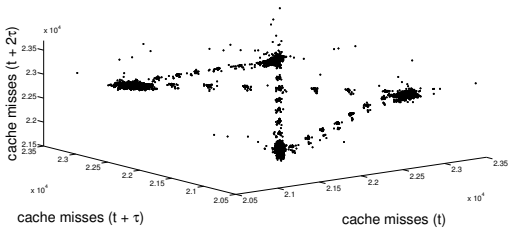



FIGURE 1. A 3D projection of a delay-coordinate embedding of the trace from Figure **??** with a delay ($\tau$) of 100,000 instructions.

## 1. Unused Figures

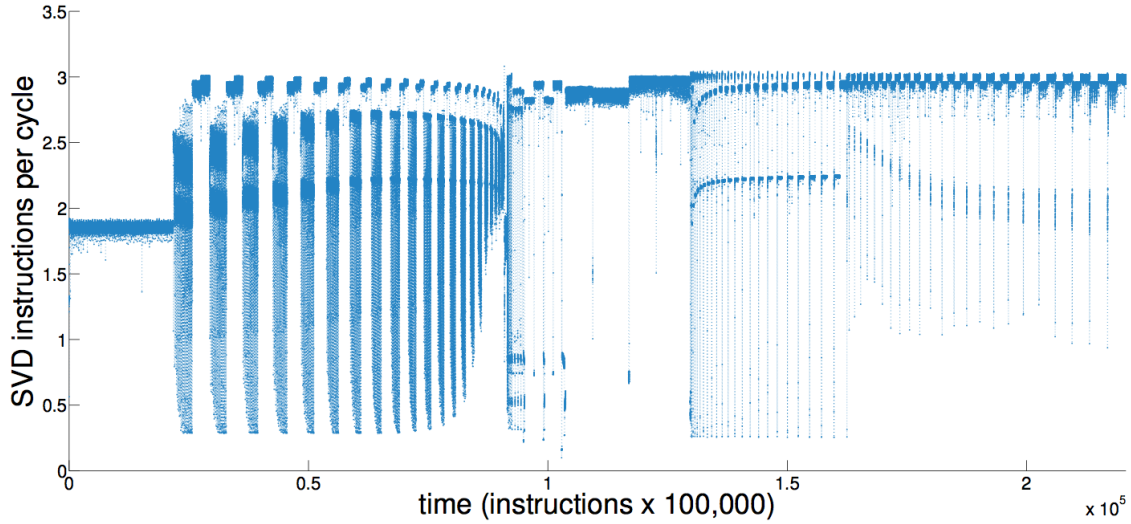
## 2. Unused Tables

TABLE 1. Average nRMSE over 15 runs for each signal and average wpe at word length 5 and 6 for each signal.

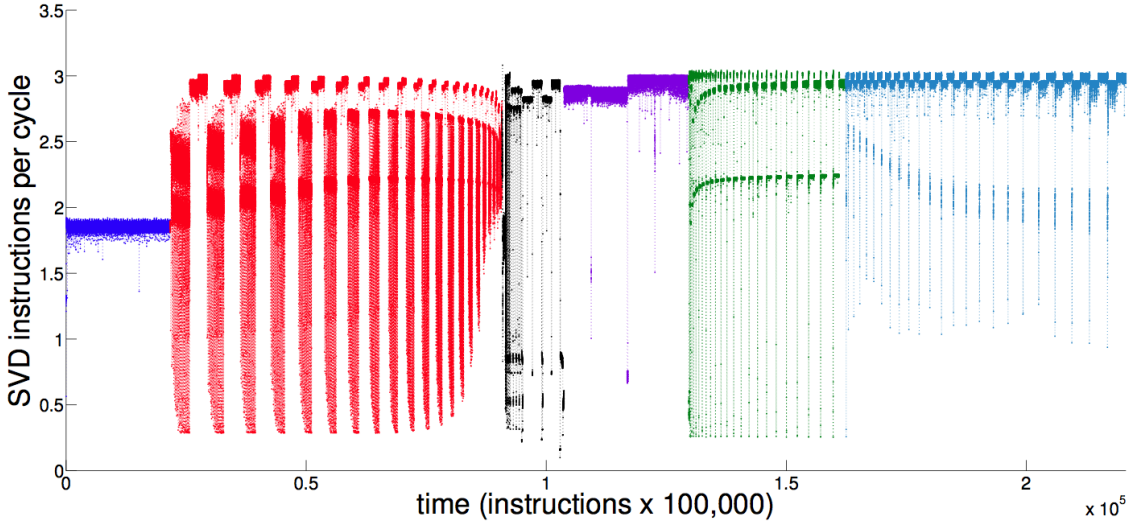
| | nRMSE LMA | nRMSE naïve | $l = 5$ | $l = 6$ |
|---|---|---|---|---|
| gcc | $0.1407 \pm 0.0063$ | $0.1487 \pm 0.0066$ | $0.9510 \pm 0.0011$ | $0.9430 \pm 0.0013$ |
| col_major | $0.0252 \pm 0.0061$ | $0.1975 \pm 0.0436$ | $0.5636 \pm 0.0031$ | $0.5131 \pm 0.0034$ |
| SVD_IPC_Regime1 | $0.1680 \pm 0.0317$ | $0.3517 \pm 0.5223$ | $0.9761 \pm 0.0084$ | $0.9572 \pm 0.0156$ |
| SVD_IPC_Regime2 | $0.1716 \pm 0.0043$ | $0.1762 \pm 0.0012$ | $0.8760 \pm 0.0052$ | $0.8464 \pm 0.0044$ |
| SVD_IPC_Regime3 | $0.0507 \pm 0.0011$ | $0.5413 \pm 0.0005$ | $0.7768 \pm 0.0073$ | $0.7157 \pm 0.0056$ |
| SVD_IPC_Regime4 | $0.1288 \pm 0.0471$ | $0.2308 \pm 0.0867$ | $0.9073 \pm 0.0080$ | $0.8246 \pm 0.0077$ |
| SVD_IPC_Regime5 | $0.0235 \pm 0.0022$ | $0.1306 \pm 0.0003$ | $0.7333 \pm 0.0076$ | $0.6776 \pm 0.0068$ |
| SVD_IPC_Regime6 | $0.0196 \pm 0.0022$ | $0.0508 \pm 0.0003$ | $0.8101 \pm 0.0135$ | $0.7475 \pm 0.0106$ |

TABLE 2. Average nRMSE over 15 runs for each signal and average wpe at word length 5 and 6 for each signal.

| | Horizon | nRMSE LMA | nRMSE naïve | LMA MASE (ARIMA) | $l = 5$ |
|---|---|---|---|---|---|
| gcc | 4,542 | $0.1407 \pm 0.0063$ | $0.1487 \pm 0.0066$ | 1.5416(0.929) | $0.9510 \pm 0.00$ |
| col_major | 14,709 | $0.0252 \pm 0.0061$ | $0.1975 \pm 0.0436$ | 0.0519(0.07UF/0.5740/0.9936) | $0.5636 \pm 0.00$ |
| SVD_Full | 22,072 | $0.0323 \pm 0.0019$ | $0.1784 \pm 0.0040$ | $0.8765 \pm 0.0044$ | $0.8508 \pm 0.00$ |
| SVD_IPC_Regime1 | 2,158 | $0.1680 \pm 0.0317$ | $0.3517 \pm 0.5223$ | 0.7954(0.724UF) | $0.9761 \pm 0.00$ |
| SVD_IPC_Regime2 | 6,902 | $0.1716 \pm 0.0043$ | $0.1762 \pm 0.0012$ | 1.9990(1.138) | $0.8760 \pm 0.00$ |
| SVD_IPC_Regime3 | 947 | $0.0507 \pm 0.0011$ | $0.5413 \pm 0.0005$ | 1.0089 (1.165215) | $0.7768 \pm 0.00$ |
| SVD_IPC_Regime4 | 2,929 | $0.1288 \pm 0.0471$ | $0.2308 \pm 0.0867$ | 0.8976 (0.840548) | $0.9073 \pm 0.00$ |
| SVD_IPC_Regime5 | 3,255 | $0.0235 \pm 0.0022$ | $0.1306 \pm 0.0003$ | 0.7487 (1.260/2.3446) | $0.7333 \pm 0.00$ |
| SVD_IPC_Regime6 | 5,804 | $0.0196 \pm 0.0022$ | $0.0508 \pm 0.0003$ | 0.7848 (0.7831) | $0.8101 \pm 0.01$ |



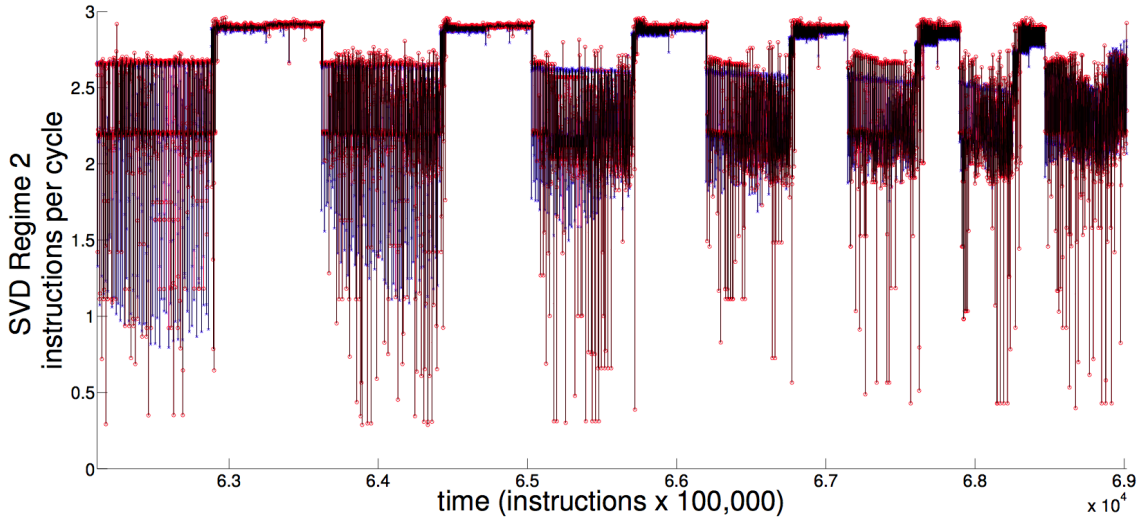

(A) SVD IPC without coloring



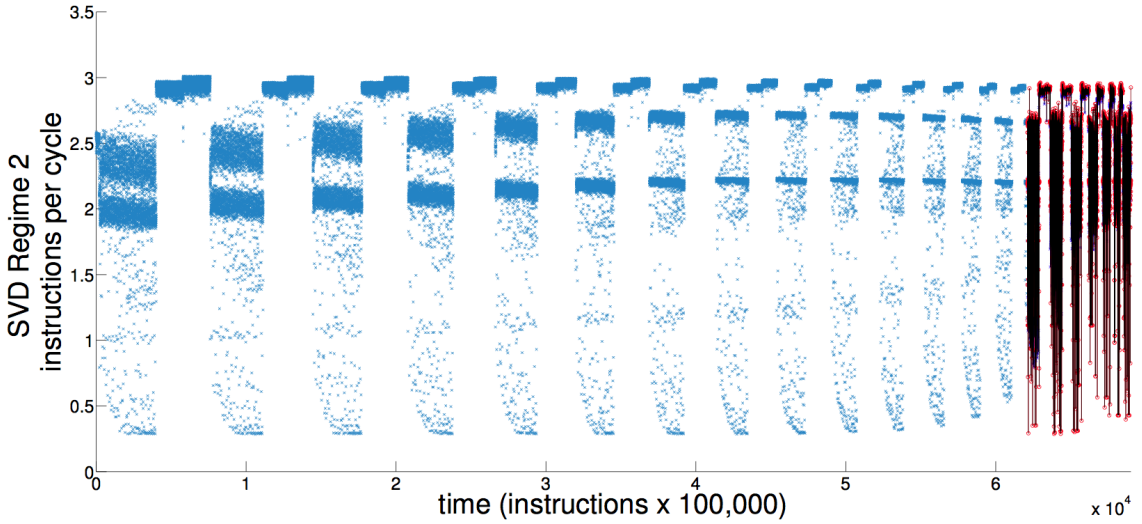

(B) SVD IPC with coloring

FIGURE 2. SVD Full Time Series

Table **??** shows the permutation entropy results for the examples considered in this paper, with the nRMSPE prediction accuracies from the previous section included alongside for easy comparison.




(A) SVD Regime 2 LMA Prediction



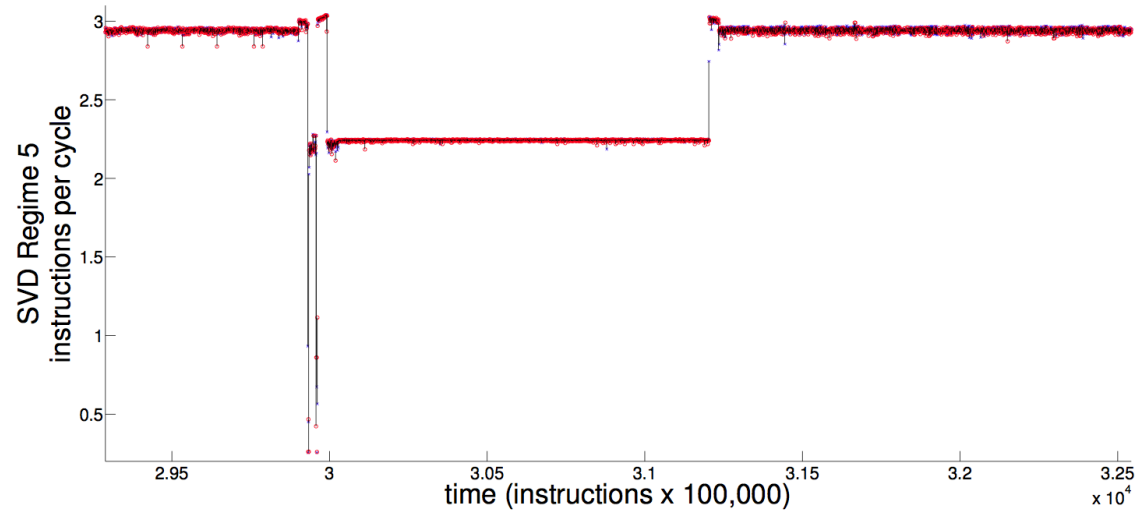

(B) SVD Regime 2 LMA Prediction with Full Time Series

FIGURE 3. SVD Regime 2 LMA Prediction Figures

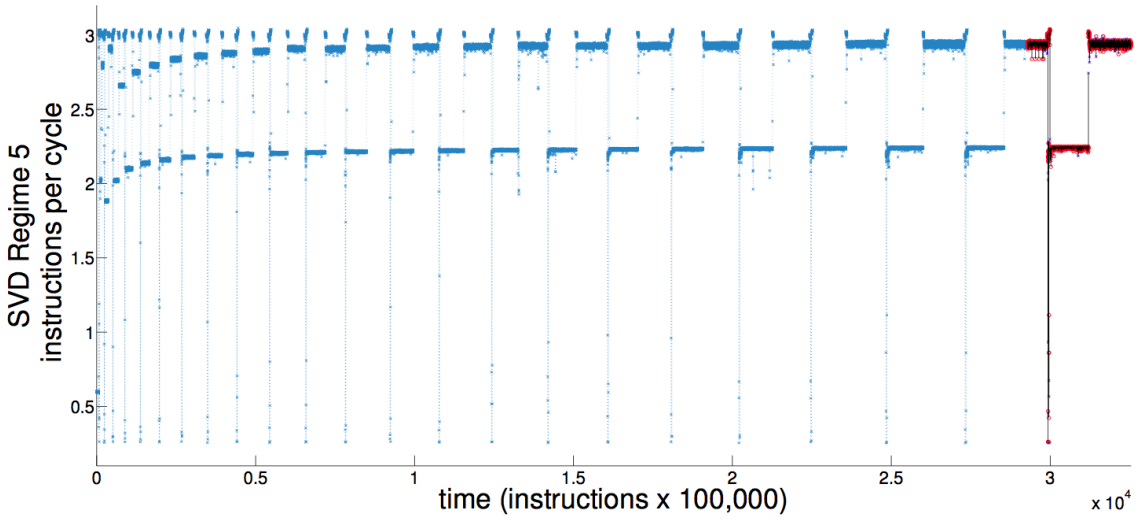
## 3. Unused Text

## 4. Introduction: What is Permutation Entropy

When analyzing a time series there are several standard characterizations of complexity, including (but not limited to) entropy, Lyapunov exponents and fractal dimensions. Rigorous definitions of all these quantities exist if the observation is a *noise-free* trajectory of




(A) SVD Regime 5 LMA Prediction

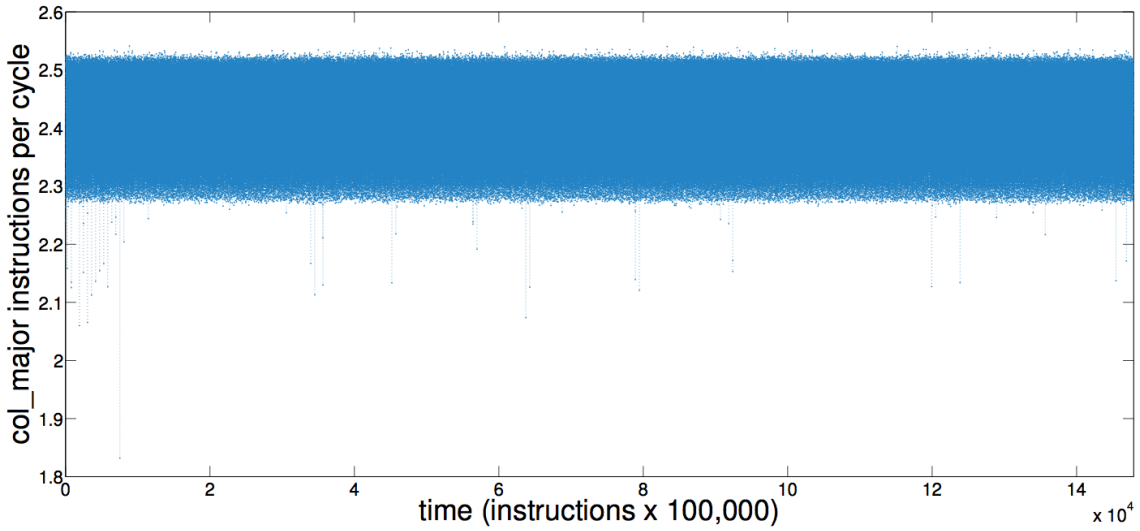



(B) SVD Regime 5 LMA Prediction with Full Time Series

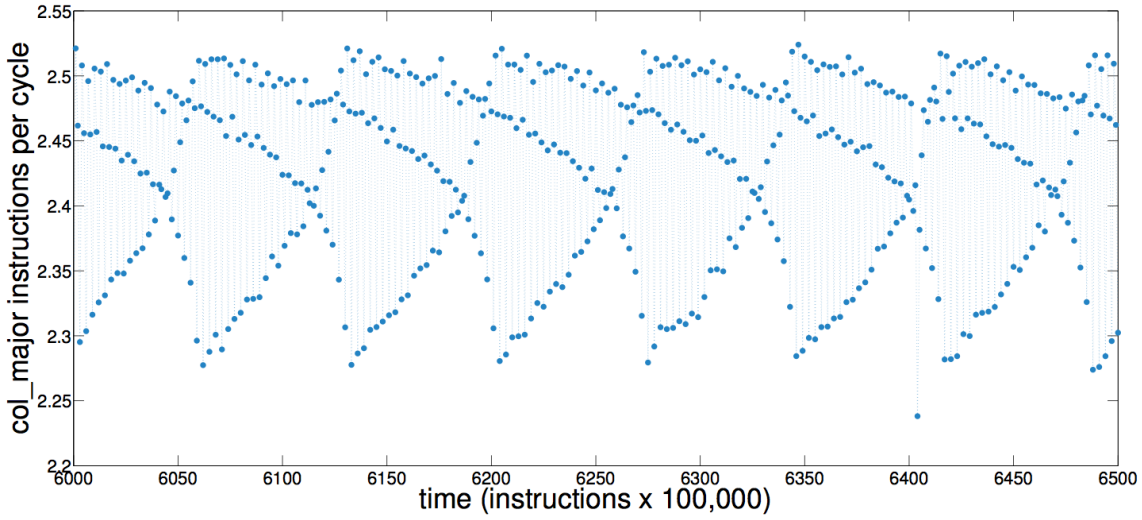
FIGURE 4. SVD Regime 5 LMA Prediction Figures

a *single* dynamical system. To an experimental-time-series analyst these are luxuries that are rarely (if ever) realized. Many times experimental observation have noise involved such as sensor error, or very commonly in the computer world finite-precision error.

Unfortunately the standard algorithms for calculating many of these complexity measures are highly sensitive to noise and give spurious results in the presence of noisy observations. Even worse, if a time-series observation measures multiple dynamical systems




(A) col IPC Full Time Series

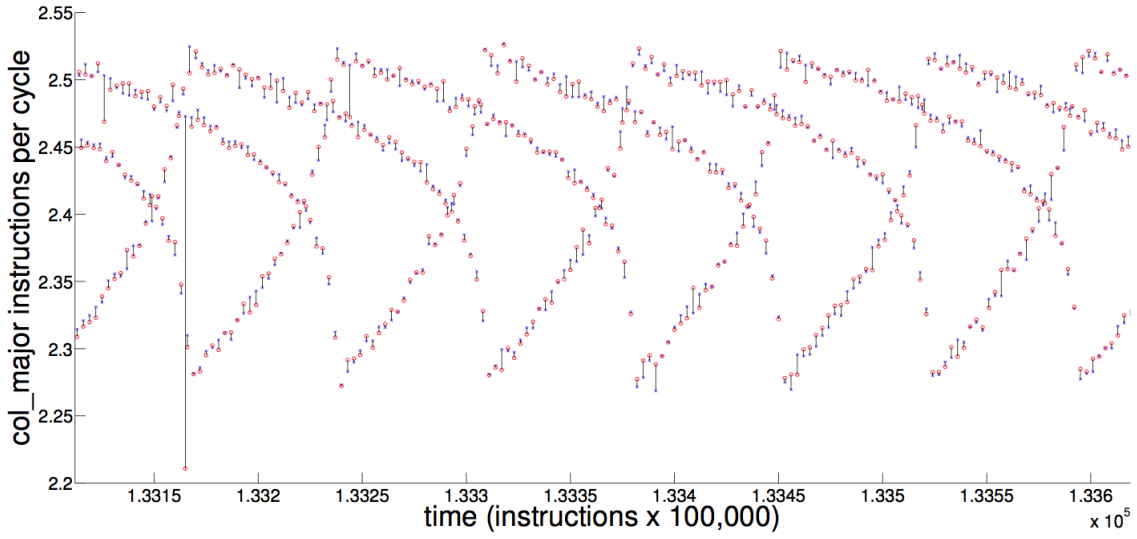



(B) col IPC Short Time Series for structure illustration

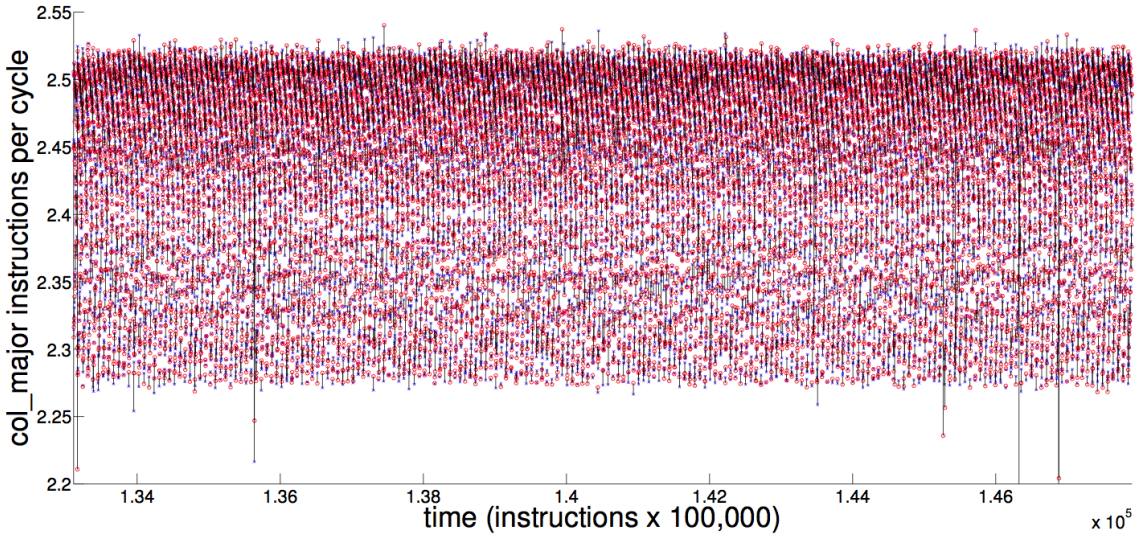
FIGURE 5. col Time Series

or dynamical systems that undergo bifurcations[1] then all of the standard algorithms for estimating complexity, (e.g., correlation dimension) go out the window. Bandt and Pompe

[1]A drastic change in the behavior of the dynamical system, generally associated with parameter drift.




(A) col IPC Full Time Series




(B) col IPC Short Time Series for structure illustration

FIGURE 6. col_major Predictions

first introduced *permutation entropy* as a "natural complexity measure for time series" in 2002 [**?**].

As explained in [**?**], permutation entropy is a simple complexity measure that is easily calculated for any time series, be it stochastic, periodic, quasi-periodic, chaotic, noisy

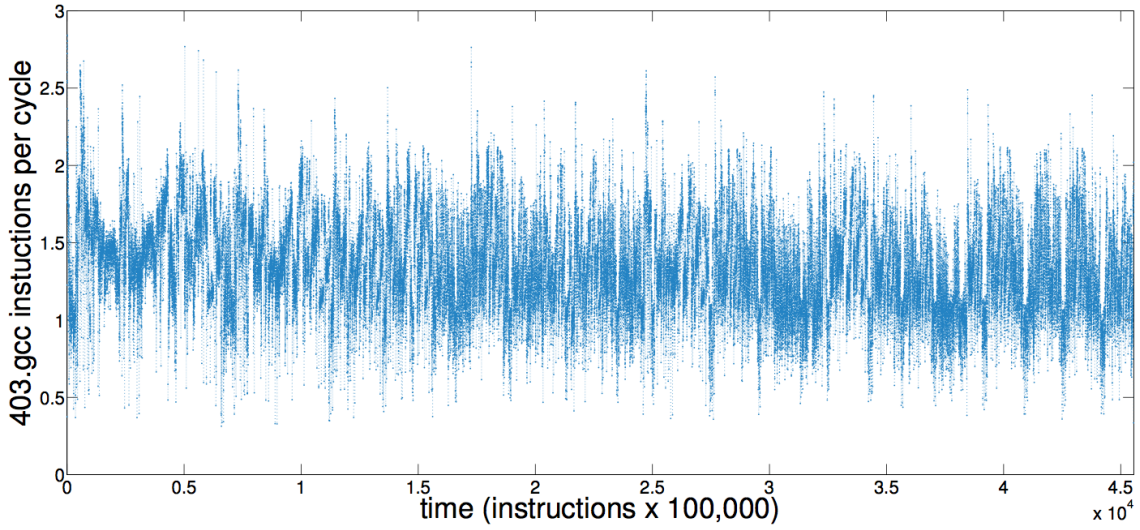



FIGURE 7. 403.gcc full time series

TABLE 3. Average MASE over 15 runs for each signal and average wpe at word length 5 and 6 for each signal.

| | MASE LMA | MASE ARIMA | MASE naïve | $l = 5$ | $l = 6$ |
|---|---|---|---|---|---|
| gcc | 1.9856± | 1.7525± | 1.7904 ± | $0.9510 \pm 0.0011$ | $0.9430 \pm 0.0013$ |
| col_major | 0.0587± | 0.9936± | 0.5713± | $0.5636 \pm 0.0031$ | $0.5131 \pm 0.0034$ |
| SVD_IPC_Regime1 | ± | ± | ± | $0.9761 \pm 0.0084$ | $0.9572 \pm 0.0156$ |
| SVD_IPC_Regime2 | 2.3753± | 0.7268± | 3.1064± | $0.8760 \pm 0.0052$ | $0.8464 \pm 0.0044$ |
| SVD_IPC_Regime3 | ± | ± | ± | $0.7768 \pm 0.0073$ | $0.7157 \pm 0.0056$ |
| SVD_IPC_Regime4 | ± | ± | ± | $0.9073 \pm 0.0080$ | $0.8246 \pm 0.0077$ |
| SVD_IPC_Regime5 | 4.9079± | 2.3366± | 21.0372± | $0.7333 \pm 0.0076$ | $0.6776 \pm 0.0068$ |
| SVD_IPC_Regime6 | ± | ± | ± | $0.8101 \pm 0.0135$ | $0.7475 \pm 0.0106$ |

or any combination of these. Most importantly for analysis of computer performance, permutation entropy as defined in [**?**] yields meaningful results even if observational or dynamical noise is present [**?**].

Analyzing entropies as a complexity measure is not a new concept, however the way that the entropy is estimated in [**?**] is novel and particularly useful when little is known about the underlying process. As discussed in [**?**], the standard entropy one considers is Shannon entropy.

As a quick review of Shannon entropy, we follow the treatment in [**?**]. Given a finite alphabet $\mathcal{S} = \{s_1, ...., s_{|\mathcal{S}|}\}$, define an information source as a discrete-time, stationary stochastic process $X = \{X_n\}_{n \in \mathbb{N}_0}$ where $X_n$ are random variables on a common probability

TABLE 4. Embedding Parameters for reference if needed.

| | $\tau$ | $m$ |
|---|---|---|
| gcc | 10 | 13 |
| col_major | 2 | 12 |
| SVD_Full | 10 | 12 |
| SVD_IPC_Regime1 | 5 | 14 |
| SVD_IPC_Regime2 | 10 | 12 |
| SVD_IPC_Regime3 | 2 | 9 |
| SVD_IPC_Regime4 | 3 | 11 |
| SVD_IPC_Regime5 | 23 | 10 |
| SVD_IPC_Regime6 | 30 | 12 |

TABLE 5. TCM NRMSE and PE NORMS

| TCM | NRMSE | $m = 3$ | $m = 4$ | $m = 5$ | $m = 6$ |
|---|---|---|---|---|---|
| `row_major` | 0.0102 | 0.7900 | 0.6751 | 0.5458 | 0.4491 |
| `col_major` | 0.0055 | 0.6411 | 0.5029 | 0.4515 | 0.3955 |
| `403.gcc` | 0.0865 | 0.9954 | 0.9916 | 0.9880 | 0.9835 |
| `482.sphinx3` | 0.1142 | 0.9958 | 0.9913 | 0.9866 | 0.9802 |

TABLE 6. IPC NRMSE and PE NORMS

| ipc | NRMSE | $n = 3$ | $n = 4$ | $n = 5$ | $n = 6$ |
|---|---|---|---|---|---|
| `row_major` | 0.0095 | 0.9912 | 0.9723 | 0.9354 | 0.8876 |
| `col_major` | 0.0071 | 0.9428 | 0.8356 | 0.7601 | 0.6880 |
| `403.gcc` | 0.1805 | 0.9918 | 0.9862 | 0.9814 | 0.9764 |
| `482.sphinx3` | 0.2946 | 0.9978 | 0.9951 | 0.9914 | 0.9849 |

TABLE 7. Normalized root mean squared error (nRMSE) of 4000-point predictions of memory & processor performance from different programs.

| | cache miss rate | instrs per cycle |
|---|---|---|
| `row_major` | 0.0324 | 0.0778 |
| `col_major` | 0.0080 | 0.0161 |
| `403.gcc` | 0.1416 | 0.2033 |
| `482.sphinx3` | 0.2032 | 0.3670 |

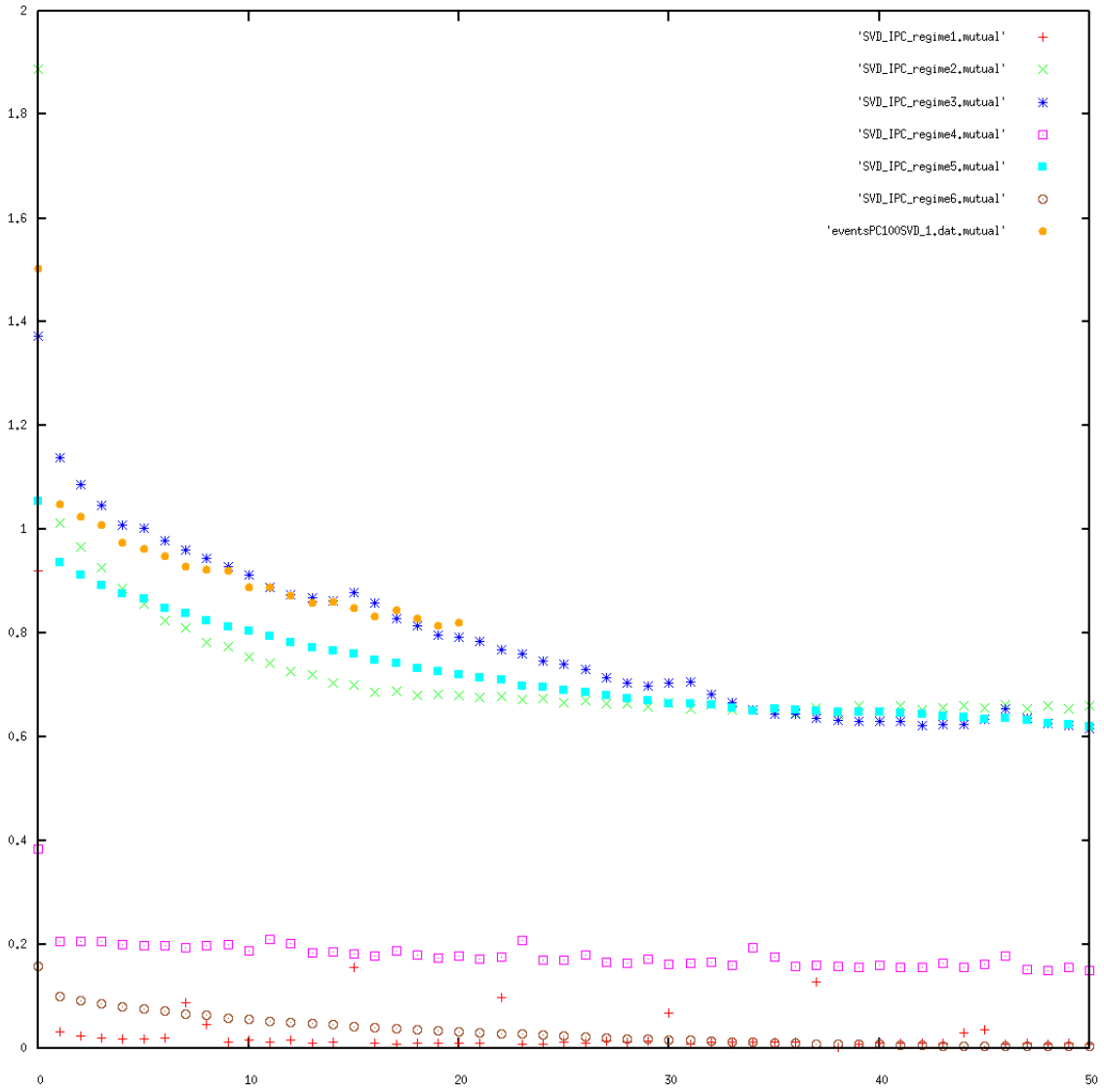



FIGURE 8. SVD mutuals for full and regimes

space taking on values from $\mathcal{S}$. A realization of $X$ is a one-sided sequence $x_0^\infty = (x_n)_{n\in\mathbb{N}_0}$ called a *message*. The elements $x_n$ of $\mathcal{S}$ are the *symbols*(a.k.a letters) of the language. A finite segment of the message, say $x_k^{k+m-1} = x_k x_{k+1} \dots x_{k+m-1}$ is called a *word* (of length m). We then define $p(x_0^{m-1})$ to be the probability of the word $x_0^{m-1}$ to appear some where

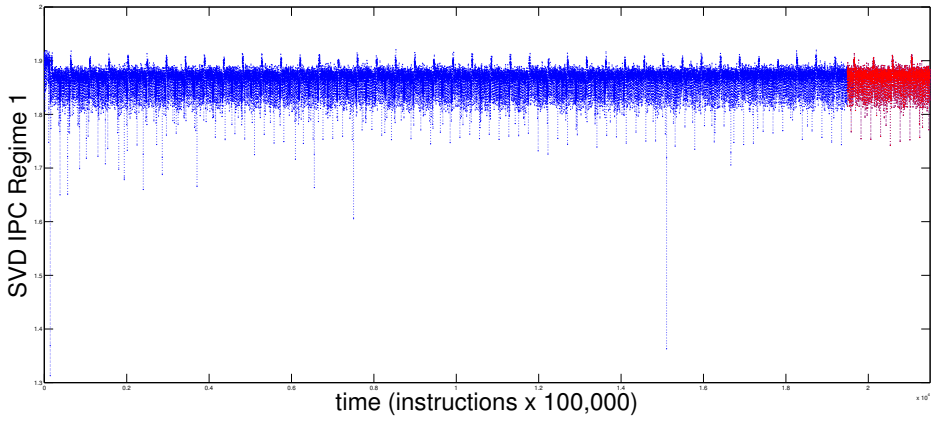



FIGURE 9. SVD IPC Regime 1 Time Series

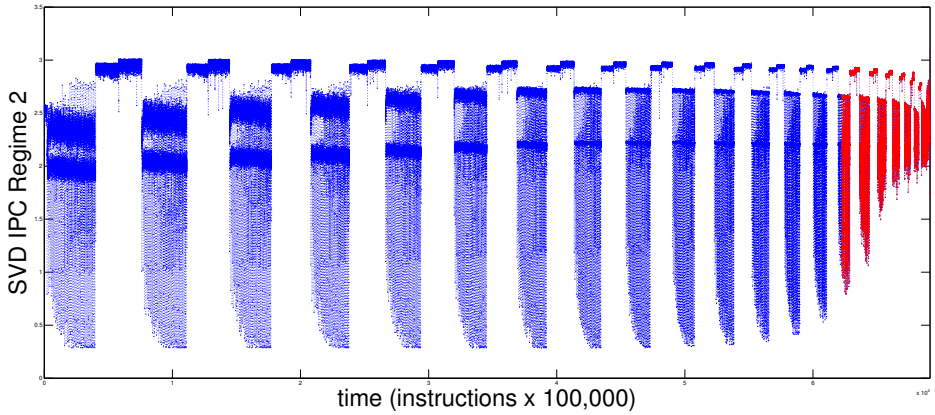



FIGURE 10. SVD IPC Regime 2 Time Series

in the message. The shannon entropy of the data source $X$ is defined as:

$$h(X) = -\lim_{m\to\infty} \frac{1}{m} \sum p(x_0^{m-1}) \log p(x_0^{m-1}) \tag{1}$$

For the purposes of this paper once can view entropy as a measure of complexity and predictability in a time series. A high entropy almost completely unpredictable and a small entropy is completely predictable. This spectrum is outlined very nicely in [**?**]: "Periodic or quasiperiodic sequences have vanishing or negligible complexity. At the opposite end, independent and identically distributed random sequences (white noise) have asymptotically divergent permutation entropies, owing to the fact that the number of allowed (or

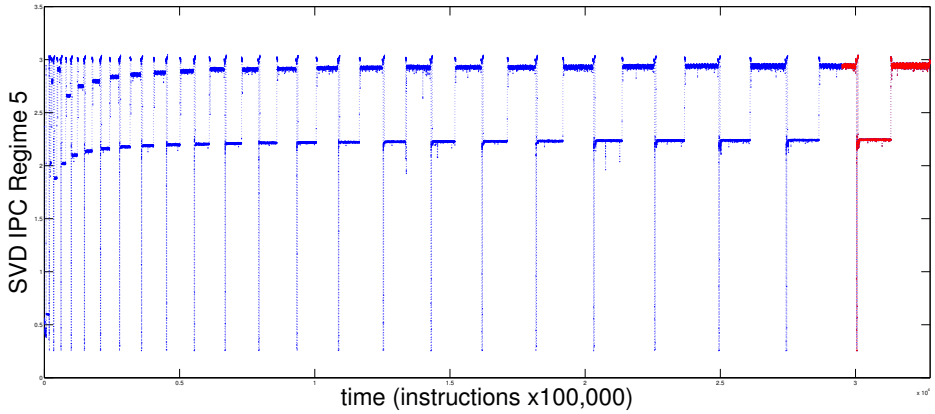



FIGURE 11. SVD IPC Regime 5 Time Series

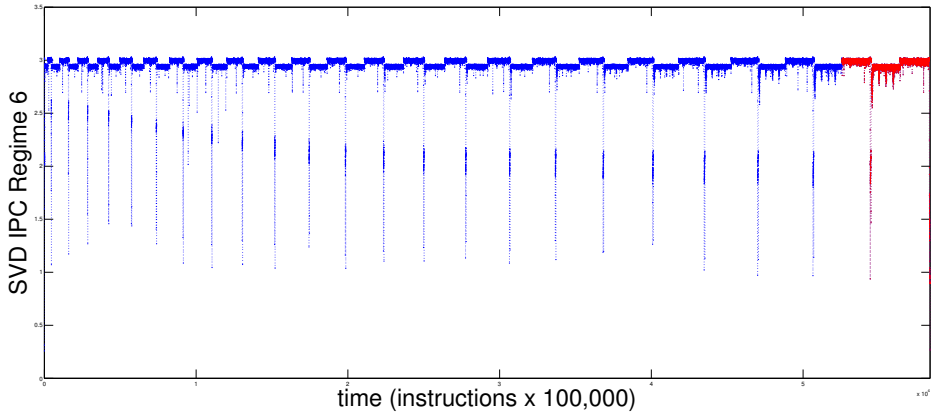



FIGURE 12. SVD IPC Regime 6 Time Series

"admissible?) ordinal patterns grows superexponentially with length. Between both ends lie the kind of sequences we are interested in."

This illustrates why utilizing entropy as a measure for complexity is, in theory, a great candidate. There are several fundamental drawbacks with this approach when analyzing real-world time series. The two most fundamental problems are as follows:

(1) Elements of an experimental time series $x_t \in \mathbb{R}$ which implies $x_t \notin \mathcal{S}$, for a finite alphabet $\mathcal{S}$.
(2) Time series are always finite in length so the limit in 1 never exists in practice.

The common practice to address item 1 is to perform *symbolization* on the time series you want to analyze. Symbolization is the process of defining a dynamic preserving map $\phi : \mathbb{R}^m \to \mathcal{S}$, i.e., for each $x_t$ (or block $x_k^{k+m-1} = x_k x_{k+1} \dots x_{k+m-1}$) we must assign a symbol from $\mathcal{S}$ in such a way that the time series has the same fundamental properties. There are several approaches to symbolization, the standard approach is partitioning. Assume, $x_t$ has the capability of taking on infinitely many values, defining the space $X$ to be the space where these $x_t$ are drawn. We then define a partition on this space $X = P_1 \cup ... \cup P_m$, and define $\mathcal{S} = \{1, \dots, m\}$. Then $\phi(x_t) = i$ if $x_t \in P_i$. We can then calculate the Shannon entropy of the transformed time series, which in the limit of $m$ converges to the Kolmogorov-Sinai entropy [**?**]. This approach to symbolization is computationally infeasable as we need an infinite number of partitions of an infinite time series. For *extremely* rare cases, there exists a *generating* partition where this limit is not necessary. However, finding such a partition even for a simple two-dimensional map like Henon is non-trivial [**?**]. Finding a generating partition for an arbitrary time series is for all intensive purposes impossible.

[[[Maybe also talk about Mischaikow's approach to this with isolating neighborhoods and homology...but might be off topic. ]]]

.....

Instead of this approach to symbolization, Bandt et al. take the approach that "the symbol sequence must come naturally for the time series, without any further model assumptions" [**?**]. To do this they perform symbolization through ordinal analysis. Ordinal analysis of a time series is the process of mapping the ordered successive elements of a time series to a permutation of the same size based on their original time-based ordering.[[Need a better way of wording this]]. This is also referred to in the literature as the permutation complexity [**?**] For example, if $(x_1, x_2, x_3) = (9, 1, 7)$ then $\phi$ maps $(x_1, x_2, x_3)$ to the permutation 231 since $x_2 < x_3 < x_1$. A great explanation of why this is an interesting symbolization is found in [**?**] :

> The study of permutation complexity, which we call ordinal analysis, can be envisioned as a new kind of symbolic dynamics whose basic blocks are ordinal patterns. Interesting enough, it turns out that under some mild mathematical assumptions, not all ordinal patterns can be materialized by the orbits of a given one- or multi-dimensional deterministic dynamics, not even if this dynamic is chaotic? contrarily to what happens with the symbol patterns. As a result, the existence of ?forbidden? (i.e., not occurring) ordinal patterns is always a persistent dynamical feature, in opposition to properties such as proximity and correlation which die out with time in a chaotic dynamic. Moreover, if an ordinal pattern is forbidden, its absence pervades all longer patterns in form of more missing ordinal patterns, called outgrowth forbidden patterns. Admissible ordinal patterns grow exponentially with length, while forbidden patterns do superexponentially. Since random (unconstrained) dynamics has no forbidden patterns with probability 1, their existence can be used as a fingerprint of deterministic orbit generation.

Essentially by analyzing the permutation entropy of a time series we are gaining information as to the complexity of the system without any other information available, whether the time series is generated by a random process or the trajectory of a deterministic dynamical system. Now that we have justified why the study of permutation entropy is interesting let us define it. For this we follow [**?**].

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

$$p(\pi) = \frac{|\{t|t \leq T-n, \phi(x_{t+1},\dots,x_{t+n}) = \pi\}|}{T-n+1} \tag{2}$$

*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* $0 \leq H(n) \leq \log_2(n!)$*,*[**?**]*. With this in mind, it is common in the literature to present permutation entropy normalized, i.e.,* $\frac{H(n)}{\log_2(n!)}$*. With this convention,* low *entropy is close to 0 and* high *entropy is entropy close to 1.*

Notes: According to [**?**] equation (2) estimates the frequency of $\pi$ as good as possible for a finite series of values. To determine $p(\pi)$ exactly, we have to assume an infinite time series and take the limit as $T \to \infty$ in (2). Bandt et al., also point out that this limit exists under a very weak stationarity condition: for $k \leq n$, the probability for $x_t < x_{t+k}$ should not depend on $t$.

It should also be noted that this is discrete-metric (or shannon) permutation entropy, to be distinguished from topological permutation entropy (a.k.a. permutation capacity). Topological permutation entropy is defined in [**?**] as:

$$h(\{x_t\}_{t=1,\dots,\infty}) = \lim_{L\to\infty} \frac{1}{L} logN(L)$$

where $N(L)$ is the number of distinct ordinal patterns that occur of order $L$. This is a measure of how many distinct permutations of length $L$ are realized as opposed to counting the frequency at which each occurs as is the case in metric permutation entropy [**?**]. It can be shown that metric and topological entropy both converge to the true value with probability 1 in the limit. (citation for this). When we say permutation entropy in this paper we are referring to that defined in 2.

## 5. Detecting changes in a time series

Permutation entropy is a measure of the complexity of time series as a whole, but it can also be used to calculate a dynamical shift in a time series as illustrated in [**?**]. To detect dynamical changes in a time series we follow the methods of [**?**]. Partition the time series $\{x_t\}_{t=1,\dots,T}$ into overlapping[2] (or nonoverlapping) blocks. Calculate $H(n)$ for each block,

[2]Following [**?**] we use maximal overlapping block. (A window shift of 1.)

and treat $H(n)$ as a function of time. According to [**?**] drastic changes in permutation entropy can indicate change in the underlying model.

In addition to introducing this block method, [**?**] introduce the use of a *lag* when calculating permutation entropy. This is identical to the lag used in delay coordinate embedding and often the same language is used. Cao et al. describe the process of symbolization in [**?**] as embedding the time series in an $m$ dimensional symbol space. When we symbolize a scalar time series,$\{x_t\}_{t=1,\ldots,T}$, we first embed the scalar time series into $m$ dimensional vectors before matching this to a symbol in $\mathcal{S}_m$. So pre-symbolization our time series changes from scalar to objects of the form:

$$[x(i),\, x(i+1),\, \ldots, x(i+(m-1)]$$

This vector is then mapped by $\phi$ to a symbol in $\mathcal{S}_m$. Cao et al. argue that we can view this as embedding the time series in an $m$ dimensional space with lag 1. Cao et al. then state "Since in practice the optimal $L$[lag] may be different from 1, we shall present the idea for any $m$[embedding dimension] and $L$."[3] To not confuse notation we define $\tau$ to be the lag parameter. The process of symbolization then starts with transforming the scalar time series into a set of vectors of the form

$$[x(i),\, x(i+\tau),\, \ldots, x(i+(m-1)\tau]$$

where $\tau \geq 1$.

5.1. **Example: Concatenated Time Series.** As an example of this detection, consider Figure **??** (a). This is a plot of a concatenated time series comprised of several orbits of the Logistic map with varying parameters as well as intermittent noise. Each concatenated band (either an orbit of the Logistic map or noise) is 50,000 points. The bands of noise are generated with Matlab's `randsample`, using the chaotic bands as the sampling population. Each Logistic map trajectory begins at $x = 0.6$. The period-two trajectories have an $R$ of 3.2, the period-three trajectory has an R of 3.838, and the chaotic trajectories have an $R$ of 3.65.

The concatenated time series is comprised of two period-two trajectories($t = 0 - 50,000$ and $t = 250,000 - 300,000$), one period-three trajectory($t = 100,000 - 150,000$), two chaotic trajectories ($t = 50,000 - 100,000$ and $t = 200,000 - 250,000$, and two different bands of noise($t = 150,000 - 200,000$ and $t = 300,000 - 350,000$). Visually the bands of noise and the chaotic trajectory are indistinguishable. If this were an observed time series from a real process, it would be impossible to visually identify which bands were generated deterministically and which were simply noise. If we calculate block permutation entropy as discussed in [**?**] and plot this as a function of time it becomes very clear which bands have structure and which do not.

In Figure **??** (b) we plot the concatenated time series, as well as permutation entropy as a function of time(red, blue and green curves). The red, blue and green curves are the permutation entropy calculated using $m = 5$ and $\tau = 1, 2, 3$ (respectively). Using these curves it is clear that bands 2 and 5 ($t = 50,000 - 100,000$ and $t = 200,000 - 250,000$) are

[3]No further explanation is given about why this is the case.

chaotic because they have structure[4]. On the other hand, bands 4 and 7 ($t = 150,000 - 200,000$ and $t = 300,000 - 350,000$) have no structure[5] and are the noise bands. This is in fact the case when comparing to the way the time series was constructed.

The main advantage we can see to the use of lags other than 1 is for analysis of periodicity. This is also the major advantage listed in [**?**]. Notice in the period 2 bands the blue curve (permutation entropy calculated with a lag of 2) is zero. This means that with a lag of 2 the period 2 orbit is perfectly structured[6]. If we increase the lag to 3 (the green curve) the two period-two bands have a little bit of structure(as some permutations will be realized) but the period 3 band has zero permutation entropy.

and what is it good for with forward pointer to the other sections

5.2. **Example: The transient Logistic map.** In Section 5.1 we discussed a time series which stayed in a parameter regime for long amounts of time and then abruptly changed. As we showed, permutation was a great proxy to detect this change. Another type of parameter change that may occur in a physical system is constant parameter drift, i.e., some parameter slowly changes over time. To explore this type of parameter drift we construct the same time series as was used in [**?**]. Iterate the logistic map, starting with $R = 2.8$, then at every iteration increase $R$ by $10^{-5}$ until $R = 4$. The time series, which can be seen in Figure 13 appears very similar to the standard bifurcation diagram for the Logistic map. The curves in red and blue are the permutation entropy calculated with $m = 5$ and $\tau = 1, 2$ (respectively).

As we can see, as bifurcations occur in the dynamics the permutation entropy changes as well. An interesting feature of block calculations are the spikes that occur at bifurcation points. As the block begins to encompass multiple regimes, e.g., fixed point and period two, permutations are being realized for both regimes. This overlap causes an increase in the time series complexity, and thus a spike in the permutation entropy. The dips you see in the second half correspond to bands of periodicity which occur between the chaos. It is worth noting, that the blue curve (permutation entropy with lag 2) does not go to zero in the period 2 parameter range. This is because this is not a true period 2 orbit but transient that is moving across *several* period 2 orbits, thus it maintains a higher level of complexity, albeit still very low.

This illustrates that permutation entropy can not only distinguish between instant (and drastic) parameter shifts but also very slow and gradual parameter drift.

## 6. Possible Applications to Computer Systems Performance

(1) In this section I want to discuss the symbolization of performance traces and why this is advantageous over generating partitions and similar approaches.
(2) No model is assumed, some traces do appear to be stochastic some appear to be deterministic this provides "proof" thereof. albeit not mathematical proof but CS proof for sure.

[4]Permutation entropy less than one.

[5]Permutation entropy of one.

[6]Equivalently, not enough permutations are being realized to impact the calculation of $H(n)$

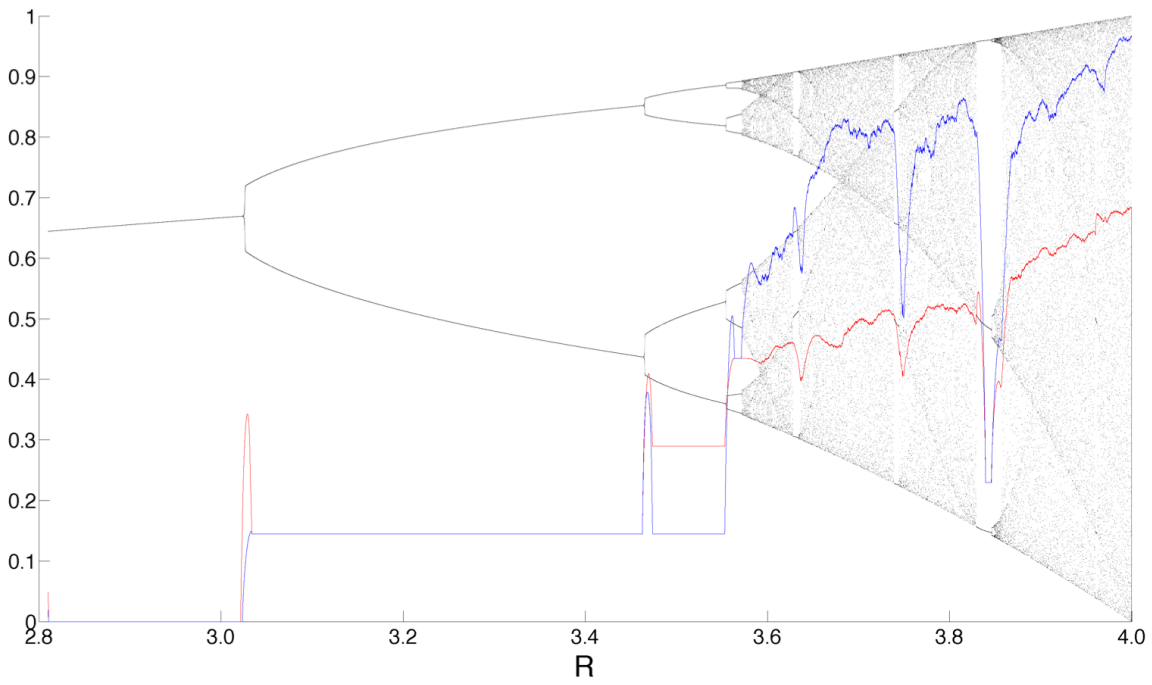



FIGURE 13. Transient Logistic map time series and permutation entropy curves. Both permutation entropy curves are calculated with $m = 5$, the red and blue correspond to $\tau = 1, 2$ respectively.

(3) We have essentially infinite data. So limits like $m!$ where $m$ is word length isn't really an issue for us.

6.1. **Predictability and model change detection.** Talk about how some explanatory models may be spinning their wheels because no transferable information. That is if you are introducing pure entropy at every time step really no use in using past values and statistically the mean is the most accurate.

(1) Comp Performance goes under constant bifurcations. This allows autonomous detection of when a model is no longer valid. (need to be careful here with use of valid as to not be confused with model validation.
(2) Maybe can tell us if we should use a linear-stochastic, nonlinear-deterministic model or just guess the mean

## 7. Pro and Cons

7.1. **Advantages of Permutation Entropy.** Permutation entropy allows for symbolization of a time series without any further assumptions being made about the underlying system. In fact, virtually nothing needs to be known before applying this techniques. Other

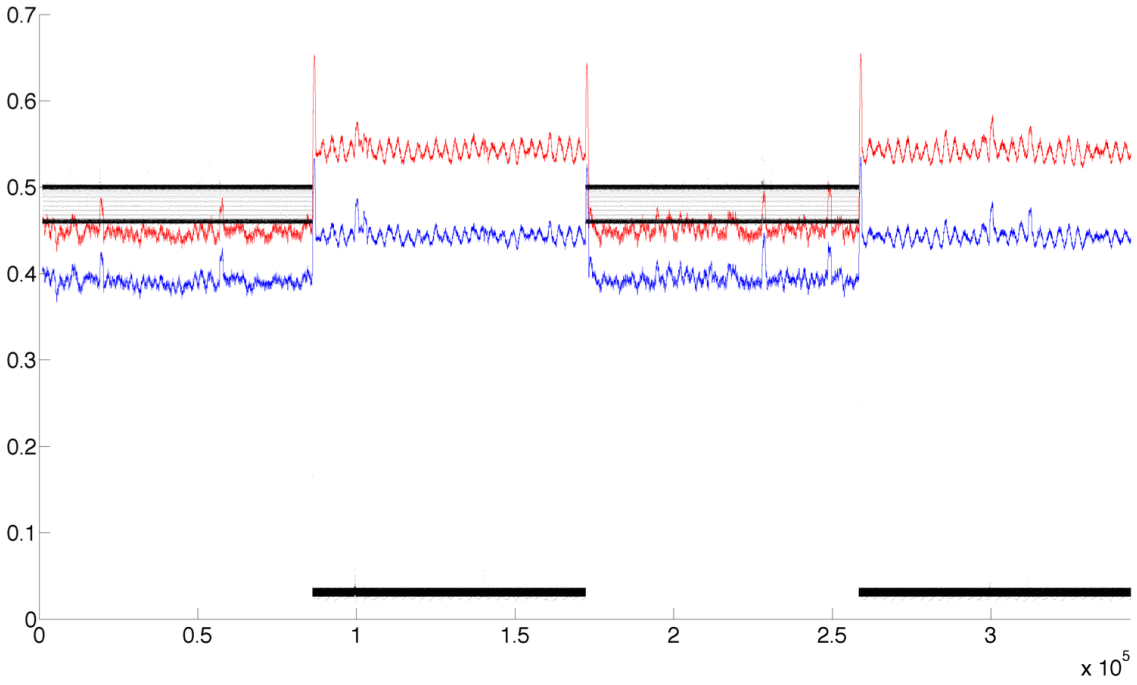



FIGURE 14. A time series of cache misses during the execution of the row-col microkernel. The red and blue curves are permutation entropy calculated using $\tau = 1$ and $m = 5, 6$ (respectively).

methods of symbolization require expert knowledge of the underlying system, such as generating partitions. These techniques have been applied to small data sets, this is truly an advantage especially when dealing with a lot of biological systems whose large datasets can be in the tens of thousands. The algorithm for computing permutation entropy is very straight forward and easy to explain. The actual code takes longer than I would like to run, but this seems to be an issue with my implementation and not the algorithm, as others state they have exceptionally fast algorithms. Although none of these algorithms are ever provided in the literature. The major advantages of permutation entropy is that it can quickly give us a rough estimate of where drastic changes in the time series which may have implications in model selection techniques. Permutation entropy also can give you a quick idea about if a system has deterministic structure. This has direct implications on what prediction strategies to implement.

7.2. **Disadvantages of PE.** There are three disadvantages with permutation entropy we can immediately see that need further thought to be overcome. The first is that it is sometimes hard to tell when a change in the dynamical system is not drastic enough to have a huge change on the entropy. For example, consider a dynamical system switching from a chaotic to a (different) chaotic regime. The permutation entropy may be very

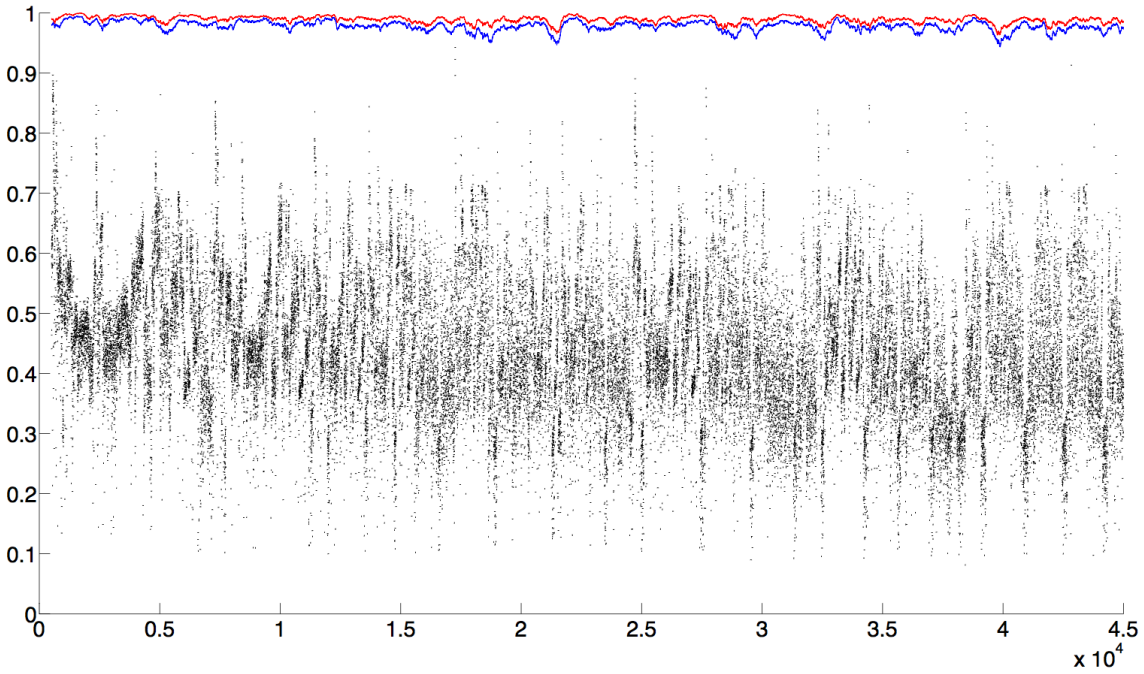



FIGURE 15. A time series of instructions per cycle for the gcc spec benchmark. The red and blue curves are permutation entropy calculated using $\tau = 1$ and $m = 3, 4$ (respectively). The lack of structure in such a signal illustrates that the use of deterministic models is most likely not the most advantageous.

similar while the model is slightly different. Permutation entropy seems to be ill-suited for fine grain detection of this nature. Our current implementation is too slow for on-the-fly analysis. We do not believe this is an issue with permutation entropy itself but with our implementation. The literature all claims that fast algorithms exist.

The biggest disadvantage for us in using permutation entropy is selection of parameters. To calculate permutation entropy, we have to choose an embedding dimension, a lag, and a block size. In all the literature we have found thus far, parameter selection is made based on persistence or matching the permutation entropy to known values. The problem comes when analyzing real time series these parameters seem to have a real impact on the results. Finding (or creating) a rigorous method for parameter selection is paramount in the use of this method for the analysis of real world time series.