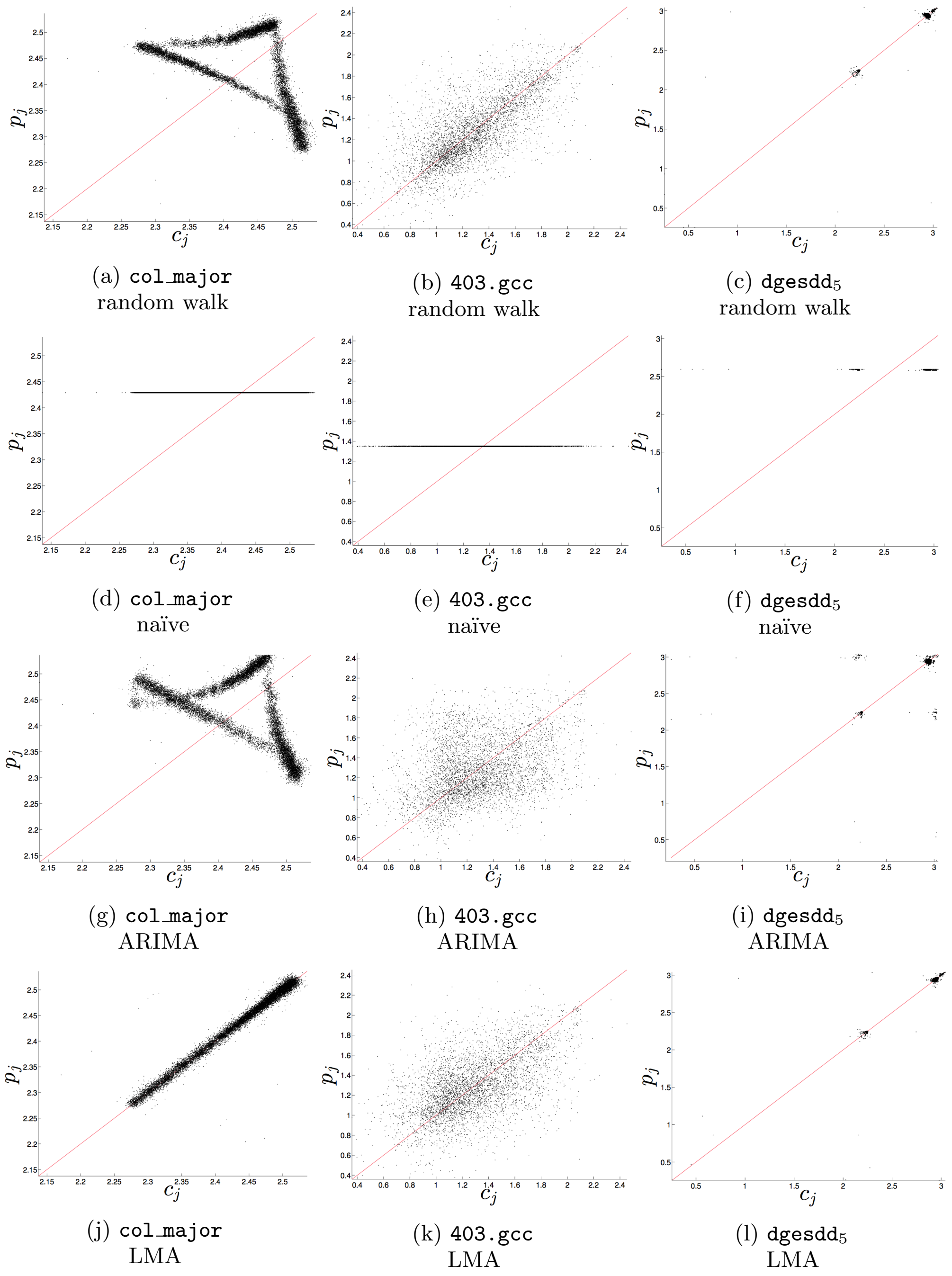
As a numerical measure of prediction accurary, we calculate the mean absolute scaled error (MASE) between the true and predicted signals:

$$MASE = \sum_{j=n+1}^{k+n+1} \frac{|p_j - c_j|}{\frac{k}{n-1}\sum_{i=2}^{n}|x_i - x_{i-1}|}$$

This error metric was introduced in [37] as a "generally applicable measurement of forecast accuracy without the problems seen in the other measurements." MASE is a normalized measure: the scaling term in the denominator is the average in-sample forecast error for a random-walk prediction over the initial training signal $\{x_i\}_{i=1}^{n}$. That is, $MASE < 1$ means that the prediction error in question was, on the average, smaller than the error of a random-walk forecast on the same data. Analogously, $MASE > 1$ means that the corresponding prediction method did *worse*, on average, than the random-walk method. We chose this error metric because it allows for fair comparison across varying methods, prediction horizons, and signal scales.

MASE scores for all 360 experiments are tabulated in the three middle columns in Table II. Comparing these with the geometry of the plots in Figure I, one can see some obvious correspondences. The average LMA MASE score for the `col_major` signals was 0.050, for instance, while ARIMA scored much worse (0.599). That is, LMA performed roughly 20 times better on `col_major` signals than a random-walk predictor, while ARIMA only outperformed random walk by a factor of 1.7. This is in accordance with the visual appearance of the corresponding

[6] The temporal *ordering* of the points on the ARIMA p_j vs. c_j plot does not match that of a projected embedding of the time series, however.



(a) `col_major` random walk

(b) `403.gcc` random walk

(c) `dgesdd`$_5$ random walk

(d) `col_major` naïve

(e) `403.gcc` naïve

(f) `dgesdd`$_5$ naïve

(g) `col_major` ARIMA

(h) `403.gcc` ARIMA

(i) `dgesdd`$_5$ ARIMA

(j) `col_major` LMA

(k) `403.gcc` LMA

(l) `dgesdd`$_5$ LMA

FIG. 6: Predicted (p_j) versus true values (c_j) for forecasts of `col_major`, `403.gcc`, and `dgesdd`$_5$ and all four prediction strategies.

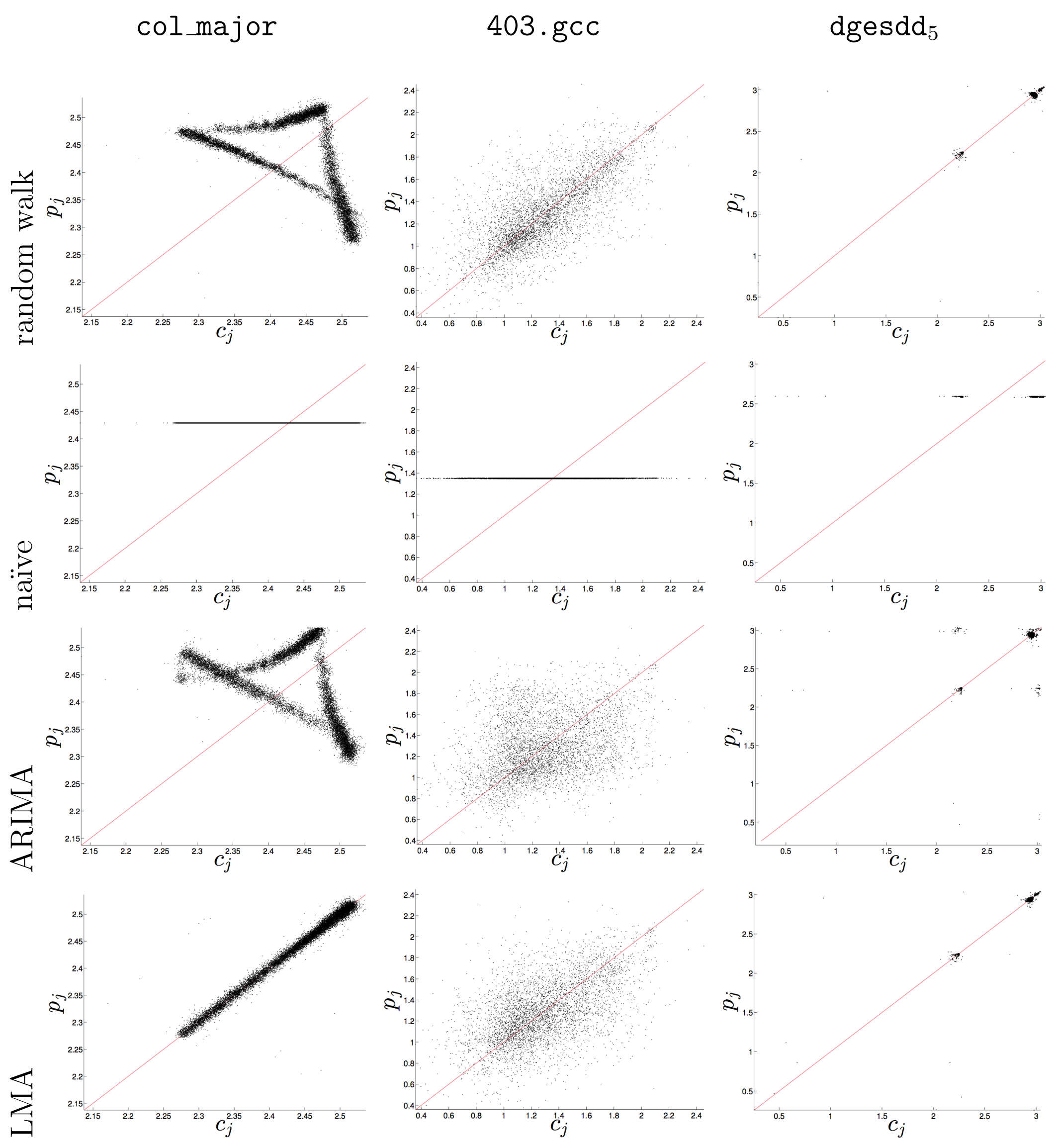


TABLE I: Predicted (p_j) versus true values (c_j) for forecasts of `col_major`, `403.gcc`, and `dgesdd`$_5$ and all four prediction strategies.

images in Figure I. While the correspondence between MASE score and the visual appearance of these kinds of plots is clear cut in this case, that is not always so. The plots of LMA predictions of `col_major` and `dgesdd`$_5$ both lie near the diagonal, for instance, but the corresponding MASE scores were very different: 0.050 for `col_major` and 0.718 for `dgesdd`$_5$ (resp., 20 and 1.4 times better than random-walk forecasts of the same signals). This is a result of the normalization in the MASE score calculation. Recall from Section IV A that the random-walk method

TABLE II: Mean absolute scaled error (MASE) scores and weighted permutation entropies for all eight processes studied in this paper

Signal	LMA MASE	ARIMA MASE	naïve MASE	Weighted Permutation Entropy
`col_major`	0.050 ± 0.002	0.599 ± 0.211	0.571 ± 0.002	0.513 ± 0.003
`403.gcc`	1.530 ± 0.021	1.837 ± 0.016	0.951 ± 0.001	0.943 ± 0.001
$\texttt{dgesdd}_1$	0.827 ± 0.076	0.714 ± 0.075	2.676 ± 4.328	0.957 ± 0.016
$\texttt{dgesdd}_2$	1.279 ± 0.020	2.163 ± 0.027	3.054 ± 0.040	0.846 ± 0.004
$\texttt{dgesdd}_3$	0.619 ± 0.021	0.713 ± 0.010	31.386 ± 0.282	0.716 ± 0.006
$\texttt{dgesdd}_4$	0.779 ± 0.036	0.979 ± 0.032	2.661 ± 0.074	0.825 ± 0.008
$\texttt{dgesdd}_5$	0.718 ± 0.048	2.370 ± 0.051	20.870 ± 0.192	0.678 ± 0.007
$\texttt{dgesdd}_6$	0.739 ± 0.068	1.438 ± 0.061	2.197 ± 0.083	0.748 ± 0.011

performs especially poorly on signals that oscillate rapidly. `col_major` fits this description, so the random-walk error on this signal—the denominator of the three MASE scores in the first row of the Table—is pathologically high, which skews those scores down. Random-walk prediction is very effective for the $\texttt{dgesdd}_5$ signal, on the other hand, so the denominator is small and the MASE scores are skewed upwards. Normalization is, as is often the case, a double-edged sword. Here, it facilitates comparisons across signals of different types and lengths, but its particular pathologies must be taken into account, as discussed at more length in Section VI.

V. MEASURING STRUCTURAL COMPLEXITY

Estimating the entropy of an arbitrary, real-valued time series is a challenging problem. Our approach to this problem, which is the topic of this section, draws upon methods and results from a variety of fields including time-series analysis, dynamical systems, and stochastic processes.

For the purposes of this paper, we view the Shannon entropy—in particular its growth rate with respect to word length (the *Shannon entropy rate*)—as a measure of complexity and unpredictability in a time series. Time-series consisting of i.i.d. random variables, such as white noise, have maximal entropy rates, whereas highly structured time-series, for example periodic, have very low (or zero) entropy rates. A time series with a high entropy rate is almost completely unpredictable; conversely, one with low entropy rate is often quite predictable. This can be made more rigorous: Pesin's relation [38] states that in chaotic dynamical systems, the Kolmogorov-Sinai (KS) entropy is equal to the sum of the positive Lyapunov exponents, λ_i. The Lyapunov exponents directly quantify the rate at which nearby states of the system diverge with time: $|\Delta x(t)| \approx e^{\lambda t} |\Delta x(0)|$. The faster the divergence, the larger the entropy. The KS entropy is defined as the supremum of the Shannon entropy rates of all partitions [39]. The partition that achieves this supremum is the *generating partition* discussed in Section II.

From a different point of view, we can consider the information (as measured by the Shannon entropy) contained in a single observable, say the *present*, of the system. This information can be partitioned into two components: the information shared with past observations—e.g., the mutual information between the past and present—and the information in the present that is not contained in the past (aka "the conditional entropy of the present given the past"). The first part is known as the *redundancy*, as it is information in the present that is also in the past. The second part is the aforementioned *Shannon entropy rate.* We conjecture that the more redundancy in a signal, the more predictable it is. Redundancy is critical to prediction, in general. And the specific *form* of the redundancy dictates whether a particular prediction method will work well or poorly on the corresponding signal. A linear method cannot capture or make use of nonlinear redundancy, for instance. This issue is central to the claims in this paper and the discussion in the following section.

Previous approaches to measuring temporal complexity via the Shannon entropy rate [12, 13] required categorical data: $x_i \in \mathcal{S}$ for some finite or countably infinite *alphabet* $\mathcal{S}$. Data taken from real-world systems are, however, effectively[7] real-valued. To analyze real-valued data using a method that requires categorical values, one must discretize the data—typically by binning. Unfortunately, this is rarely a good solution, as the binning of the values introduces spurious dynamics [6]. The field of symbolic dynamics offers discretization methods that do not disturb the intrinsic behavior. These methods are, however, fragile in the face of noise; worse yet, they require knowledge of the underlying system. This is not useful in our context, where we want to measure the entropy to quantify predictability of an arbitrary time-series.

Bandt and Pompe introduced the *permutation entropy* (PE) as a "natural complexity measure for time series" [5]. The permutation entropy employs a method of discretizing real-valued time series that follows the intrinsic behavior of the system under examination. It has many advantages, including being robust to observational noise, and its application does not require any knowledge of the underlying mechanisms. Rather than looking at the statistics of sequences of values, as is done when computing the Shannon entropy, permutation entropy looks at the statistics of the *orderings* of sequences of values using ordinal analysis. Ordinal analysis of a time series is the process of mapping successive time-ordered elements of a time series to their value-ordered permutation of the same size. By way of example, if $(x_1, x_2, x_3) = (9, 1, 7)$ then its *ordinal pattern*, $\phi(x_1, x_2, x_3)$, is 231 since $x_2 \leq x_3 \leq x_1$. The ordinal pattern of the permutation $(x_1, x_2, x_3) = (9, 7, 1)$ is 321.

[7] Measurements from finite-precision sensors are discrete, but data from modern high-resolution sensors are, for the purposes of entropy calculations, effectively continuous.

Definition (Permutation Entropy)**.** *Given a time series $\{x_i\}_{i=1,\ldots,N}$. Define $\mathcal{S}_\ell$ as all $\ell!$ permutations π of order ℓ. For each $\pi \in \mathcal{S}_\ell$ we determine the relative frequency of that permutation occurring in $\{x_i\}_{i=1,\ldots,N}$:*

$$P(\pi) = \frac{|\{i | i \leq N - \ell, \phi(x_{i+1}, \ldots, x_{i+\ell}) = \pi\}|}{N - \ell + 1}$$

where $P(\pi)$ quantifies the probability of an ordinal and $|\cdot|$ is set cardinality. The permutation entropy *of order $\ell \geq 2$ is defined as*

$$H(\ell) = -\sum_{\pi \in \mathcal{S}_\ell} P(\pi) \log_2 P(\pi)$$

Notice that $0 \leq H(\ell) \leq \log_2(\ell!)$ [5]. With this in mind, it is common in the literature to normalize permutation entropy as follows: $\frac{H(\ell)}{\log_2(\ell!)}$. With this convention, "low" PE is close to 0 and "high" PE is close to 1. Finally, it should be noted that the permutation entropy has been shown to be identical to the Kolmolgorov-Sinai entropy for many large classes of systems [40], as long as observational noise is sufficiently small. As mentioned before, this is equal to the Shannon entropy rate of a generating partition of the system. This transitive chain of equalities, from permutation entropy to Shannon entropy rate via the KS entropy, allows us to approximate the redundancy—being the dual of the Shannon entropy rate—of a signal by $1 - \frac{H(\ell)}{\log_2(\ell!)}$.

Here we will be utilizing a variation of the basic permutation entropy technique, the *weighted permutation entropy* (WPE), which was introduced in [41]. The intent behind the weighting is to correct for observational noise that is larger than the trends in the data, but smaller than the larger scale features. Consider, for example, a signal that switches between two fixed points and contains some additive noise. The PE is dominated by the noise about the fixed points driving it to ≈ 1, which in some sense hides the fact that the signal is actually quite structured. In this situation, the terms of the weighted permutation entropy are dominated by the switching rather than the stochastic fluctuations. To accomplish this, the *weight* of a permutation is taken into account:

$$w(x_{i+1}^{\ell}) = \frac{1}{\ell} \sum_{j=i+1}^{i+\ell} \left(x_j - \bar{x}_{i+1}^{\ell}\right)^2$$

where x_{i+1}^{ℓ} is a sequence of values $x_{i+1}, \ldots, x_{i+\ell}$, and $\bar{x}_{i+1}^{\ell}$ is the arithmetic mean of those values.

The weighted probability of a permutation is defined as:

$$P_w(\pi) = \frac{\displaystyle\sum_{i \leq N-\ell} w(x_{i+1}^{\ell}) \cdot \delta(\phi(x_{i+1}^{\ell}), \pi)}{\displaystyle\sum_{i \leq N-\ell} w(x_{i+1}^{\ell})}$$

where $\delta(x, y)$ is 1 if $x = y$ and 0 otherwise. Effectively, this weighted probability enhances permutations that are involved in "large" features and de-emphasizes permutations that are small in amplitude relative to the features of the time series. Using the standard form of an entropy, the weighted permutation entropy is:

$$H_w(\ell) = -\sum_{\pi \in \mathcal{S}_\ell} P_w(\pi) \log_2 P_w(\pi),$$

which can also be normalized by dividing by $\log_2(\ell!)$, making $0 \leq \text{WPE} \leq 1$. This normalization is used in all the results that follow.

In practice, calculating permutation entropy and weighted permutation entropy involves choosing a good value for the word length ℓ. The primary consideration in that choice is that the value be large enough that forbidden ordinals are discovered, yet small enough that reasonable statistics over the ordinals are gathered. If an average of 100 counts per ordinal is considered to be sufficient, for instance, then $\ell = \text{argmax}_{\hat{\ell}}\{N \gtrapprox 100\hat{\ell}!\}$. In the literature, $3 \leq \ell \leq 6$ is a standard choice—generally without any formal justification. In theory, the permutation entropy should reach an asymptote with increasing ℓ, but that can require an arbitrarily long time series. In practice, what one should do is calculate the *persistent* permutation entropy by increasing ℓ until the result converges, but data length issues can intrude before that convergence is reached. We used this approach to choose $\ell = 6$ for the experiments in the following section. This value represents a good balance between accurate ordinal statistics and finite-data effects.

As a final note: there has been prior work under a similar title to ours [42], but there are only superficial similarities between the two research projects. Haven *et al.* utilize the relative entropy to quantify the difference in predictability between two distributions: one evolved from a small ensemble of past states using the known dynamical system, and the other the observed distribution. Our work quantifies the predictability of a single observed time series using weighted permutation entropy and makes no assumptions about the generating process.

VI. PREDICTABILITY, COMPLEXITY, AND PERMUTATION ENTROPY

In this section, we offer an empirical validation of the two conjectures offered in Section I, namely:

1. That the weighted permutation entropy (WPE) of a noisy real-valued time series is correlated with prediction accuracy—i.e., that the predictable structure in a time-series data set can be quantified by its WPE.

2. That the way information is generated and processed by a system is correlated with the effectiveness of a given predictor on time-series data from that system

The experiments below involve three different prediction methods applied to time-series data from eight different systems: `col_major`, `403.gcc`, and the six different regimes of the `dgesdd` signal in Figure 4. The objective of these experiments was to explore how prediction accuracy is related to WPE, and how that relationship depends on the generating process and the prediction method. Working from the first 90% of each signal, we generated a prediction of the last 10% using the naïve, ARIMA, and LMA prediction methods, as described in Section IV D, then calculated the MASE value of those predictions. We also calculated the WPE of each time series using a wordlength of six, as described in Section V. In order to assess the run-to-run variability of these results, we repeated all of these calculations on 15 separate trials: i.e., 15 different runs of each program.

Figure 7 shows the *best* of these predictions for each trial of each system: that is, the lowest error over all three methods for each of the eight programs. The WPE is plotted against the corresponding MASE value here in order to bring out the correlation between these two quantities. As the plot shows, the best-case prediction error

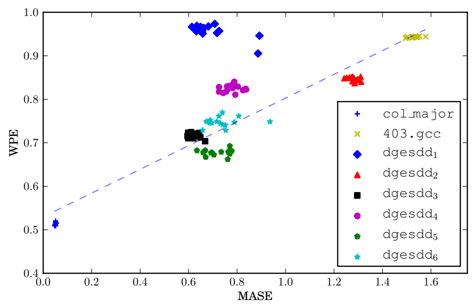


FIG. 7: Weighted permutation entropy vs. mean absolute scaled error (MASE) of the best prediction of each trial for each system. The dashed line is a least-squares linear fit of all the points except those from dgesdd$_1$, which we have excluded for reasons explained in the text.

for each of these eight systems is roughly proportional to the weighted permutation entropy, which is consistent with our first conjecture. The dashed line in the figure, a linear least-squares fit to these best-case errors[8] captures this rough proportionality. This is not a formal result. The three forecast methods used here were chosen to span the space of standard prediction strategies, but they do not cover that space exhaustively. Our goal here is an empirical assessment of predictability and complexity, not formal results about a "best" predictor for a given time series. There may be other methods that produce lower MASE values than those in Figure 7, but the sparseness of the points in the upper-left and lower-right quadrants of this plot strongly suggests that the underlying predictability of a time series is inversely correlated with its WPE. The rest of this section describes our results in more detail—including the measures taken to assure meaningful comparisons across methods, trials, and programs—and elaborates on the meaning of the dashed line in the figure.

Figure 8 shows WPE vs. MASE plots for the full set of experiments. There are 15 points in each cluster, one for each trial. (The points in Figure 7 are the leftmost of the points for the corresponding system in any of the three plots in Figure 8.) The WPE values do not vary very much across trials. For most traces, the variance in MASE scores is low as well, resulting in small, tight clusters. In some cases—ARIMA predictions of `col_major`, for instance—the MASE variance is larger, which spreads out the clusters horizontally. The distribution of the cluster *positions* is quite different in the three plots. The mean MASE scores of predictions generated with the nonlinear LMA method are generally closer to the dashed line; the ARIMA method clusters are more widely spread, the naïve clusters even more so. This geometric validation of the second conjecture in this paper is discussed in more detail in the following paragraphs.

Though `col_major` is a very simple program, its dynamics are actually quite complicated, as discussed in Section I. Recall from the graphical representations in the left-hand column of Figure I that the naïve and ARIMA prediction methods do not perform as well on this signal as the nonlinear LMA method. The MASE scores of the naïve and ARIMA predictions are 0.571 ± 0.002 and 0.599 ± 0.211, respectively, across all 15 trials. That is, these methods are

[8] with the exception of dgesdd$_1$, for reasons described below

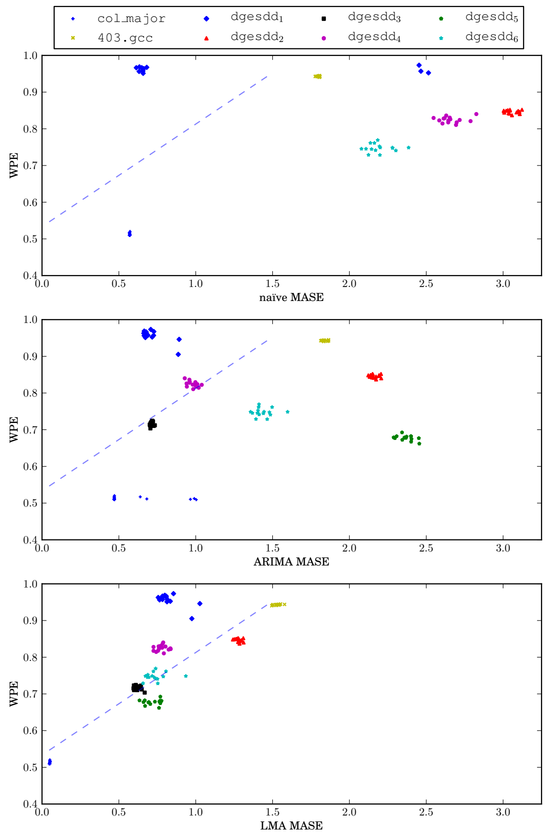


FIG. 8: WPE vs. MASE for all trials, methods, and systems—with the exception of `dgesdd`$_1$, `dgesdd`$_3$, and `dgesdd`$_5$ are omitted from the top plot for scale reasons. Numerical values, including means and standard deviations of the errors, can be found in Table II. The dashed line is the same as in the previous figure.

performing only ≈ 1.7 times better than the random-walk method, a primitive strategy that simply uses the current value as the prediction. However, the WPE value for the `col_major` trials is 0.513 ± 0.003, which is in the center of the complexity spectrum described in Section I.

This disparity—WPE values that suggest a high rate of forward information transfer in the signal, but predictions with comparatively poor MASE scores—is obvious in the geometry of the top two images in Figure 8, where the `col_major` clusters are far to the right of the dashed line. *This indicates that these methods are not leveraging the available information in the signal.* The dynamics of `col_major` may be complicated, but they are not unstructured. This signal is nonlinear and deterministic[7], and if one uses a prediction technique that is based a nonlinear model (LMA)—rather than a method that simply predicts the running mean (naïve) or one that uses a linear model (ARIMA)—the MASE score is much better: 0.050 ± 0.001. This prediction is 20 times more accurate than a random-walk forecast, which is more in line with the level of predictive structure that the low WPE value suggests is present in the signal.

The dashed line in Figures 7 and 8 is a heuristic that captures the argument at the end of the previous paragraph. It can allow practitioners to recognize when a prediction method is not well matched to the task at hand: that is, when the time series has more predictive structure than the method is able to use. When the predictions are well below that line (as in the top and middle images in Figure 8), one should try a different method. Conversely, the position of the LMA cluster—well to the left of the other two, near the dashed line—reflects this method's ability to capture and exploit the structure that is present in this signal.

Note that the `col_major` points (blue + icons) are clustered very tightly in the lower left quadrant of the LMA and naïve MASE vs. WPE plots, but spread out horizontally in the ARIMA plot. This is because of the way that these models are built [24]. If a KPSS test of the time series in question indicates that it is nonstationary, an integration term is added to the model. This test gives mixed results in the case of the `col_major` process, flagging five of the 15 trials as stationary and ten as nonstationary. ARIMA models without an integration term perform more poorly on this signal, which increases the error, thereby spreading out the points[9].

[9] We tested this hypothesis by forcing the inclusion of an integration term in the five cases where a KPSS test

The WPE of `dgesdd`$_5$ (0.677 ± 0.006) is higher than that of `col_major`. This indicates that the rate of forward information transfer of the underlying process is lower, but that time-series data from this system still contain a significant amount of structure that can, in theory, be leveraged to predict the future course of the time series. As mentioned above, though, the effectiveness of any prediction strategy depends on how well it leverages the available information: methods that use linear models, for instance, may fail when applied to nonlinear processes. This match between model and task is an issue in the `dgesdd`$_5$ experiments as well. The MASE scores of the naïve and ARIMA predictions for this system are 20.870 ± 0.192 and 2.370 ± 0.051, respectively: that is, 20.87 and 2.37 times worse than a simple random walk forecast of the same signals[10]. As before, the positions of these points on a WPE vs. MASE plot—significantly below and to the right of the dashed line—should suggest to a practitioner that there is more structure in this signal than the corresponding method is able to leverage, and that a different method might do better. Indeed, for `dgesdd`$_5$, the LMA method produces a MASE score of 0.718 ± 0.048 and a cluster of results that is much closer to the dashed line on the WPE-MASE plot. Again, this validates our second conjecture: the LMA method can capture and reproduce the way in which the `dgesdd`$_5$ system processes information, but the naïve and ARIMA prediction methods cannot.

The WPE of `403.gcc` is higher still: 0.943 ± 0.001. This system transmits very little information forward in time and provides almost no structure for prediction methods to work with. Here, the naïve method performs best ($MASE = 0.951 \pm 0.001$), simply because its calculation of the mean filters out the noise in the signal. Since fitting a hyperplane using least squares should have similar effects, the fact that LMA outperforms ARIMA (1.530 ± 0.021 vs. 1.837 ± 0.016) is somewhat counterintuitive. However, the small amount of predictive structure that is present in this signal is nonlinear (cf., [7]), and LMA is designed to capture and exploit that kind of structure.

`dgesdd`$_1$—the dark blue segment of Figure 4—behaves very differently than the other seven systems in this study. Though its weighted permutation entropy is very high (0.957 ± 0.016), two of the three prediction methods do quite well on this signal, yielding MASE scores of 0.827 ± 0.076 (LMA) and 0.714 ± 0.075 (ARIMA). This pushes the corresponding clusters of points in the bottom two images in Figure 8 well above the trend followed by the other seven signals. The MASE scores of the predictions that were produced by the naïve method for this system, however, are highly inconsistent. The majority of the blue diamond-shaped points on the top plot in Figure 8 are clustered near a MASE score of 0.6, which is better than LMA and ARIMA. In five of the 15 `dgesdd`$_1$ trials, however, there were step changes in the signal. The naïve method has a very difficult time with signals like this, particularly if there are multiple step changes. This raised the MASE scores of these trials, pushing the corresponding points to the right[11], and in turn raising both the mean and variance of this set of trials.

The effects described in the previous paragraph are exacerbated by the way MASE is calculated. Recall that MASE scores are scaled *relative to a random-walk forecast*, which uses the current value as the prediction. This strategy works very badly on signals with frequent, large, rapid transitions. Consider a signal that oscillates from one end of its range to the other at every step. A signal like this will have a low WPE, much like `col_major`. However, a random-walk forecast of this signal will be 180 degrees

indicated that such a term was not needed. This action removed the spread, pushing all 15 of the `col_major` ARIMA points in Figure 8 into a tight cluster.

[10] The naïve MASE score is large because of the bimodal nature of the distribution of the values of the signal, which makes guessing the mean a particularly bad strategy. The same thing is true of the `dgesdd`$_3$ signal.

[11] This includes the cluster of three points near MASE ≈ 2.5, as well as two points that are beyond the domain of the graph, at MASE 11.2 and 14.8.

out of phase with the true continuation. Since random-walk error appears in the denominator of the MASE score, this effect can shift points leftwards on a WPE vs. MASE plot, and that is exactly why the $\texttt{dgesdd}_1$ clusters for ARIMA and LMA are above the dashed line in Figure 8. This time series, which is shown in closeup in Figure 9, is not quite to the level of the worst-

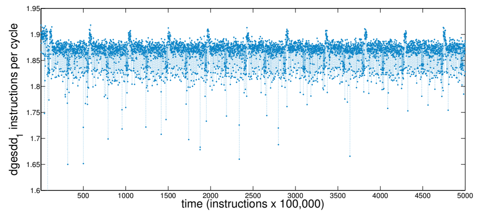


FIG. 9: A small portion of the $\texttt{dgesdd}_1$ time series

case signal described above, but it still poses a serious challenge to random-walk prediction. It is dominated by a noisy regime (between ≈1.86 and ≈1.88 on the vertical scale in Figure 9), punctuated by short excursions above 1.9. In the former regime, which makes up more than 80% of the signal, there are frequent dips to 1.82 and occasional larger dips below 1.8. These single-point dips are the bane of random-walk forecasting. In this particular case, roughly 40% of the forecasted points are off by the width of the associated dip, which skews the associated MASE scores. Signals like this are also problematic for the naïve prediction strategy, since the outliers have significant influence on the mean. This compounds the effect of the skew in the scaling factor and exacerbates the spread in the $\texttt{dgesdd}_1$ MASE values. For all of these reasons, we view $\texttt{dgesdd}_1$ as an outlier and exclude it from the fit calculation of the dashed line in Figure 7.

VII. CONCLUSIONS & FUTURE WORK

Forecast strategies that are designed to capture predictive structure are ineffective when signal complexity outweighs information redundancy. This poses a number of serious challenges in practice. Without knowing anything about the generating process, it is difficult to determine how much predictive structure is present in a given time series. And even if predictive structure exists, a given forecast method may not work, simply because it cannot exploit the structure that is present (e.g., a linear model of a nonlinear process). If a forecast model is not producing good results, a practitioner needs to know why: is the reason that the time series contains no predictive structure—i.e., that no model will work—or is the model that s/he is using simply not good enough?

In this paper, we have argued that redundancy [3] is a useful definition of the inherent predictability of a time series. To operationalize that definition, we use an approximation of the Kolmogorov-Sinai entropy [4], estimated using a weighted version of the permutation entropy of [5]. This WPE technique—an ordinal calculation of forward information transfer in a time series—is ideal for our purposes because it works with real-valued data and is known to converge to the true entropy value. Using a variety of forecast models and more than a hundred time- series data sets from a computer-performance experiment, we have shown that prediction accuracy is indeed correlated with weighted permutation entropy: the higher the WPE, in general, the higher the prediction error. Supporting our hypothesis, forecast errors for the `col_major` and $\texttt{dgesdd}_5$ signals are higher than the corresponding WPEs suggest—except for the nonlinear LMA predictor. There are some exceptions: $\texttt{dgesdd}_1$ is easier to predict than its WPE value suggests using any of the three forecast strategies studied here. On the whole, though, signals with higher weighted permutation entropy are harder to predict.

An important practical corollary to this empirical correlation of predictability and WPE is a practical strategy for assessing appropriateness of forecast methods. If the forecast produced by a particular method is poor but the time series contains a significant amount of predictive structure, one can reasonably conclude that

that method is inadequate to the task and that one should seek another method. The nonlinear LMA method, for instance, performs better in most cases because it is more general. (This is particularly apparent in the `col_major` and `dgesdd`$_5$ examples.) The naïve method, which simply predicts the mean, can work very well on noisy signals because it effects a filtering operation. The simple random-walk strategy outperforms LMA, ARIMA, and the naïve method on the `403.gcc` signal, which is extremely complex—i.e., extremely low redundancy. The dashed line in Figures 7 and 8 operationalizes the discussion in this paragraph. It is a preliminary, but potentially useful, empirical guideline for knowing when a model is not well-matched to the task at hand: a MASE score that is more than $3.58 \cdot \text{WPE} - 1.91$, where WPE is that of the corresponding signal, is an indication that that time series has more predictive structure than that forecast model can capture and exploit.

Given that information is a fundamental limitation in predictability, then gathering and using more information is an obvious next step. But there is an equally obvious tension here between data length and prediction speed: a forecast that requires half a second to compute is not useful for the purposes of real-time control of a computer system with a MHz clock rate, for instance. Another alternative is to sample several system variables simultaneously and build multivariate models. This is a particular challenge in nonlinear LMA-type models, since multivariate delay-coordinate embedding (e.g., [43, 44]) can be computationally prohibitive. We are working on alternative methods that sidestep that complexity.

Nonstationarity is a serious challenge in any time-series modeling problem. Indeed, one of the first applications of permutation entropy was to recognize the regime shift in brainwave data that occurs when someone has a seizure [45]. Recall that the signal in Figure 4 was especially useful for the study in this paper because it contained a number of different regimes. We segmented this signal visually, but one could imagine using permutation entropy to do so instead. Automating regime-shift detection would be an important step towards a fully adaptive modeling strategy, where old models are discarded and new ones are rebuilt whenever the time series enters a new regime. WPE would be particularly powerful in this scenario, as its value can not only help with regime-shift detection, but also suggest what kind of model might work well in each new regime.

All of the experiments described in this paper involve a particular physical system. We chose that system because its behavior spans a broad spectrum of classical time series patterns. Since these patterns (dynamical chaos, periodicities, noise, etc.) are generic, we believe that our results will generalize beyond this particular system. Testing that claim with different examples is another important next step—and one that will be critical to validating our claims about the utility of the dashed line as a heuristic for determining whether a particular forecast method is well-matched to a particular task. Of particular interest would be the class of so-called *hybrid systems* [46], which exhibit discrete transitions between different continuous regimes—e.g., a lathe that has an intermittent instability or traffic at an internet router, whose characteristic normal traffic patterns shift radically during an attack. Effective modeling and prediction of these kinds of systems is quite difficult; doing so adaptively and automatically—in the manner that is alluded to at the end of the previous paragraph—would be an even more important challenge.

ACKNOWLEDGMENT

This work was partially supported by NSF grant #CMMI-1245947 and ARO grant #W911NF-12-1-0288. We would also like to acknowledge the Santa Fe Institute whose interdisciplinary halls created a place for this work to be born.

[1] A. Turing, Proceeding of the London Mathematical Society (1936).

[2] A. Weigend and N. Gershenfeld, eds., *Time Series Prediction: Forecasting the Future and Understanding the Past* (Santa Fe Institute, 1993).

[3] J. P. Crutchfield and D. P. Feldman, Chaos: An Interdisciplinary Journal of Nonlinear Science **13**, 25 (2003).

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

[5] C. Bandt and B. Pompe, Phys Rev Lett **88**, 174102 (2002).

[6] E. M. Bollt, T. Stanford, Y.-C. Lai, and K. Życzkowski, Physica D: Nonlinear Phenomena **154**, 259 (2001).

[7] T. Myktowicz, A. Diwan, and E. Bradley, Chaos **19**, 033124 (2009), doi:10.1063/1.3187791.

[8] J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to automata theory, languages, and computation* (Pearson/Addison Wesley, 2007).

[9] U. G. Yule, Philosophical Transactions of the Royal Society of London Series. Series A, Containing papers of a Mathematical or Physical Character **226**, 267 (1927).

[10] L. A. Smith, Physica D: Nonlinear Phenomena **58**, 50 (1992).

[11] M. Casdagli, J. Roy. Stat. Soc. B **54**, 303 (1992).

[12] C. E. Shannon, Bell Systems Technical Journal **30**, 50 (1951).

[13] R. N. Mantegna, S. V. Buldyrev, A. L. Goldberger, S. Havlin, C. K. Peng, M. Simons, and H. E. Stanley, Physical review letters **73**, 3169 (1994).

[14] M. Eisele, Journal of Physics A: Mathematical and General **32**, 1533 (1999).

[15] P. J. Mucci, S. Browne, C. Deane, and G. Ho, in *In Proceedings of the Department of Defense HPCMP Users Group Conference* (1999) pp. 7–10.

[16] Z. Alexander, T. Mytkowicz, A. Diwan, and E. Bradley, in *Advances in Intelligent Data Analysis IX*, Vol. 6065, edited by N. Adams, M. Berthold, and P. Cohen (Springer Lecture Notes in Computer Science, 2010).

[17] T. Mytkowicz, *Supporting experiments in computer systems research*, Ph.D. thesis, University of Colorado (November 2010).

[18] J. L. Henning, SIGARCH Comput. Archit. News **34**, 1 (2006).

[19] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999).

[20] R. Meese and K. Rogoff, Journal of International Economics **14**, 3 (1983).

[21] J. Mućk and P. Skrzypczyński, *Can we beat the random walk in forecasting CEE exchange rates?*, National Bank of Poland Working Papers 127 (National Bank of Poland, Economic Institute, 2012).

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

[23] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting, second edition* (Springer-Verlag, New York, 2002).

[24] R. J. Hyndman and Y. Khandakar, Journal of Statistical Software **27**, 1 (2008).

[25] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin, Journal of Econometrics **54**, 159 (1992).

[26] F. Canova and B. E. Hansen, Journal of Business & Economic Statistics **13**, 237 (1995).

[27] H. Akaike, IEEE Transactions on Automatic Control **19**, 716 (1974).

[28] N. Packard, J. Crutchfield, J. Farmer, and R. Shaw, Physical Review Letters **45**, 712 (1980).

[29] T. Sauer, J. Yorke, and M. Casdagli, Journal of Statistical Physics **65**, 579 (1991).

[30] F. Takens, in *Dynamical Systems and Turbulence, Warwick 1980*, Lecture Notes in Mathematics, Vol. 898, edited by D. Rand and L.-S. Young (Springer Berlin / Heidelberg, 1981) pp. 366–381.

[31] A. M. Fraser and H. L. Swinney, Physical Review A **33**, 1134 (1986).

[32] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, Physical Review A **45**, 3403 (1992).

[33] M. Casdagli and S. Eubank, eds., *Nonlinear Modeling and Forecasting* (Addison Wesley, 1992).

[34] A. Pikovsky, Sov. J. Commun. Technol. Electron. **31**, 911 (1986).
[35] G. Sugihara and R. M. May, Nature **344**, 734 (1990).
[36] E. N. Lorenz, Journal of the Atmospheric Sciences **26**, 636 (1969).
[37] R. J. Hyndman and A. B. Koehler, International Journal of Forecasting , 679 (2006).
[38] Y. B. Pesin, Russian Mathematical Surveys **32**, 55 (1977).
[39] K. Petersen and K. Petersen, *Ergodic Theory*, Cambridge Studies in Advanced Mathematics (Cambridge University Press, 1989).
[40] J. Amigó, *Permutation Complexity in Dynamical Systems: Ordinal Patterns, Permutation Entropy and All That* (Springer, 2012).
[41] B. Fadlallah, B. Chen, A. Keil, and J. Príncipe, Physical Review E **87**, 022911 (2013).
[42] K. Haven, A. Majda, and R. Abramov, Journal of Computational Physics **206**, 334 (2005).
[43] L. Cao, A. Mees, and K. Judd, Physica D **121**, 75 (1998).
[44] E. Deyle and G. Sugihara, PLoS ONE **6**, e18295 (2011).
[45] Y. Cao, W. Tung, J. Gao, V. Protopopescu, and L. Hively, Phys Rev E Stat Nonlin Soft Matter Phys **70**, 046217 (2004).
[46] R. G. R. Sanfelice and A. Teel, IEEE Control Systems Magazine **29**, 28 (2009).