Determinism, Complexity, and Predictability in Computer Performance
Quantifying structured complexity and its role in predictability: With applications to predicting computer performance.
Quantifying Predictability through Structural Complexity
Quantifying Predictability using Structural Complexity
Decoding Predictability in Complicated Real-Valued Time Series: The balance between complexity and temporal structure
Decoding Complicated Time Series: The balance between complexity and temporal structure and the implications for predictability

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

February 27, 2014

Abstract

[[Joshua: Needs to be rewritten to be more general and use computers as the application not the focus]] Computers are deterministic dynamical systems [1]. Among other things, that implies that one should be able to use deterministic forecast rules to predict aspects of their

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

behavior. That statement is sometimes—but not always—true. The memory and processor loads of some simple programs are easy to predict, for example, but those of more-complex programs like `403.gcc` are not. The goal of this paper is to determine why that is the case. We conjecture that, in practice, complexity can effectively overwhelm the predictive power of deterministic forecast models. To explore that, we build models of a number of performance traces from different programs running on different Intel-based computers. We then calculate the *permutation entropy*—a temporal entropy metric that uses ordinal analysis—of those traces and correlate those values against the prediction success.

1 Introduction

BRAINSTORMING

- Complexity need not be hard to predict (can point at the simple predictions paper) [[move to introduction]]
- random walk for example is best predicted by guess what just happend[[move to introduction]]
- The kind of complexity present matters, i.e., that is whether the complexity is structured or not.[[use here and mention in intro]]
- `col_major` brings about the point nicely that some prediction strategies cannot utilize the processes internal information transfer method. That is a nonlinear internal information transfer system cannot be predicted effectively with a linear strategy. This gives a practitioner leverage on when to give up and when to keep working. [[use in this section as bridge to next section]]

Things to add to introduction

1. Different kinds of complexity exist in time series and this makes choosing prediction models difficult

   NOTE: RW and chaos are both complex. One is predictable and one is not.

2. Make an argument that Computer Performance is a great testing ground as it omits signals that completely cover the spectrum of complexity `col_major` ... `403.gcc`
3. When deterministic structure even complex structure exists that structure can be utilized for prediction.
4. For noisy real-valued time series distinguishing randomness (WN,RW) complexity from structured nonlinear / chaotic /high period / high dimensional etc complexity is (until now) very hard.

   for this provide predictions of `403.gcc` and `col_major` side by side and discuss "How can we tell if we did a bad job because the method is inadequate vs the signal being too complex. Lead this into is it possible to tell if there exists structure in a time series to know if we should find a better model or not. Maybe even having 4 predictions. top being ARIMA of the above signals and bottom being LMA of the above signals. Show that one improved and one did not. Is it that we used the wrong method to predict or is it that we simply can't predict the signal better than a random walk due to high levels of internal signal complexity.

5. Introduce the two main contributions of the paper which are outlined at the begining of the results section

NOTES END HERE*

****SECTION START****

Complicated time series are ubiquitous in modern scientific research. Every observed time series, complicated or not, exists on a spectrum of *complexity*[1]. On the low end of that spectrum are time series that exhibit perfect predictive structure, i.e, signals whose future value are perfectly predicted by knowledge of past values from some "ideal" model. In particular, there is a an underlying process that generates and transmits information from the past to the future in a perfectly predictable fashion. Some examples of this are constant or low-period signals. On the opposite end of this spectrum are signals which are "fully complex", i.e., the underlying generating process creates information so rapidly that no information at all is transmitted from the past to the future. Some examples, of this are white noise or random walk processes. With signals on this side of the spectrum, knowledge of the past gives absolutely no insight into the future, *regardless* of the chosen model. In the middle of this spectrum are where complicated *and* forecast-able signals exist, e.g., deterministic chaotic trajectories, high period signals with some level of noise on top. With time series in this portion of the spectrum, enough information is being transmitted from the past to the future that given the *ideal* model the future of the observed system could be forecast with high accuracy.

Unfortunately, as a corollary of the undecidability of the halting problem: there cannot exist *one* universally ideal forecasting scheme for even completely noise-free deterministic time series[**?**], let alone an arbitrary time series. This naturally leads to an important and hard question: given a complicated real-valued time series, is it possible to reliably quantify where on this spectrum the time series lies? That is, when little or nothing is known about the underlying system's generating process (e.g., linear, nonlinear, deterministic, stochastic, stationary, non-stationary etc.) how can we infer from a noisy real-valued time series if the signal is too *complex* to forecast?

To answer this question and for the purposes of this paper we focus on real-valued (possibly) noisy scalar time series which appear *complicated*, i.e., they exhibit interesting behavior, for instance, not of low period, strictly monotonic, or constant. We make no assumptions about the generating process's properties,(e.g., linear, nonlinear, deterministic, stochastic, stationary, non-stationary, etc.) and we attempt to measure empirically where on the previously mentioned spectrum of *complexity* the time series exists. We argue that *permutation entropy* [2], a method for measuring the entropy of a real-valued-finite-length noisy time series through ordinal analysis, is an effective way to explore that conjecture. For the purposes of this paper we define complexity as *permutation entropy*, which we define and explain rigorously in Section 7. To validate these claims we model the time series using a variety of forecasting models which we introduce in Section 4. We then compare the empirically estimated *complexity* with the accuracy of each forecast method. Which results in two primary findings:

1. The complexity of a noisy real-valued time series is quantifiable.
2. The way information is processed internally by a given process plays a crucial role in the success of different forecasting schema.

[1] An approx of (blah) entropy which we make more rigorous in blah

[[Move this up]] To explore this conjecture we require a test bed for generating a broad array of time series to analyze which covers the full spectrum of complexity. We argue that the ideal test bed for this purpose is computer performance. Computers are among the most complex engineered artifacts in current use. Modern microprocessor chips contain multiple processing units and multi-layer memories, for instance, and they use complicated hardware/software strategies to move data and threads of computation across those resources. These features—along with all the others that go into the design of these chips—make the patterns of their processor loads highly complicated. Accurate forecasts of these quantities, if one could construct them, could be used to improve computer design. Computer performance traces are, however, very complicated and range across the entire spectrum of complexity, making this an ideal test bed. Even a simple "microkernel," like a three-line loop that repeatedly initializes a matrix in column-major order, can produce everything from periodic to *chaotic* performance traces [1], depending on the architecture running the software. Such a performance trace is shown in Figure 1. With a single system producing complicated time

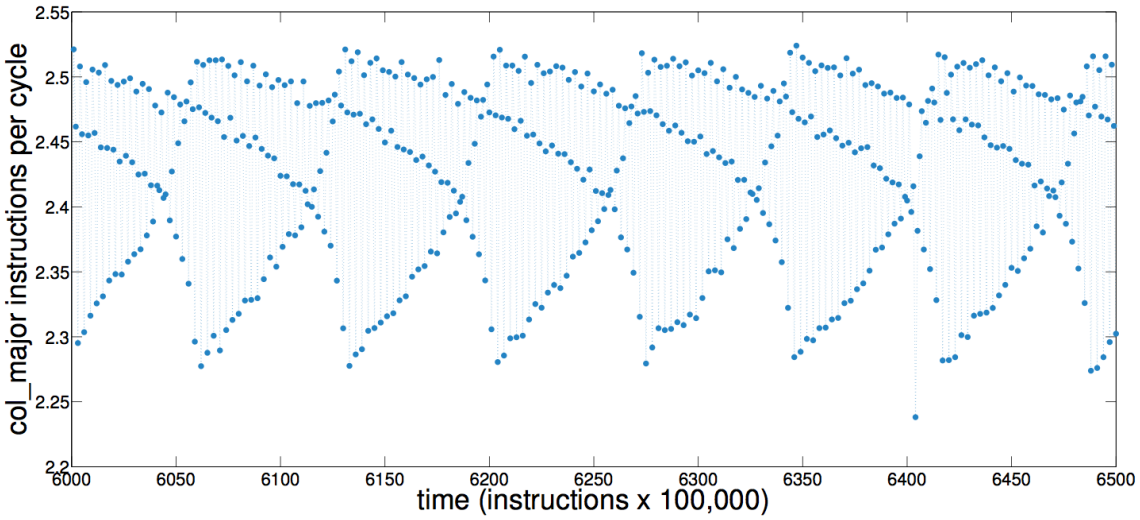


Figure 1: A small snippet of the instructions per cycle(ipc) of `col_major` , a three-line C program that repeatedly initializes a matrix in column-major order, running on an Intel i7®-based machine. Even this simple program exhibits chaotic performance dynamics.

series that vary so greatly in complexity, it would be invaluable to known a priori if a trace contained enough predictive structure to forecast before time and effort was spent on trying to model and forecast it.

The computer systems community has applied a variety of prediction strategies to traces like this, most of which employ regression. An appealing alternative builds on the recently established fact that computers can be effectively modeled as deterministic nonlinear dynamical systems [1]. This result implies the existence of a deterministic forecast rule for those dynamics. In particular, one can use *delay-coordinate embedding* to reconstruct the underlying dynamics of computer performance, then use the resulting model to forecast the future values of computer performance metrics such as memory or processor loads [3]. In the case of simple microkernels like the one that produced the trace in Figure 1, this deterministic modeling and forecast strategy works very well. In more-

complicated programs, however, such as numerical software or compilers, this forecast strategy—as well as the traditional methods—break down some of the time, but work fine others.

This paper is a first step in understanding when, why, and how deterministic forecast strategies fail when they are applied to deterministic systems. We focus here on the specific example of computer performance but believe the results apply to a much broader class of time series. We conjecture that the complexity of traces from these systems—which results from the inherent dimension, non-linearity, and non-stationarity of the dynamics, as well as from measurement issues like noise, aggregation, and finite data length—can make those deterministic signals *effectively* unpredictable. We argue that *permutation entropy* [2], a method for measuring the entropy of a real-valued-finite-length time series through ordinal analysis, is an effective way to explore that conjecture. We study three examples—a simple microkernel and two complex programs: one from the SPEC 2006CPU benchmark suite, and one from LAPACK—running on an Intel i7-based machine. For each program, we calculate the permutation entropy of the processor load (instructions per cycle), then compare that to the prediction accuracy attainable for that trace using a series of deterministic models.

It is worth taking a moment to consider the theoretical possibility of this task. We are not attempting to predict the state of the CPU at an arbitrary point in the future — this, at least with perfect accuracy, would be tantamount to solving the halting problem. What we are attempting is to predict aspects or functions of the running of the CPU: instructions executed per second, cache misses per 100,000 instructions, and similar statistics. Prediction of these quantities at some finite time in the future, even with perfect accuracy, does not violate the Rice-Shapiro theorem.

The rest of the paper is organized as follows. Section 5 describes the experimental setup, as well as the nonlinear modeling and forecast strategies. In Section 7, we review permutation entropy, calculate its value for a number of different computer performance traces, and compare the results to the prediction accuracy. In Section 9, we discuss these results and their implications in regard to our conjecture, and consider future areas of research.

2 Related Work

Attempting to model and characterize predictability is a very old problem which arguably began with Yule in 1927 when he invented AR and many attempts to quantify predicatbility have followed including ....

Entropy though Generating partitions works **if** you have generating parition but not if you don't

Redundancy (details in SFI forecasting), relies heavily on either estimating the generating partiion which is hard to do *and* estimating the positive lyap spectrum which is hard to do for noisy systems and impossible for systems that are not deterministic.

DVS plots, gives pros and cons in (SFI prediction book)

predicting local predictive capacity (radial basis functions stuff, trying to predict error bounds on next forecast based on ensemble uncertainty) but does not aggregate tell you at what level the time series exhibits complexity only locally predictive structure, this actualy gets at the interesting point that different regions of a time series may exhibit differnt levels of complexity which we will illustrate with `dgesdd`

Distrubtion of error. For many methods, if your error is not normally distributed this signals that there is stil more predictve structure to that could be used by for example a larger order ARMA process. But if error is normally distributed, this suggests that you have used up all the predictive structure that **that** model can use, this doesn't quantify if preidctuce structure exists that isn't being used by this process. For example, nonlinear structure which is ignored by a linear predictor.

But this method is differnt becasue it uses no assumption about the underlying model, does not require generating partitions, is applicable to noisy real-valued time series.

Maybe talk about " wpe has been applied to predicting irregularities in brain wave data but no on one has examined it's correlation with predictive structure. " but kind of puts the cart before the horse

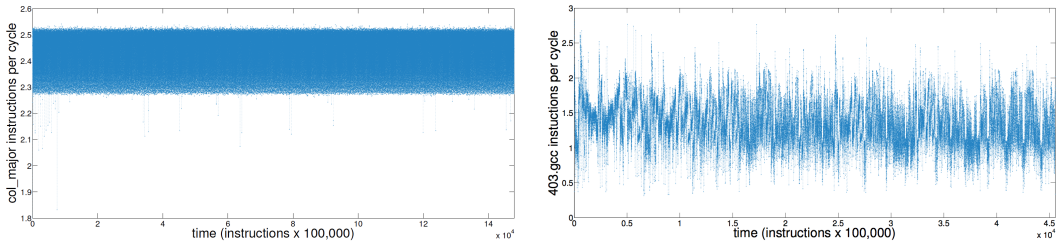
3 Experimental Methods

3.1 Time Series Collection

Experimental methods (how we collect the time series and what the times series are This should be HPM PAPI, which programs we model

3.2 The Programs

Include traces of full `col_major` `dgesdd` and `403.gcc`



(a) `col_major` Time Series (b) `403.gcc` Time Series

Figure 2: In (a) the instructions executed per CPU clock cycle (IPC) during the execution of `col_major` . Each point is the average IPC during a 100,000 instruction period. Similarly in (b) is a time series of the IPC during the execution of `403.gcc` .

¡¡¡¡¡¡¡ HEAD

4 Modeling

======= ¡¡¡¡¡¡¡ HEAD

5 Modeling

6 Modeling

¿¿¿¿¿¿¿ aa0da17e55aece4b9f08f9eec10d8d7e6e3c29f0 Section outline:

1. Description of DCE and parameter estimation
2. Description of auto ARIMA

   (this should be limited and explain it is meant to be out of the box) point at the paper for auto arima for more details.
3. Description of the two naive methods (random walk and mean), make sure to explain that these methods are naive and simple but not necessarily bad.
4. ✓Add a section talking about evaluation methods i.e., MASE, this text is currently written and just sitting at the beginning of the results.

6.1 Lorenz Method of Analogues (LMA)

TODO:

1. update examples
2. remove section on RMSE
3. update figures

6.1.1 Reconstructing hidden dynamics

Delay-coordinate embedding allows one to reconstruct a system's full state-space dynamics from a *single* scalar time-series measurement—provided that some conditions hold regarding that data. Specifically, if the underlying dynamics and the measurement function—the mapping from the unknown state vector $\vec{X}$ to the scalar value x that one is measuring—are both smooth and generic, Takens [4] formally proves that the delay-coordinate map

$$F(\tau, m)(x) = ([x(t)\ x(t+\tau)\ \ldots\ x(t+m\tau)])$$

from a d-dimensional smooth compact manifold M to Re^{2d+1}, where t is time, 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 build a formal model of the full system dynamics without measuring (or even knowing) every one of its state variables. This is the foundation of the modeling approach that is used in this paper. The first step in the process is to estimate values for the two free parameters in the delay-coordinate map: the delay τ and the dimension m. We follow standard procedures for this, choosing the first minimum in the average mutual information as an estimate of τ [5] and using the false-near(est) neighbor

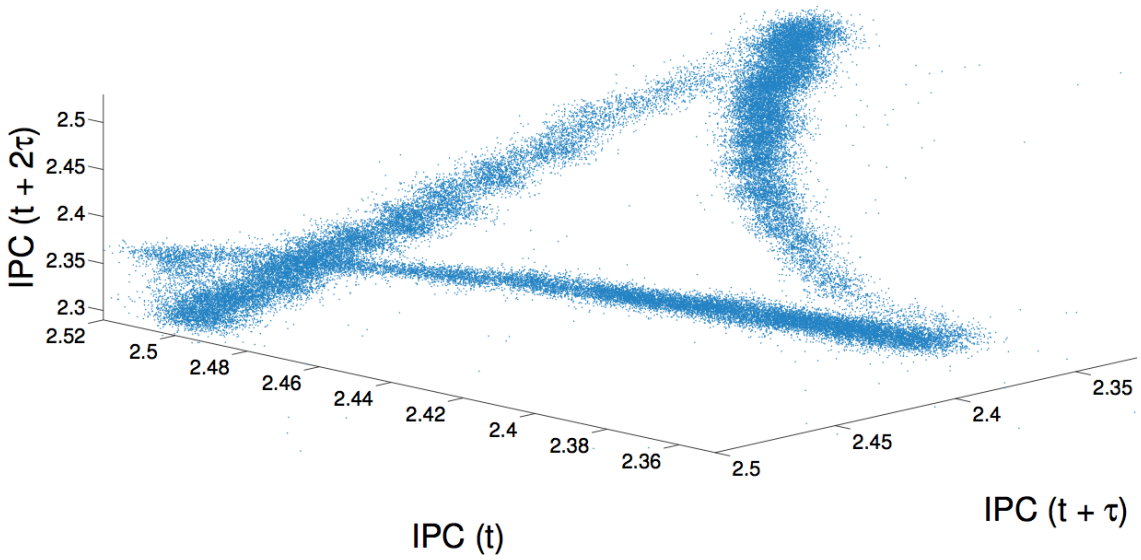


Figure 3: [[Joshua:Perhaps add a similar graphic of `403.gcc` to show contrast]]A 3D projection of a delay-coordinate embedding of the trace from Figure 1 with a delay (τ) of 100,000 instructions.

method of [6], with a threshold of 10%, to estimate m. A plot of the data from Figure 1, embedded following this procedure, is shown in Figure 3.

The coordinates of each point on this plot are differently delayed elements of the `col_major` L2 cache miss rate time series $y(t)$: that is, $y(t)$ on the first axis, $y(t+\tau)$ on the second, $y(t+2\tau)$ on the third, and so on. Structure in these kinds of plots—clearly visible in Figure 3—is an indication of determinism[2]. That structure can also be used to build a forecast model.

6.1.2 LMA: Using dynamics in forecasting

Given a nonlinear model of a deterministic dynamical system in the form of a delay-coordinate embedding like Figure 3, one can build deterministic forecast algorithms by capturing and exploiting the geometry of the embedding. Many techniques have been developed by the dynamical systems community for this purpose (e.g., [7, 8]). Perhaps the most straightforward is the "Lorenz method of analogues" (LMA), which is essentially nearest-neighbor prediction in the embedded state space [9]. 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 forecast—works quite well on the trace in Figure 1, as shown in Figure **??**. On the other hand, if we use the same approach to forecast the processor load[3] of the `482.sphinx3` program from the SPEC cpu2006 benchmark suite, running on an Intel i7®-based machine, the prediction is far less accurate; see Figure **??**.

This begs the question: If these traces all come from deterministic systems—computers—then

[2] A deeper analysis of Figure 3—as alluded to on the previous page—supports that diagnosis, confirming the presence of a chaotic attractor in these cache-miss dynamics, with largest Lyapunov exponent $\lambda_1 = 8000 \pm 200$ instructions, embedded in a 12-dimensional reconstruction space [1].

[3] Instructions per cycle, or IPC

why are they not equally predictable? Our conjecture is that the sheer complexity of the dynamics of the SPEC benchmarks running on the Intel i7® machine make them effectively impossible to predict.

6.2 Autoregressive-integrated-moving average(ARIMA)

6.3 Naive: Random Walk and naïve

6.4 Prediction Accuracy: Mean Absolute Scaled Error (MASE)

In order to analyze correctness of each prediction we split each time series into two pieces: the first 90% referred to as the "learning" or "training" signal, $\{X_{i,obs}\}_{i=1}^{n}$ and the last 10% known as the "test" or "correct" signal $\{c_j\}_{j=n+1}^{k+n+1}$. The learning signal is used to train an initial model (e.g., LMA or ARIMA) as described in Section 5. The test signal is used both to assess the models forecasting accuracy and for any refitting that may be necessary. In particular, we perform k 1-step predictions, after each 1-step prediction[4] we append the training signal with the next point in the correct signal c_j, refit the model taking into account the new system measurement and perform another prediction. This is repeated k times to obtain $\{p_j\}_{j=n+1}^{k+n+1}$.

As a figure of merit we calculate the Mean Absolute Squared Error (MASE)[10] between the true and predicted signals:

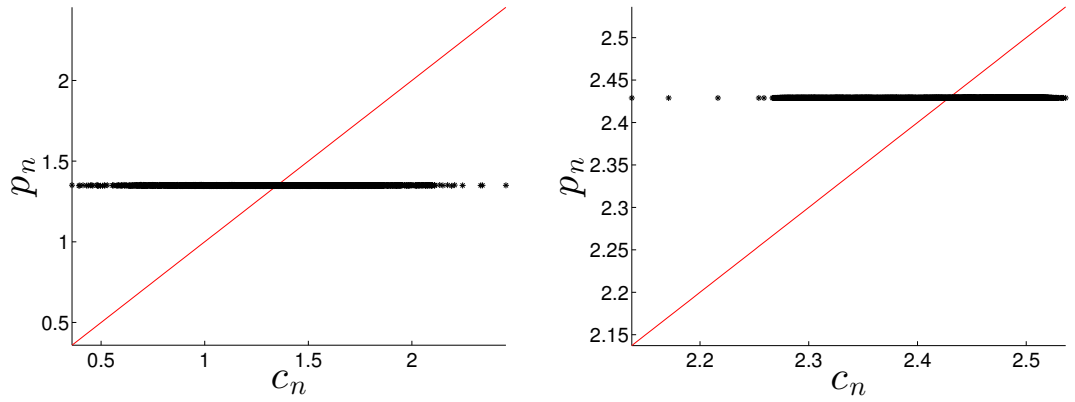
$$MASE = \sum_{j=n+1}^{k+n+1} \frac{|c_j - p_j|}{\frac{k}{n-1}\sum_{i=2}^{n}|X_{i,obs} - X_{i-1,obs}|}$$

The scaling term for MASE:

$$\frac{1}{n-1}\sum_{i=2}^{n}|X_{i,obs} - X_{i-1,obs}|$$

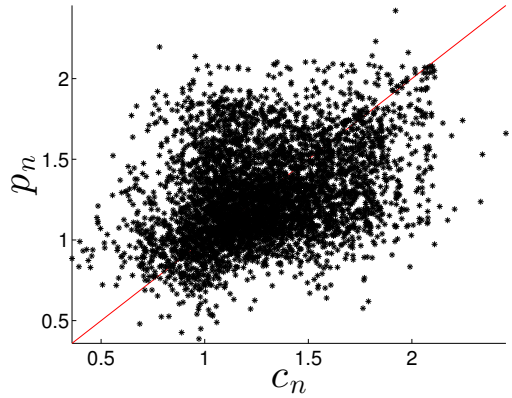
is the average in-sample forecast error for a random walk prediction ($p_i = X_{i-1,obs}$). This error method was introduced in [10] as a "generally applicable measurement of forecast accuracy without the problems seen in the other measurements." The major advantage of MASE is that it allows fair comparison across methods, prediction horizons and varying signal scales. When a forecast results in a $MASE < 1$ this means that the prediction method gave, on average, smaller errors than the 1-step errors from the in-sample random walk forecast strategy. Analogously, $MASE > 1$ means that the prediction method did worse, on average than the 1-step errors for the in-sample random walk forecast strategy. In Table **??** we provide the distribution [[Joshua: Ryan, Is this the right word? we give mean ± std. dev but some have very skewed right tails]] of MASEs for each of the 8 signals and 3 prediction strategies, these are averaged over 15 runs of each type (signal + method). [[and cherry pick a few examples of `403.gcc` and `col_major` to put in the text For comparison Table **??** also has the distribution of weighted permutation entropies for word lengths of $l = 6$.]]

[4]We would like to note that this rebuilding occurs due to a problem with ARIMA models converging to a mean prediction if too long of a prediction horizon is used, this is not a handicap of either LMA or naïve.

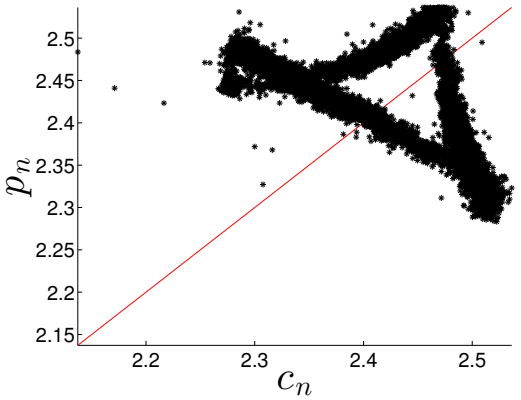


(a) `403.gcc` naïve

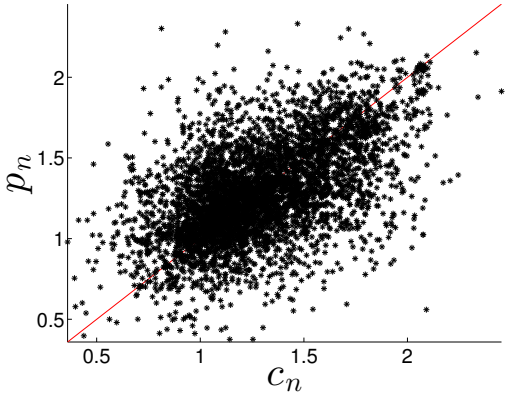
(b) `col_major` naïve



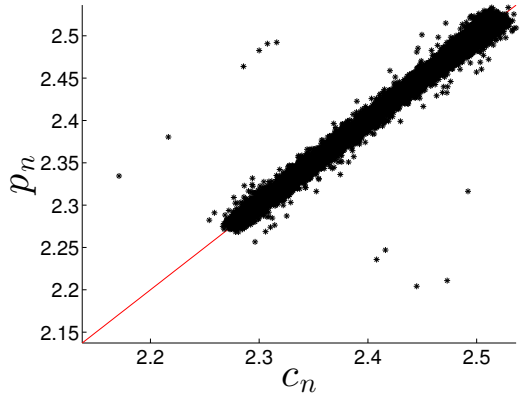
(c) `403.gcc` ARIMA



(d) `col_major` ARIMA



(e) `403.gcc` LMA



(f) `col_major` LMA

Figure 4: For each of these, we plot the predicted value p_n against the correct value c_n. On this type of plot a perfect prediction lies exactly on the diagonal, that is the line $p_n = c_n$, e.g., 4f is a near perfect prediction where-as 4c is a very poor prediction.

7 Measuring Structural Complexity

THINGS TO DO

1. rewrite for consistency in notation. For example, some parts of the paper call word length l and some call it n.
2. make sure it is clear and we justify that detecting and characterizing structural complexity is hard for real-valued noisy time series.
3. Quantifying structured and unstructured complexity is nontrivial in the case of real-valued noisy time series but WPE does this. [[talk about again here but justify in information theory section]]
4. justify and explain the regime split in `dgesdd`

For the purposes of this paper, one can view entropy as a measure of complexity and predictability in a time series. A high-entropy time series is almost completely unpredictable—and conversely. This can be made more rigorous: Pesin's relation [11] states that in chaotic dynamical systems, the Shannon entropy rate 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 will diverge with time: $|\Delta x(t)| \approx e^{\lambda t} |\Delta x(0)|$. The faster the divergence, the more difficult prediction becomes.

Utilizing entropy as a measure of temporal complexity is by no means a new idea [12, 13]. Its effective usage requires categorical data: $x_t \in \mathcal{S}$ for some finite or countably infinite *alphabet* $\mathcal{S}$, whereas data taken from real-world systems is effectively real-valued. To get around this, one must discretize the data— typically achieved by binning. Unfortunately, this is rarely a good solution to the problem, as the binning of the values introduces an additional dynamic on top of the intrinsic dynamics whose entropy is desired. The field of symbolic dynamics studies how to discretize a time series in such a way that the intrinsic behavior is not perverted, but these methods are fragile in the face of noise and require further understanding of the underlying system, which defeats the purpose of measuring the entropy in the first place.

Bandt and Pompe introduced the *permutation entropy* (PE) as a "natural complexity measure for time series" [2]. Permutation entropy employs a method of discretizing real-valued time series that follows the intrinsic behavior of the system under examination. 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 $x_2 \leq x_3 \leq x_1$. This method has many features; among other things, it is generally robust to observational noise and requires no knowledge of the underlying mechanisms.

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

$$p(\pi) = \frac{|\{t | t \leq T - n, \phi(x_{t+1}, \ldots, x_{t+n}) = \pi\}|}{T - n + 1}$$

Where $|\cdot|$ *is set cardinality. The* permutation entropy *of order* $n \geq 2$ *is defined as*

$$H(n) = -\sum_{\pi \in \mathcal{S}_n} p(\pi) \log_2 p(\pi)$$

Notice that $0 \leq H(n) \leq \log_2(n!)$ [2]. With this in mind, it is common in the literature to normalize permutation entropy as follows: $\frac{H(n)}{\log_2(n!)}$. With this convention, "low" entropy is close to 0 and "high" entropy is close to 1. Finally, it should be noted that the permutation entropy has been shown to be identical to the Shannon entropy for many large classes of systems [14].

Here we will be utilizing a variation of the permutation entropy, the *weighted permutation entropy* (WPE) [15]. The weighted permutation entropy attempts to correct for observational noise which is larger than some trends in the data, but smaller than the larger scale features — for example, a signal that switches between two fixed points with noise about those fixed points. The weighted permutation entropy would be dominated by the switching rather than by the stochastic fluctuation. To accomplish this, the *weight* of a permutation is taken into account:

$$w(x_{t+1:t+n}) = \frac{1}{n} \sum_{x_i \in \{x_{t+1:t+n}\}} (x_i - \bar{x}_{t+1:t+n})^2$$

where $x_{t+1:t+n}$ is a sequence of values $x_{t+1}, \ldots, x_{t+n}$, and $\bar{x}_{t+1:t+n}$ is the arithmetic mean of those values.

The weighted probability of a permutation is then:

$$p_w(\pi) = \frac{\sum_{t \leq T-n} w(x_{t+1:t+n}) \cdot \delta(\phi(x_{t:t+n}), \pi)}{\sum_{t \leq T-n} w(x_{t+1:t+n})}$$

where $\delta(x, y)$ is 1 if $x = y$ and 0 otherwise. Effectively, this weighted probability enhances permutations involved in "large" features and demotes permutations which are small in amplitude relative to the features of the time series. The weighted permutation entropy is then:

$$H_w(n) = -\sum_{\pi \in \mathcal{S}_n} p_w(\pi) \log_2 p_w(\pi),$$

which can also be normalized by dividing by $\log_2(n!)$, and will be in all the results of this paper.

In practice, calculating permutation entropy and weighted permutation entropy involves choosing a good value for the word length n. The primary consideration is that the value be large enough that forbidden ordinals are discovered, yet small enough that reasonable statistics over the ordinals are gathered: e.g.:

$$n = \underset{\ell}{\operatorname{argmax}}\{T \gtrapprox 100\ell!\},$$

assuming an average of 100 counts per ordinal is sufficient. In the literature, $3 \leq n \leq 6$ is a standard choice — generally without any formal justification. In theory, the permutation entropy should reach an asymptote with increasing n, but that requires an arbitrarily long time series. In

practice, what one should do is calculate the *persistent* permutation entropy by increasing n until the result converges, but data length issues can intrude before that convergence is reached.

The weighted permutation entropy for the `SVD` program is given in Fig. 5. To generate this image a window of 5,000 values slid over the time series. Within each of those windows, the statistics over words of length 4 are computed and the WPE is calculated. The gray bands denote regions where the 5,000 value window overlapped visually-distinct regimes. It can be seen that the behaviors of the weighted permutation entropy vary between regimes. [[I think here it would be good to add a paragraph explaining the windowed WPE was used for regime choices on SVD...emphasizing that over a time series permutation entropy fluctuates illustrating within a single time series different levels of complexity and predictability exist. Maybe point at some of the predicting predictability papers.]]

Liz commented this image out temporarily so that her mac doesn't hang when she scrolls past p12

Figure 5: [Joshua: I think adding the colored SVD trace to this would be good or putting it above this figure but need to figure how to line them up properly. Also we need to label that the numbers on the bottom of WPE are regimes not instructions...]]The weighted permutation entropy of one run of SVD. The gray bands are regions where the window overlaps regimes. The window size used is 5,000 $\times$ 100,000 instructions and the word length is 4.[[Joshua:why is $l = 4$ and not 5,6 like the rest of the paper?]]For reference the instructions per cycle of `dgesdd` are plotted as a ghost behind this plot. Each color on the ghosted time series corresonds to the different regimes as selected by rapid shifts in WPE. From left to right each change in color represents a change in regime for 6 regimes in total.

8 Predictability, Complexity, and Permutation Entropy

In this section, we offer an empirical validation of the two key conjectures introduced in Section 1, 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 5. 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 6.4, 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 7. 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 6 shows the *best* of these 45 predictions for each system: i.e., the lowest error over all 15 trials and 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.

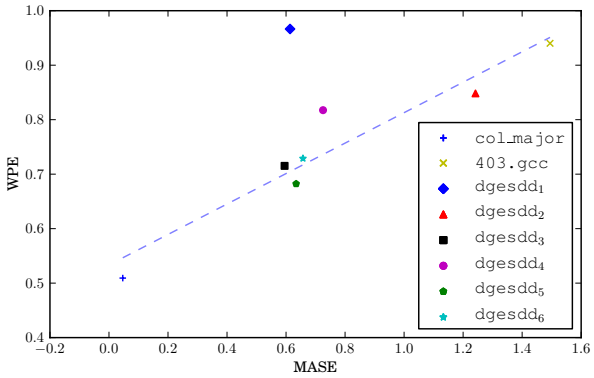


Figure 6: Weighted permutation entropy vs. mean absolute scaled error (MASE) of the best prediction of each system. The dashed line is a least-squares linear fit of all the points except `dgesdd`$_1$, which we have excluded for reasons explained in the text. [[Ryan, is that correct?]]

The best-case prediction error 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[5] captures this rough proportionality. This is not a formal result. The three 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 6, 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 7 shows WPE vs. MASE plots for all 360 experiments. There are 15 points in each cluster, one for each trial. (The points in Figure 6 are the leftmost of the points for the corresponding system in any of these three plots.) 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: clusters of predictions generated with the nonlinear LMA method are closer to the dashed line, whereas the naive and ARIMA prediction clusters are more spread out [[need to revisit this wording when get the third image]]. This is a geometric validation of the second conjecture in this paper, as described

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

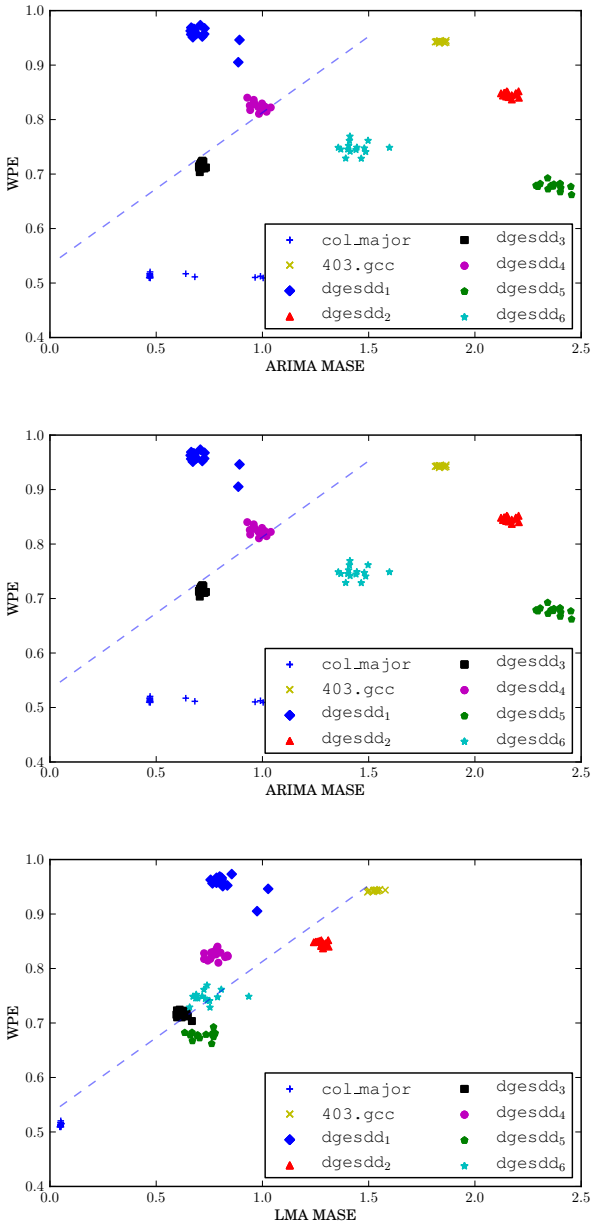


Figure 7: WPE vs. MASE for all trials, methods, and systems. `dgesdd`$_1$, `dgesdd`$_3$, and `dgesdd`$_5$ are omitted from the top plot for scale reasons; their MASE scores are 2.676 ± 4.328, 31.386 ± 0.282, and 20.870 ± 0.192 respectively. Replace the top plot with a plot of the naive results—but running from 0 to 3.1 on the horizontal axis and excluding SVD 1, 3, and 5. The dashed line is the same as in the previous figure [[Ryan, is this correct?]]. Numerical values, including means and standard deviations of the errors, can be found in Table **??** in the appendix.

in the following paragraphs.

Though `col_major` is a very simple program, its dynamics are actually quite complicated, as discussed in Section 1. Recall from the graphical representations in the right-hand column of Figure 4 that the naïve and ARIMA prediction methods do not perform as well on this signal as the nonlinear LMA method. The MASE values of the naïve and ARIMA predictions are $0.57\pm$ 0.001 and 0.59 ± 0.21, respectively, across all 15 trials. That is, the errors produced by these methods are a little over half as large as the errors produced by 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 0.51 ± 0.003, which is in the center of the complexity spectrum described in Section 1. [[This suggests that roughly half of the signal is predictive structure.]] [[I'm still uncomfortable with that phrasing; let's revisit it after we have the terminology and definitions hammered out.]]

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 7, 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, 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 is a heuristic that captures this argument. 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 (viz., the top images in Figure 7), one should try a different method.

Note that the `col_major` points are clustered in LMA but spread out horizontally in ARIMA, far to the right of the dashed line (i.e., much worse than the other methods). [[We need to come up with a hypothesis about this spread and skew, even if it's totally tentative. We may want to merge this discussion into one of the other `col_major` paragraphs—or perhaps put it with the $\texttt{dgesdd}_1$ discussion later on?]]

The WPE of $\texttt{dgesdd}_5$ (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, in the discussion of `col_major`, the effectiveness of any prediction strategy will depend on how well it leverages the available information; methods that use linear models, for instance, may fail when applied to nonlinear processes. The match between model and task is an issue in the $\texttt{dgesdd}_5$ experiments as well. The MASE scores of the naïve and ARIMA predictions for this system were 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[6]. Again, 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

[6]The naïve MASE score is large because the amount of variance in this signal is high, which makes guessing the mean a particularly bad strategy. The same thing is true of the $\texttt{dgesdd}_3$ signal.

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 much higher: 0.940 ± 0.001. This system transmits very little information forward in time and provides almost no structure for prediction methods to work with. [[What we see: naïve does the best, then LMA, then ARIMA. Hypothesis: that that 5% of the structure is nonlinear, so LMA can use it better than ARIMA can. naïve does better than LMA because it's filtering out the noise.]]

`dgesdd`$_1$ behaves very differently than the other seven systems in this study: though its WPE is very high (0.95 ± 0.02), two of the three prediction methods do quite well, yielding MASE scores of 0.82 ± 0.08 (LMA) and 0.714 ± 0.07 (ARIMA). This pushes the corresponding clusters of points in the bottom two images in Figure 7 well above the trend followed by the other seven clusters. Also, the MASE scores of the predictions of this system produced by the naïve method are highly inconsistent: 2.67 ± 4.33. These effects, we believe, are artifacts of 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 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 `dgesdd`$_1$ clusters for ARIMA and LMA are above the dashed line in Figure 7. This time series, which is shown in closeup in Figure 8, is not quite to the level of the

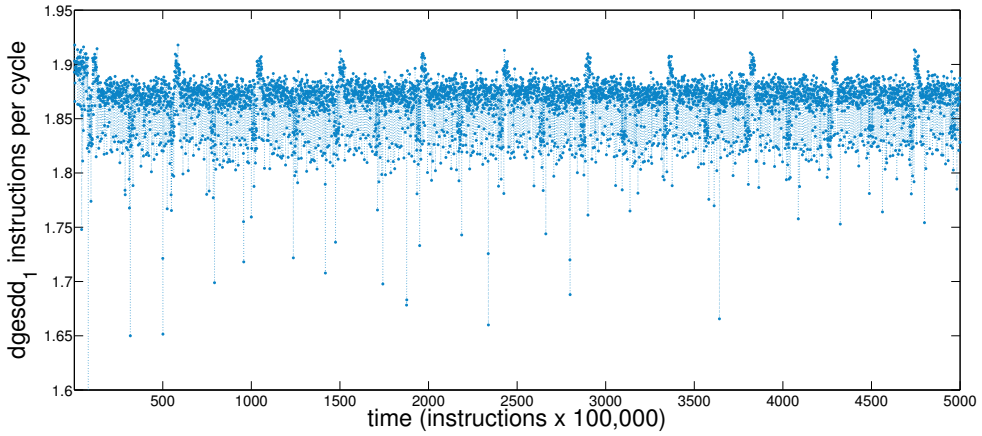


Figure 8: A small portion of the `dgesdd`$_1$ time series

worst-case signal described above, but it still poses a serious challenge to random-walk prediction. It is dominated by a white-noise regime (between $\approx$1.86 and $\approx$1.88 on the vertical scale in the Figure), 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 creates a large spread in the `dgesdd`$_1$ MASE values. For all of these reasons, we view `dgesdd`$_1$ as an outlier and exclude it from the fit calculation of the dashed line in Figure 6.

9 Conclusions & Future Work

1. This need to be rewritten to conclude and address this paper...

Moved the following material here from the results section, since they're more general. But there's a lot of repeated material that will obviously need to get compressed and streamlined.

Just because there is forward information transfer, that does not mean that an arbitrary predictor can interpret or utilize it, but WPE can tell us when this structure is present.

These results support that time series with low to moderate complexity ($0 \leq WPE \leq 0.85$) can be predicted more efficiently than a naïve random walk *and* that complexity can be qualitatively measured for a real-valued noisy time series using WPE. This will allow practitioners to stop spinning their wheels in the case of signals who are simply better predicted with a simple strategy like random walk. The analysis of these results illuminates an interesting point: The way structure, information and complexity are processed by a generating process plays a crucial role in the success of a given prediction scheme.

As we discussed in Section 7 distinguishing structured from unstructured complexity in the case of real-valued time series is non trivial, i.e., distinguishing when a complex signal omits predictable structure and when the signal is effectively random is not a trivial task. As described in Section 1, for a practitioner this can be frustrating because it can be nearly impossible to find the source of faulty predictions: Is it simply that we need to use a more ideal (possibly more advanced) predictor or is it the case that the time series is simply so complex that using a simple (yet inconsistent) forecast strategy such as random walk is the best we can do. Fortunately, weighted permutation entropy (WPE) allows us to make this distinction for noisy real-valued time series.

Needs to be more stuff in the WPE section for this to tie back to as well as in the new related work section. [[Joshua, can you elaborate on this so that Ryan can fill this in?]]

The results presented here suggest that permutation entropy—a ordinal calculation of forward information transfer in a time series—is an effective metric for predictability of computer performance traces. Experimentally, traces with a persistent PE $\gtrapprox 0.97$ have a natural level of complexity that may overshadow the inherent determinism in the system dynamics, whereas traces with PE $\lessapprox 0.7$ seem to be highly predictable (viz., at least an order of magnitude improvement in nRMSPE).Further, the persistent WPE values of 0.5– 0.6 for the `col_major` trace are consistent with dynamical chaos, further corroborating the results of [1].

If information is the limit, then gathering and using more information is an obvious next step. 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. Another alternative is to sample several system variables

simultaneously and build multivariate delay-coordinate embeddings. Existing approaches to that are computationally prohibitive [16]. We are working on alternative methods that sidestep that complexity.

Acknowledgment

This work was partially supported by NSF grant #CMMI-1245947 and ARO grant #W911NF-12-1-0288.

References

[1] T. Myktowicz, A. Diwan, and E. Bradley. Computers are dynamical systems. *Chaos*, 19:033124, 2009. doi:10.1063/1.3187791.

[2] C. Bandt and B. Pompe. Permutation entropy: A natural complexity measure for time series. *Phys Rev Lett*, 88(17):174102, 2002.

[3] J. Garland and E. Bradley. Predicting computer performance dynamics. In *Proceedings of the 10th International Conference on Advances in Intelligent Data Analysis X*, pages 173–184, Porto, Portugal, 2011.

[4] 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.

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

[6] 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.

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

[8] A. Weigend and N. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Santa Fe Institute, 1993.

[9] E. N. Lorenz. Atmospheric predictability as revealed by naturally occurring analogues. *Journal of the Atmospheric Sciences*, 26:636–646, 1969.

[10] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pages 679–688, 2006.

[11] Ya B Pesin. Characteristic Lyapunov exponents and smooth ergodic theory. *Russian Mathematical Surveys*, 32(4):55, 1977.

[12] C. E. Shannon. Prediction and entropy of printed English. *Bell Systems Technical Journal*, 30:50–64, 1951.

[13] RN Mantegna, SV Buldyrev, AL Goldberger, S. Havlin, CK Peng, M. Simons, and HE Stanley. Linguistic features of noncoding DNA sequences. *Physical review letters*, 73(23):3169–3172, 1994.

[14] J. Amigó. *Permutation Complexity in Dynamical Systems: Ordinal Patterns, Permutation Entropy and All That*. Springer, 2012.

[15] Bilal Fadlallah, Badong Chen, Andreas Keil, and José Príncipe. Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information. *Physical Review E*, 87(2):022911, 2013.

[16] Liangyue Cao, Alistair Mees, and Kevin Judd. Dynamics from multivariate time series. *Physica D*, 121:75–88, 1998.