JOURNAL OF COMPUTATIONAL PHYSICS 104, 162-172 (1993)

# Direct Calculation of Metric Entropy from Time Series

KEVIN M. SHORT†

*Vitro Corporation, Advanced Technology Lab, 14000 Georgia Avenue, Silver Spring, Maryland 20906-2972*

Received August 17, 1990; revised April 20, 1992

In this paper we develop an algorithm which allows for a direct computation of the metric entropy from time series data. The approach is based on the original definition and enables us to use fine partitions and long sequence lengths. There is a discussion of the underlying theory, followed by an explanation of the computational approach, including methods of partitioning the data, computing the sequential distributions, and compactifying to reduce memory requirements. The approach is tested against periodic, random, and chaotic data for which the metric entropy is known analytically. Then the technique is applied to the Henon, Ikeda, Rossler, Lorenz, and Mackey Glass attractors. The results compare well with those found by other techniques. © 1993 Academic Press, Inc.

## I. INTRODUCTION

When studying dynamical systems, it is possible to classify them into one of three categories: periodic/quasi-periodic, chaotic, or random. One important quantity which distinguishes between these categories is the metric, or Kolmogorov, entropy which will be denoted by $K$ throughout this paper. The usefulness of the entropy arises from the fact that $K=0$ for a periodic/quasi-periodic system, $0<K<\infty$ for a chaotic system, and $K\to\infty$ for a random system.

Numerous researchers have considered entropy calculations. However, it has been problematic to calculate the entropy directly from the original definitions given in [1, 2], since the calculation requires observing the dynamical system over long periods and imposing on the system a partition which must then be varied to take the supremum over all partitions. Further, for an initial partition of size $N$, the dynamical system over time will effectively increase the number of partitions so that there will be $N^b$ partitions after $b$ time steps. In [3] Curry calculates the entropy of the Henon attractor using an initial partition of only two bins and concludes that the finer partitions that would be required for greater accuracy lead to a level of complexity that is beyond present computing capabilities. The difficulties associated with this calculation led Grassberger and Procaccia [4] to develop a technique to calculate a lower bound to the entropy based on a correlation function. The idea of generating a symbol sequence is developed in [5] by Crutchfield and Packard, and a similar approach developing a representation of the Lorenz system as a one-dimensional Ising model is developed by Shimada in [6]. In this paper, we will give a technique for calculating the metric entropy directly from the definition for finer partitions and longer time periods than have been previously used. Certain new problems arise which will be discussed as well.

The paper is divided into several sections. The next section gives a brief introduction to the theoretical underpinnings of the entropy of a dynamical system. Following that we will describe in Section III the computational approach which allows us to overcome the inherent difficulties associated with the calculation. In Section IV results will be given for some test systems where analytical values for $K$ are known. Then Section V will give results for several dynamical systems which have been studied in the literature. The last section will give conclusions and suggestions for further study.

## II. THEORETICAL BACKGROUND

The calculation of the entropy makes use of several concepts from information theory. In this context, one considers the probabilities of certain events occuring, designating the probability of event $i$ by $p_i$. Then the amount of information gained by a measurement on the system is given by [7]

$$I = -\sum_i p_i \ln p_i. \tag{1}$$

This definition is extended to sequences of events $i_1, i_2, ..., i_b$ by taking the probability that this sequence occurs with respect to all other sequences to be $p(i_1, ..., i_b)$, giving

$$I_b = -\sum_{i_1,...,i_b} p(i_1, ..., i_b) \ln p(i_1, ..., i_b). \tag{2}$$

† Present address: Dept. of Mathematics, M.I.T., Cambridge, MA 02139.

0021-9991/93 $5.00

When this concept is applied to dynamical systems, one must consider a partition of phase space, $B$, where the elements of the partition are little boxes or bins, $\beta_i$ (we will generally refer to a one-dimensional partition as a bin, and for higher dimensional partitions they will be called boxes). The events become simply the presence of an orbit of the dynamical system in a particular box, $\beta$, at a given time, $t$. This bears further explanation. Let $X(t)$ define the evolution of the dynamical system in phase space, which can be finite-dimensional or infinite-dimensional. If the system is continuous, consider a sequence of samples at regular time intervals $\Delta t$; if it is discrete, take the discrete values as the samples. In either case, one can consider a time series, $X(t_i)$, of samples. Now it becomes clear how to specify an event, i.e., the sample $X(t_i) \in \beta_j$ for some $t_i$ and some box $\beta_j$. If the dynamical system satisfies the properties of ergodicity (which essentially means that time averages are equivalent to spatial averages [8]) one can simply observe the dynamical system over a (sufficiently) long period to determine the *asymptotic probability distribution of the events*. This allows for the calculation of the information content of the asymptotic distribution, where $p_i \to p(\beta_j) \equiv p(X(t_i) \in \beta_j)\ \forall t_i$ is the probability that the dynamical system visits the box $\beta_j$ over time and the information becomes a function of the chosen partition of the phase space. The result analogous to (1) becomes

$$I(B) = -\sum_{\beta_i} p(\beta_i) \ln p(\beta_i), \tag{3}$$

where the sum is taken over all of the elements of the partition. Similarly, it is possible to study sequences of events for dynamical systems by defining the joint probability that the dynamical system visits the sequence of boxes $\beta_1, \beta_2, ..., \beta_b$ at times $t_1, t_2, ..., t_b$, respectively, to be $p(\beta_1, \beta_2, ..., \beta_b)$. Here the numbering of the $\beta$'s from 1 to $b$ is merely a convenience, and is meant to indicate an arbitrary sequence of $b$ boxes. The result that follows from (2) is

$$I_b(B) = -\sum_{\beta_1, ..., \beta_b} p(\beta_1, ..., \beta_b) \ln p(\beta_1, ..., \beta_b), \tag{4}$$

where the sum is taken over all possible sequences of length $b$. It is also useful to note that when the probability of a given sequence is 0, we have as a limit $p \ln p \to 0$, so there is no contribution from such sequences to the sum. The maximum occurs when all of the sequences are unique and, hence, have equal probabilities. We will say that an $X\%$ separation level has been reached when the number of distinguished sequences is equal to $X\%$ of the total possible number of sequences for finite data. Before going on to define the entropy, it is important to observe that the definition given by (4) is dependent on the chosen partition, the length of the sequences, and, for a continuous system, the sampling interval.

In this study of dissipative dynamical systems, we will consider only the attractor on which the long-term evolution takes place. Information on the technical subtleties in the definitions of attractors can be found elsewhere [5]. The attractors that arise for periodic systems are either fixed points (e.g., for a damped pendulum) or limit cycles, which are closed loops in phase space (e.g., for an undamped pendulum); similarly for quasi-periodic systems, where the attractor is an $n$-torus because the $n$ closed oscillatory variables have incommensurate periods. Chaotic systems generally exhibit attractors that lie in a bounded, finite-dimensional subset of phase space and have the property that they generally evolve on a set which has fractional dimension. Related to this is the property termed sensitive dependence on initial conditions, which means that the orbits of two points initially close in phase space will diverge over time. Truly random systems would in general exhibit *attracting sets which fill the available phase space*.

The definition of the metric entropy, first proposed by Kolmogorov [1] and later clarified by Sinai [2], relies heavily on the information-theoretic concepts described above. The actual definition makes use of an invariant ergodic probability measure, $\mu$, on the phase space of the dynamical system. These matters are discussed at great length elsewhere, notably in Eckmann and Ruelle [9], as well as in the expanded version of this paper [22]. Given an attractor $A$ with measure $\mu$, contained in some bounded $n$-dimensional subset $M^n$ of phase space, we can see that a partition $B$ on $M^n$ will induce a partition $\Phi$ on $A$. We then associate the $\mu$-measure of a set $\phi_i \in \Phi$ in $A$ with the probability that the dynamical system visits the partition $\beta_i$, so $\mu(\phi_i) \equiv p(\beta_i)$. This extends to sequences through refinement of the initial partition $\Phi$ and the equivalent extension in terms of probabilities of sequences. The entropy is defined ([1, 2]; see also [11]) as the supremum over all partitions and all time intervals $\Delta t$, of the sequential information (4) as the sequence length goes to infinity, with a normalization factor $1/b\,\Delta t$, giving

$$K = \sup_{B, \Delta t} \left[ \lim_{b \to \infty} \left( -\sum_{\beta_1, ..., \beta_b} p(\beta_1, ..., \beta_b) \times \ln p(\beta_1, ..., \beta_b) \right) \Big/ b\,\Delta t \right], \tag{5}$$

where the sum is over all sequences of length $b$. This can be written in another form which will be useful in some situations,

$$K = \sup_{B, \Delta t} [\lim_{b \to \infty} (I_{b+1}(B) - I_b(B))/\Delta t], \tag{6}$$

where we have used (4) to simplify the notation.

Some of the inherent difficulties in computing the entropy from (5) or (6) have been discussed in the Introduction. It is immediately obvious that the limit as the sequence length goes to infinity is intractable, since any time series data must be finite. For an experimental situation, the supremum over the sampling interval $\Delta t$ is generally inaccessible, since the sampling rate is chosen at the time of data collection. The partitioning of the sampled data can be varied, but it is certainly only reasonable to do the calculation for a few different partitions. With these provisos in mind, we must hope that the values for $K$ which are calculated are asymptotic to the correct value within a reasonable number of time steps and for partitions which are not too fine.

For certain special dynamical systems, known as Axiom A systems [12], there is in fact a simple initial partition, called the generating partition, which produces the exact value for $K$ (see, e.g., the discussion in [3]). In the absence of any knowledge of the availability of a generating partition for a dynamical system, we can only take finer partitions to better approximate the generating partition, since a subdivision of a generating partition is generating. As was pointed out in [9, 3], the use of finer partitions improves the accuracy of the estimate of $K$. Of course, there are some restrictions to this, since any finite number of distinct samples can always be placed into separate boxes. If these boxes are taken to be the partition, then the system will be in its maximum information state for that number of samples, and it will not be possible to calculate the entropy accurately. This assumes that we have a strange attractor or a random system, since a periodic/quasi-periodic system can have identical points. The net effect of this restriction is that the size$(\beta) \to 0$ limit is not available, where size$(\beta)$ refers to the physical size of the boxes in our partition. Further, as the sequence length is increased, the initial partition is refined, so for finite data the situation can arise where after several time steps each sequence in the sample data resides in its own box in the refined partition. In fact, it will be shown later that the values of $K$ calculated for a finite number of samples begin to lose accuracy when more than 20% of the sequences have been separated. This concept of "relatively infinite" sequences and samples presents only a slight problem because it can be easily resolved by considering segments of the samples. This will be expounded upon below.

In the computations that will be described, the dynamical systems which are considered were allowed to evolve for fairly long time periods. The computations were performed in single precision on a BBN GP-1000 parallel processor, with 16 nodes available (each based on a Motorola 68020 processor). The code runs efficiently in serial, although the distributed memory was used. This will be discussed in more detail later. The difference equations were simply allowed to evolve, and the differential equations were integrated using a fourth-order Runge–Kutta method with integration intervals on the order of 0.01. Generally, several integrations were performed before a sample data point was recorded in the data set, with the sample interval dependent on the length of a "characteristic time" [13] for the system, where we follow [13] by using this to mean either the mean time between intersections of a Poincaré section or the time associated with a dominant feature in the power spectrum. In all cases the number of data points was large enough to "flesh out" the attractor. The study was limited to one of the dimensions from the data points for the multi-dimensional systems, although the program can be run on the full multi-dimensional data set. We considered sequences starting at every point in the data set to determine the global behavior on the attractor. In fact, roundoff error and sensitive dependence imply that the points in the data set can be taken as independent starting points, so this is a valid approach.

## III. COMPUTATIONAL APPROACH

The calculation of the entropy presents several difficulties for numerical studies. These can be attributed to problems associated with partitioning of the data and keeping track of the sequences. For a point that goes through the sequence of bins $\beta_1, \beta_2, \ldots, \beta_b$, the sequence can equivalently be considered as a point in euclidean space of $b$ dimensions by using the bin labels to create a $b$-vector. Here it becomes obvious that for an initial partition into $N$ bins, there are $N^b$ boxes in the equivalent $b$-dimensional space. For fine partitions it is evident that as $b$ grows a memory overflow would occur in even a large computer if every $b$-dimensional box were stored, so it becomes imperative that some compactification scheme is included with the sequencing. Below we set out an approach with several sections: A. Partitioning the data; B. Sequences and compactification; C. Probabilities and information, D. Entropy.

Certain conventions will be employed to make the presentation clear. Program statements will be written in a generic language which should be self-evident. Loops will be indicated by a "for $i$ = start to $i$ = finish" scheme, where we may use < instead of = as appropriate. Nested loops will have the inner loop indented and will, of course, use a different indexing variable. When "if/then" types of statements are used, the antecedent will be separated from the consequent by a comma. For arrays the convention has been chosen that when a Greek index is used, e.g., $A[\alpha]$, we are referring to the array itself, and when a Latin index is used, e.g., $A[i]$, we are referring to a particular element of $A[\alpha]$.

### A. *Partitioning*

For computational efficiency, it is imperative that the partitioning of the data be accomplished quickly and efficiently. We will assume that all of the samples of the dynamical system have been stored in a vector, Data[$\alpha$], of length *npts*, and *npts* is the number of starting points to use in the sequencing process. To perform the partitioning, we have taken advantage of the mask and shift operations available in the C programming language. To use this capability, we must convert the data into a representation of the sample values as integers. Since the attractors of interest are bounded, this is possible by adjusting the range of the data. If the input data lies in the range $[x_{\min}, x_{\max}]$, the first step is to subtract $x_{\min}$ from every element of Data[$\alpha$]. This shifts the data so that it is in the range $[0, x_{\max} - x_{\min}]$. Now scale the data by an appropriate multiplicative factor so that it covers the range [0, MaxInt], where MaxInt is the largest integer the computer can represent. Finally, set the adjusted elements of Data[$\alpha$] equal to the elements of an integer vector, Bin[$\alpha$].

The elements of Bin[$\alpha$] are integers which exhibit the same relative proportions as the initial input data. Consequently, any partition of the original data is equivalent to a *partition of the data in* Bin[$\alpha$] *simply by scaling the partition* in the same manner that the original data was scaled. It is evident that this process has no effect on the calculation of the entropy, since the entropy depends solely on the sequence of boxes through which a point moves, and this is equivalent under the indentification of the original data with the scaled data. The identification gives an isomorphism between the initial data and initial partition, and the scaled data in Bin[$\alpha$] and the scaled partition; the entropy is invariant under the isomorphism.

Since the data in Bin[$\alpha$] are integers, we can operate on them using the mask and shift capability. The mask process simply calculates a logical AND between the bit string representing an element of Bin[$\alpha$] and a programmer-specified bit string. The shift function simply right-shifts a specified bit string by a certain number of places. So, if the data in Bin[$\alpha$] is to be partitioned into $2^n$ bins, mask off all but the leftmost $n+1$ bits, then mask off the sign bit, leaving only the $n$ most significant bits representing the data value (this assumes, of course, that the most significant bits are on the left). The remaining values are then shifted to the right until they are right justified. The numbers that remain in Bin[$\alpha$] now lie in the range $[0, 2^n - 1]$; consequently, they can be interpreted as labels of the bins in the initial partition.

The partitioning has now been accomplished. It becomes a simple task to calculate the initial distribution of the data over a partition of $N = 2^n$ bins. By creating a vector Dist[$\beta$] of length $N$, the distribution can be calculated by taking the elements of Bin[$\alpha$] as the indices in Dist[$\beta$] and incrementing the elements of Dist[$\beta$]; i.e., Dist[Bin[$i$]] = Dist[Bin[$i$]] + 1, $\forall i$, where the right-hand side is meant in the usual programming sense of incrementing the previous value of Dist[Bin[$i$]] by 1.

The partitioning process described above is quite easily computed, and takes very little CPU time. The elements of Bin[$\alpha$] are all taken to be starting points of sequences, so it is clear that to determine the sequence of bins followed by a particular starting point, all that is necessary is to look at the elements of Bin[$\alpha$] following that starting point. To calculate a sequence probability $p(\beta_1, \beta_2, \ldots, \beta_b)$, all we have to do is count the number of times that the particular sequence of numbers $\beta_1, \beta_2, \ldots, \beta_b$ appears in Bin[$\alpha$]. This is not as simple as it appears, since it requires keeping track of $N^b$ data points, which soon overwhelms the memory of a computer. The resolution of this problem is discussed in Section III.B below.

Before going on to discuss sequencing and compactification, it is worthwhile to note that the use of binary partitioning does not place any restriction on the freedom to choose a partition. By adjusting the scaling factor to leave certain bins in the [0, MaxInt] partition empty and by incorporating a roundoff algorithm in the conversion from floating point to integer, one can effectively choose any *partition* [23].

## B. *Sequences and Compactification*

For the given partition into $N$ bins, it was mentioned above that a particular sequence of bins $\beta_1, \beta_2, \ldots, \beta_b$ can be represented by a point in an euclidean space of $b$ dimensions, i.e., on an integral grid $N \times N \times \cdots \times N$, where the product is taken $b$ times. This simply means that if $p$ is a point on the $b$-dimensional grid, with coordinates $(\beta_1, \beta_2, \ldots, \beta_b)$, then $\beta_1, \beta_2, \ldots, \beta_b$ represents a possible sequence of bins through which a point on the attractor might move in $b$ time steps. Consequently, it would be a simple matter to tabulate all of the sequences on the attractor if we could create a $b$-dimensional array with each dimension of length $N$ and use the elements of Bin[$\alpha$] to index the array and increment the proper positions. Let such an array be called Hist[$\beta_1, \beta_2, \ldots, \beta_b$]. Then the tabulation is given by Hist[Bin[$i$], Bin[$i+1$], ..., Bin[$i+(b-1)$]] = Hist[Bin[$i$], Bin[$i+1$], ..., Bin[$i+(b-1)$]] + 1, where $i$ varies over all *npts* starting points. Once this is calculated, the total probabilities can be computed, and hence the information. The problem, of course, is that this requires $N^b$ memory positions. Luckily, many of these positions have zero entries (corresponding to no such sequences occurring); in fact, the maximum number of non-zero entries is *npts*, since there cannot be more entries than there are starting points of sequences. This allows us to compactify the Hist[$\beta, \gamma$] array at each stage of the computation to minimize the memory requirements.

We will now describe a simple algorithm for compactifying the sequencing information. While there are probably more efficient approaches, this one has two important benefits. First, it requires no more than a fixed amount of memory and only a two-dimensional array, Hist[$\beta, \gamma$], with a maximum size less than or equal to $npts \times N$. Second, it does not lose the sequencing information, whereas any algorithm which utilized a sorting routine would have to overcome the problem of re-ordering and the subsequent loss of sequences. The algorithm will be described for the minimum case of a two-dimensional Hist[$\beta, \gamma$] array, but if more memory were available it would be a simple exercise to generalize to higher dimensions.

To begin the sequencing process, create a new vector, Compress[$\alpha$], and initialize it so that Compress[$i$] = Bin[$i$], $\forall i$. Next create the two-dimensional histogram array Hist[$\beta, \gamma$]. We will also need another variable to keep track of the number of non-zero positions in Hist[$\beta, \gamma$]; call it Newcount. Initially we set Newcount $= N$. The Hist[$\beta, \gamma$] array can either be dynamically allocated to its working size $N \times$ Newcount or set to the maximum size of $npts \times N$. The first step is to increment Hist[$\beta, \gamma$] for two-sequences (where we introduce the term $b$-sequence to mean a sequence of length $b$) according to

$$\begin{aligned}&\text{Hist}[\text{Bin}[i+1], \text{Compress}[i]]\\ &\quad = \text{Hist}[\text{Bin}[i+1], \text{Compress}[i]] + 1, \qquad (7)\end{aligned}$$

where $0 \leqslant i < npts$ is the range of $i$. Every two-sequence increments a particular position in Hist[$\beta, \gamma$], and identical two-sequences increment the same position in Hist[$\beta, \gamma$], so the Hist[$\beta, \gamma$] array represents the distribution of two-sequences.

We now make use of the possibility of interpreting a two-dimensional array as a one-dimensional vector (as is generally done in most computers when storing arrays in memory) by concatenating either over rows or columns. For example, if $A[\phi]$ is an $\alpha \times \beta$ array, then a position in $A[\phi]$ may be labeled by $A[i, j]$ for $0 \leqslant i < \alpha$, $0 \leqslant j < \beta$, or by concatenating over columns as $A[j * \alpha + i]$, or by rows as $A[i * \beta + j]$. This property will be used to set up a correspondence between two-sequences and positions in the Hist[$\beta, \gamma$] array, where we replace the starting points in Compress[$\alpha$] by their relative positions in Hist[$\beta, \gamma$], labeled by concatenation over rows. To do this, simply make the replacement

$$\text{Compress}[i] = \text{Compress}[i] + \text{Newcount} \times \text{Bin}[i+1] \qquad (8)$$

for all $i$ in the range $0 \leqslant i < npts$. No information is lost in this process, since all unique two-sequences produce unique values for the associated elements in Compress[$\alpha$]. When this is completed, Compress[$i$] contains the position in Hist[$\beta, \gamma$] corresponding to the two-sequences starting at the associated point Bin[$i$]. Note that none of the zero entries in Hist[$\beta, \gamma$] are reflected in Compress[$\alpha$], so the numbers in Compress[$\alpha$] do not, in general, come from a contiguous interval of integers. In order to minimize the memory requirements, it is necessary to renumber the values in Compress[$\alpha$], so that if there are $p$ unique values, they will be renumbered from 0 to $p-1$, with their order maintained.

The compression process is fairly simple. First, designate a variable, Newcount, to count the number of nonzero entries in Hist[$\beta, \gamma$]. Then, in one pass through Hist[$\beta, \gamma$], replace the old value of Hist[$i, j$] with the current value of Newcount and then increment Newcount. This gives simply

```
RowLength = Newcount
Newcount = 0
        for $i = 0$ to $i < N$
                for $j = 0$ to $j < $ RowLength
                        if Hist[$i, j$] > 0,
                        then Hist[$i, j$] = Newcount AND
                        Newcount = Newcount + 1
```

At the end of this step, the nonzero entries in Hist[$\beta, \gamma$] represent their relative orders of appearance in Hist[$\beta, \gamma$].

Now the compression can begin. The numbers in Compress[$\alpha$] represent the old bin numbers previously in Hist[$\beta, \gamma$], but we now replace them with the ordered numbers placed in Hist[$\beta, \gamma$] in the last step.

```
for $i = 0$ to $i < npts$
        column = Compress[$i$] mod RowLength
        row = Compress[$i$]/RowLength
                (Comment: integer divide)
        Compress[$i$] = Hist[row, column]
```

which results in the elements of Compress[$\alpha$] being numbered from 0 to (Newcount $-$ 1).

At the next stage of the calculation, when we want to consider three-sequences, the working size of Hist[$\beta, \gamma$] need only be $N \times$ Newcount. The Compress[$\alpha$] vector again holds the correct numbers to allow for the use of the Hist[$\beta, \gamma$] incrementing routine (7), where the next value in the sequence is selected from Bin[$\alpha$], i.e., Bin[$i+2$]. This process continues recursively, where for $b$-sequences we use Bin[$i + (b-1)$] in (7) and (8) and the compressed values of Compress[$\alpha$]. Note that if the allocation of Hist[$\beta, \gamma$] is dynamic within the sequencing loop, it is necessary to free the reserved memory space before going on to the next step.

## C. *Probabilities and Information*

After going through the $b$-sequence compression loop, the Compress[$\alpha$] vector contains values that uniquely identify the $b$-sequences followed by the starting points. Consequently, to calculate sequential probabilities all that must be done is to tabulate the relative frequencies of occurrence of the different sequences. This can be done simply by using the elements of Compress[$\alpha$] as indices to increment positions in a probability vector. So, create a vector, Prob[$\alpha$], of length Newcount (since that is the number of unique values in Compress[$\alpha$]) and increment the elements as below:

```
for $i = 0$ to $i < npts$
        Prob[Compress[$i$]] = Prob[Compress[$i$]] + 1
```

When the incrementing is completed, it is necessary to normalize the probabilities so the total sum is 1, so normalize by computing

```
for $i = 0$ to $i <$ Newcount
        Prob[$i$] = Prob[$i$]/$npts$
```

Once the probabilities are computed, it is an easy task to compute the information.

The calculation of the information for $b$-sequences follows from (4). This is given in a loop by

```
Info = 0
    for $i = 0$ to $i <$ Newcount
        Info = Info + Prob[$i$] * ln Prob[$i$]
```

where *Info* is the variable representing the information. From (4) it is evident that the result of the loop calculation above must be multiplied by $-1$, so the last step is to take

$$\text{Info} = (-1) * \text{Info}.$$

At this stage the information content of $b$-sequences has been computed. It is now possible to continue on to calculate the information content in the $(b+1)$-sequences by keeping Compress[$\alpha$] and the value of Newcount, clearing Hist[$\beta, \gamma$], and looping back through the operations in Sections III.B and III.C above.

### D. *Entropy*

All of the above calculations were done for a given partition $B$. Bearing in mind the provisos mentioned in Section II, it is useful to calculate an estimate of $K$ as a function of $B$, call it $K(B)$, for the partition $B$ and sampling interval $\Delta t$ of the input data. To do so, we use (5) or (6) and leave out the supremum over $B$ and $\Delta t$. From the information computations described above, it is possible to plot a graph of information vs sequence length. In theory, for an infinite amount of data, this graph should be asymptotic to a line, the slope of which corresponds to the entropy. However, with finite amounts of data this is not the case and, as was discussed in Section II, we find that after a certain number of time steps the information graph begins to approach the maximum information state for the number of sequences under consideration. As will be seen in the next section, this causes the graph to fall off from the line defining the entropy, approaching the slope = 0 state near the maximum information. In certain situations, notably the difference equations, there tends to be a noticeable linear region before the falloff toward maximum information, and in this case we will simply plot a best-fit line through the points in the linear region to estimate the entropy. This corresponds more to the definition (5). For the systems based on differential equations, there is not such a distinctive linear region, so we use (6) to estimate $K$.

The computational requirements of this approach can be calculated easily. Subtracting the minimum of the data, scaling the data, and masking and shifting the data all go as $O(npts)$. The one-dimensional histogramming, i.e., incrementing Dist[$\beta$], requires just one pass through the data, so it is $O(npts)$. Incrementing the two-dimensional array Hist[$\beta, \gamma$] is likewise accomplished in $O(npts)$. Then, renumbering Compress[$\alpha$] requires one pass through Hist[$\beta, \gamma$] which has $N \times$ Newcount elements, where $N$ is fixed by the choice of the number of initial bins and Newcount can never grow larger than $npts$, although the calculation should be cut off soon after Newcount $> 0.20 * npts$, as will be discussed later. For $N \ll npts$, the renumbering is $O(npts)$, or for larger $N$, $O(N \times npts)$. Incrementing Prob[$\alpha$] requires $npts$ operations and calculating *Info* requires $3 *$ Newcount operations. Consequently, this implementation is accomplished in $O(npts)$ or $O(N \times npts)$ operations.

The memory requirements involve only one large array Hist[$\beta, \gamma$], all others being vectors of lengths $\leqslant npts$. The working size of Hist[$\beta, \gamma$] is $N \times$ Newcount and cannot grow beyond its maximum size of $N \times npts$; actually, if the calculation is cut off soon after the 20% separation level has been reached, then the size of Hist[$\beta, \gamma$] is $\leqslant N \times npts \times A$, for $A \approx 0.20$.

The program was run on a BBN GP1000 parallel processor with 16 processing nodes. The code runs on only one node as a serial program; however, it was found that scattering the Hist[$\beta, \gamma$] matrix over the memory of different processors sped up the processing by reducing memory contention. While this was done using a utility function for parallel processing, it can be done in C by simply allocating each row of the matrix separately with an auxilliary vector holding pointers to the first point in each row, making sure that the rows are separated in memory.

## IV. TEST CASES

To test the approach described above, we will consider a periodic system, a random system, and a chaotic system with known entropy. The periodic system will be represented by sinusoidal data. The random system will be represented by a quasi-random number generator of period $2^{31} - 1$. The known chaotic system will be represented by the logistics map with $K = \ln 2$. We will refer to the imaginary curve through the points in the information vs sequence length graph as the progression curve. In all cases we will generate more values than our sample size and, since each data point in the sample is a starting point of a sequence, the extra values allow us to look at sequences extending beyond the end of the sample.

The results for the periodic system are based on a sinusoidal system, where the period was 24 time samples in length and a total of $10^5$ values were recorded. The initial partition was 64 bins. Using a 50,000-point sample and a 99,900-point sample, the asymptotic slope was zero for both, so $K=0$. At no point did either system approach its maximum information state.

The results for the random system are based on a quasi-random number generator which is optimal for a 32-bit machine. In all, $5\times10^5$ random numbers were generated. Using three samples of 50,000, 99,900, and 499,900 points taken from the $5\times10^5$ random numbers generated, it was found in all cases that the system approached the maximum information state within essentially one time step after the number of available boxes of the partition became greater than the number of points. Hence, the slope does not converge and increases as the sample size increases. This is consistent with a truly random system which would have the slope increasing without bound as the sample size increases.

The behavior of the chaotic systems is between the two extremes of periodicity and randomness, and the information vs sequence length graph in Fig. 1 exhibits some properties of both. The system under study is the logistics map

$$x[t+1]=R*x[t]*(1-x[t]),$$

where the case $R=4$ is known to have entropy $K=\ln 2$.

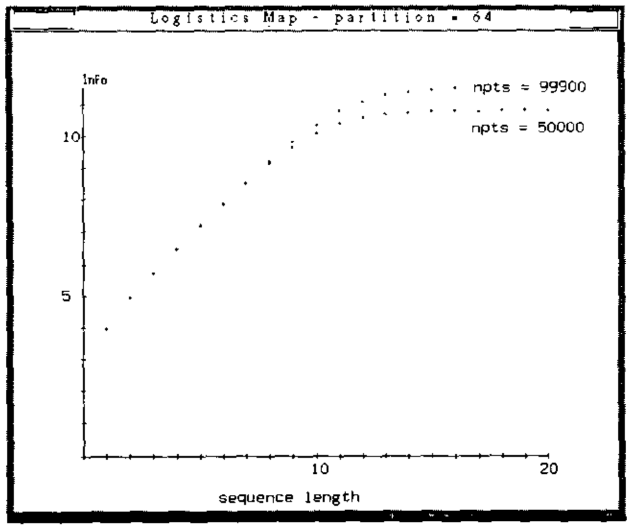

FIG. 1. Information vs. sequence length graph for the logistics map using a 64-bin initial partition on samples of 50,000 and 99,900 points.

Again $10^5$ values were generated, from which a 50,000-point sample and a 99,900-point sample were taken. For the case when the initial partition is 64 bins (Fig. 1), we see that there is initially a sharp rise in the information vs. sequence length graph, followed by a nearly linear region, and then the plots approach their maximum information states. At this point the slopes fall off from the asymptotic line, which has known slope ln 2. However, it is also evident that as we take a larger sample, the information vs. sequence length graph follows the "correct" asymptotic line for a longer sequence length. This has been verified for intermediate-sized samples as well. Consequently, for a given sample size the graph will at first rise sharply, then it will follow the asymptotic line until it is affected by its approach to the maximum information state. It appears from these and additional studies that the valid region occurs after the initial steep rise and before 20% of the sequences have been separated. In the previous section, the variable Newcount represented the total number of sequences that had been separated, so the 20% level is easily monitored. For the graph in Fig. 1 there is a distinct linear region between sequence lengths 3 and 9, so to estimate the entropy we have calculated the best-fit line in this region. This is equivalent to using (5) to compute the entropy. The slope of the best-fit line gives $0.683608\pm0.007905$, and the actual value $K=0.6913$ is in this range, so the technique is quite accurate. These results are virtually independent of the chosen partition.

## V. OTHER CHAOTIC SYSTEMS

In this section, the results for the entropy calculation for the Henon, Ikeda, Rossler, Lorenz, and Mackey–Glass attractors will be given. The first two systems are mappings, while the others are generated by systems of differential equations. The results are summarized in Table I, and a plot of the information versus sequence length graph will be given for the Henon, representative of the mappings, and for the Rossler, representative of the systems of differential equations. We will give some guidelines which help to determine if the results of a calculation represent a decent approximation to the long-time limit for $K$. All of the systems exhibited the same chaotic properties as the logistics map in the previous section regarding sample size and falloff towards the maximum information state at the 20% separation level. Consequently, we will choose a fixed sample size of 99,900 points taken from $10^5$ generated values and report results for various partition sizes.

### A. *Henon Attractor*

The Henon attractor [14] is a simple two-dimensional system. Three different initial partitions were used, corre-

TABLE I

| System | Initial partition | Entropy |
|---|---|---|
| Henon | 32 | 0.449374 ± 0.009192 |
| | 64 | 0.444722 ± 0.011422 |
| | 128 | 0.459532 ± 0.014949 |
| Ikeda | 32 | 0.375503 ± 0.010519 |
| | 64 | 0.384369 ± 0.012088 |
| | 128 | 0.397354 ± 0.016012 |
| Rossler | 8 | 0.1022145 |
| | 16 | 0.125425 |
| | 32 | 0.1572875 |
| Lorenz | 8 | 1.0239763 |
| | 16 | 1.1389654 |
| | 32 | 1.3786 |
| Mackey–Glass | 8 | 0.0078898 |
| | 16 | 0.0107874 |
| | 32 | 0.0169141 |

sponding to 32, 64, and 128 bins. The results of the calculation of information vs. sequence length are presented in Fig. 2 for the three partitions. Note that the initial steep rise lasts for three time steps, after which the graphs enter a linear region. For a finer partition the falloff toward the maximum information state occurs after fewer time steps. There is also a tendency toward bowing of the progression curve in the linear region for the 128 bin partition. This is symptomatic of the partition becoming "too fine." These are important considerations when evaluating the entropy, and their impact will be seen more clearly in other systems. In this case there is little difference, since the graphs in the linear region are essentially parallel.

Since the linear region is fairly evident, we calculate a best-fit line through the points up to the 20% separation level. The results are given in Table I. In all cases the error is purely statistical, and no attempt has been made to correct for effects at the tails of the linear region. These results compare favorably to those in the literature. The reported numerical estimate from the initial equations is 0.4192 [13, 14]. Grassberger and Procaccia [4] find for the lower bound $K_2 = 0.318 \pm 0.02$ from numerical techniques and $K_2 = 0.325 \pm 0.02$ from time series data, whereas Curry [3] finds $K \approx 0.40 \pm 5\%$.

### B. *Ikeda Map*

The Ikeda map [15] is also generated by a system of difference equations. The results of the entropy calculations are in Table I. The characteristics of the progression curves seen earlier in the other chaotic maps were quite evident here as well. The curves for the 32- and 64-bin partitions were much closer to linear than for the 128 bin partition. We attribute this to the 128-bin partition being a bit too fine for the data.

### C. *Rossler Attractor*

The Rossler attractor [16] results from the evolution under a system of coupled differential equations. In various places in the literature, different parameters are used; the ones used here correspond to those in [13]. The integration interval was taken to be 0.01 s and sample values were recorded after every 200 iterations. As discussed in [13], the characteristic time for an orbit to traverse the attractor is 6.07 s, so this implies that we are sampling three times per traversal. The results for the information vs. sequence length graph are exhibited in Fig. 3. The initial partitions used in this case are 8, 16, and 32 bins.

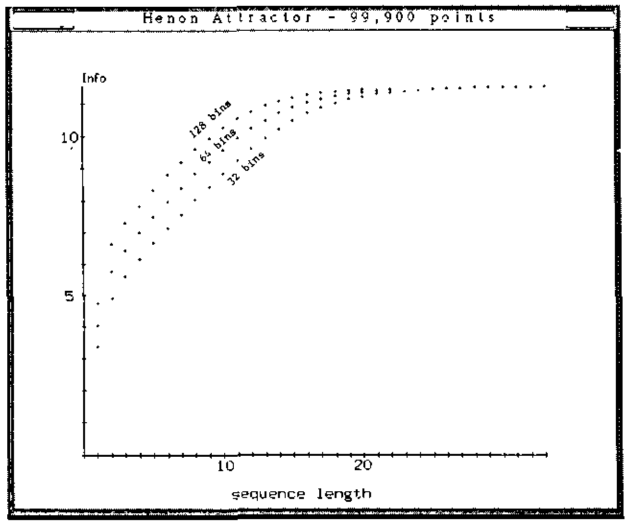

FIG. 2. Information vs. sequence length graph for the Henon attractor using 99,900 points and initial partitions of 32, 64, and 128 bins.

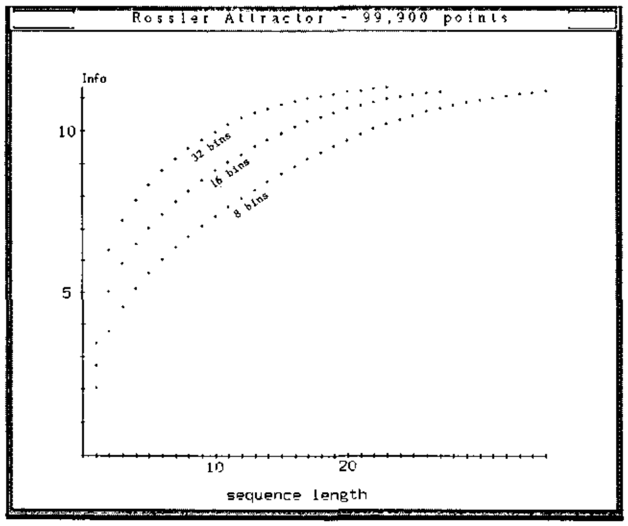

FIG. 3. Information vs sequence length graph for the Rossler attractor using 99,900 points and initial partitions of 8, 16, and 32 bins.

The Rossler system is a continuous-time system, and it exhibits some properties that are different from those of the difference equations. First, in a numerical experiment it is possible to vary the time interval between taking sample values, so it is actually possible to consider different sampling intervals. Second, there is a much greater tendency toward bowing of the information vs. sequence length graph for all of the systems based on differential equations. The bowing is very pronounced when the partition is too fine. This can be interpreted as a coalescing of the initial steep rise with the falloff toward the maximum information state. This makes it difficult to discern a linear region. Finally, for the differential equation systems, it is important to keep in mind that the long-time limit is the desired result, so the 20% separation level should not be reached before the system has evolved for a period at least several times longer than the characteristic time for the attractor.

The graph in Fig. 3 illustrates some of the difficulties with interpreting the entropy for the differential equation systems. First, the attractor approaches its asymptotic distribution of partitions much more slowly than we have seen before, since the initial steep rise lasts much longer. The graph of the 32-bin progression curve exhibits noticeable bowing, which is less pronounced in the 16-bin curve. The 8-bin curve gives a nice linear region between about time steps 8 and 18 (which, in fact, corresponds to the 20% separation limit). Since it is the long-time limit which is of interest and since the linear regions are more difficult to discern, we will use Eq. (6) to estimate the entropy. We wish to avoid the falloff region, so we will take the limit as the progression curves approach the 20% separation limit, i.e., $I_{b+1}(B)$ in (6) will be taken at the 20% separation limit. The limits were taken after 16, 24, and 36 s for the 32-, 16-, and 8-bin partitions, respectively. The characteristic time to go around the attractor was $\approx 6$ s, so the limits were taken after 2.67, 4, and 6 times around the attractor.

Comparing these results to those reported in the literature gives some good insight into our technique. According to the theorem stating that the entropy should be less than or equal to the sum of the positive Lyapunov exponents, we can compare our values to the positive Lyapunov exponent reported in [13], which is equal to $\approx 0.09$ taken to the base of the natural logarithms. We find that the best approximation to this value occurred for the 8-bin partition. This is also the partition that produced the most discernible linear region in the information vs. sequence length graph, and it also went six times around the attractor before reaching the 20% separation limit. It is apparent that the finest partition is not necessarily the most desirable candidate, and it turns out that the most accurate results are generally derived from the partition that gives the longest time limit with the flattest linear region. Numerous other runs were calculated for the Rossler attractor using varying sample intervals and varying partitions, and the entropy estimates were found to be dependent primarily on the total time before the 20% separation limit was reached. The results approached the result in [13] as this time increased.

## D. *Lorenz Attractor*

The Lorenz attractor [17] is a simple set of differential equations designed to model convection in the atmosphere, where the parameters chosen reflect the original ones in [17]. In this case it is difficult to determine a characteristic time, so we will consider the average time it takes to loop through one of the wings in the typical "butterfly" representation (see, e.g., [13]) of the Lorenz attractor. For the parameters given here, this time is approximately 0.75 s. The system was again integrated with an integration interval of 0.01 s, and samples were recorded every 55 iterations, so this is better than once per loop on average.

The results are in Table I, and the information vs. sequence length progression curves exhibited the same characteristics as for the Rossler attractor. There was slight bowing in the 32-bin progression curve, but more problematic was the fact that there were very few points in the linear region before the fall-off toward the maximum information. For the 32-bin curve there were only two points in the linear region. For the 16-bin and 8-bin curves, there were four and six points, respectively, in the linear region. For the parameters used, the 32-bin result corresponds to 3.67 loops, the 16-bin result corresponds to 5.13 loops, and the 8-bin result corresponds to 6.6 loops. Coupled with the fact that the linear region for the 8-bin graph was distinct and flat, this would lead us to expect that the 8-bin result would represent the best estimate of the entropy. This appears to be the case since Caputo and Atten [11] report a lower bound on the entropy as $K_2 = 0.9 \pm 0.1$.

## E. *Mackey–Glass Equation*

The Mackey–Glass equation [18] is a delay-differential equation which deals with a biological system. For the system under study, we have chosen the delay = 17. The system was evolved numerically using an integration interval of 0.05 s, and samples were recorded every 400 iterations.

The results of the information vs. sequence length computation exhibited less distinctive linear regions, especially for the 32-bin partition. In fact, there was some bowing present in all of the curves, although there was a reasonably linear region in the 16-bin and 8-bin curves. The results are in Table I. The 20% limit occurred after 8.20 s, 13.20 s, and 20.20 s for the 32-bin, 16-bin, and 8-bin partitions, respectively. The values for the entropy reported in the literature are $K = 0.00525 \pm 0.0001$ from Lyapunov exponent equivalence [4], the lower bound $K_2 = 0.0042 \pm 0.0002$ [4]. We see that the result for the 8-bin partition is approaching the correct value. The fact that the progression curves were more bowed for the Mackey–Glass attractor than for the previous attractors indicates that the sample interval was probably not optimal, and this accounts for the results being farther away from the reported values.

## VI. CONCLUSIONS

The primary result of this research is the demonstration that it is possible to calculate the metric entropy from the fundamental definition. This can be accomplished for fine partitions and long sequence lengths by using the partitioning and sequence compression techniques described in Section III. Further, there is a major benefit in this approach because it requires less than some fixed amount of memory, which is dependent only on the partition and the number of starting points. The procedure developed here is designed for use on time series data and should provide a useful tool for determining whether experimental data is chaotic.

The calculations of information vs. sequence length presented in Sections IV and V exhibit progression curves that provide a typical signature of a chaotic system, where there is an initial steep rise in the graph, followed by a linear region, with a final falloff toward the maximum information state for a given number of starting points. The progression curves are monotonically increasing while their slopes are monotonically decreasing. The results of these investigations have shown that if the entropy is calculated as the slope approaches the 20% separation limit, then the values derived reflect an upper bound to the true entropy. The effect of partition selection is not as drastic as might be expected, because too fine a partition causes the data to reach the 20% separation limit before the long-time limit has been reached. The most accurate results can be discerned by selecting the partition (and time sampling interval, if applicable) that produces the flattest linear region and longest time limit at the 20% separation level. The results are derived using only time series data, with no dependence on the equations of motion, and the results were favorable in comparison to those reported elsewhere.

The procedure for application to experimental data would be to collect the data, select a partition, then calculate the information vs sequence length graph and the entropy for that partition. Observing the progression curve, it would then be necessary to determine the acceptability of the result based on the flatness of the linear region and the length of time before the 20% separation limit is reached. If the length of time is not several times longer than a characteristic time for the system, and if the initial steep rise overlaps the falloff region, there are several possible ways to improve the results. One obvious possibility is to choose a different partition. In some circumstances it may be possible to vary the sampling interval. Finally, it may be necessary to consider more data points over a longer period of observation.

The results reported here considered one-dimensional samples of the attractors. The computational approach is easily extended to handle multi-dimensional data. It is also possible to consider the entropy in a reconstruction of the attractor (based on mutual information [19, 20], for example). This can be accomplished by stepping through the $\text{Bin}[\alpha]$ vector according to the desired reconstruction scheme (in (7) and (8) take $\text{Bin}[i+1]$ as $\text{Bin}[i+\tau]$ and choose $\tau$ accordingly). Preliminary findings suggest that the entropy may not vary much from that derived for the one-dimensional samples, but more work must be completed before that can be concluded. Finally, it may be possible to extend the partitioning procedure with compactification to calculate the Hausdorff or box counting dimension efficiently [21].

## ACKNOWLEDGMENTS

This work was accomplished under the Independent Research and Development Program of Vitro Corporation. Computational results were obtained by using the facilities in Vitro Corporation's Parallel Processing Laboratory. The author thanks Dr. Frederick Menkello and Ms. Rochelle Payzer for their assistance and encouragement.

## REFERENCES

1. A. N. Kolmogorov, *Dokl. Akad. Nauk SSSR* **119**, 861 (1958) [Russian]; English summary, *Math. Rev.* **21**, 386 (1960).
2. Ya. Sinai, *Dokl. Akad. Nauk SSSR* **124**, 768 (1959) [Russian]; English summary, *Math. Rev.* **21**, 386 (1960).
3. J. H. Curry, *J. Stat. Phys.* **26**, 683 (1979).
4. P. Grassberger and I. Procaccia, *Physica D* **13**, 34 (1984).
5. J. P. Crutchfield and N. H. Packard, *Physica D* **7**, 201 (1983).
6. I. Shimada, *Progr. Theor. Phys.* **62**, 61 (1979).
7. C. E. Shannon, *Bell Syst. Tech. J.* (1948); in *The Mathematical Theory of Communication* (Univ. of Illinois Press, Urbana, IL, 1963).
8. A. M. Yaglom, *An Introduction to the Theory of Stationary Random Functions*, transl. by R. A. Silverman (Printice–Hall, Englewood Cliffs, NJ, 1962).
9. J.-P. Eckmann and D. Ruelle, *Rev. Mod. Phys.* **57**, 617 (1985).
10. L.-S. Young, *Physica A* **124**, 639 (1984).
11. J. G. Caputo and P. Atten, *Phys. Rev. A* **35**, 1311 (1987).
12. S. Smale, *Bull. Am. Math. Soc.* **73**, 747 (1985).
13. A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, *Physica D* **16**, 285 (1985).
14. M. Henon, *Commun. Math. Phys.* **50**, 69 (1976).
15. K. Ikeda, *Opt. Commun.* **30**, 257 (1979).
16. O. E. Rossler, *Phys. Lett. A* **57**, 397 (1976).
17. E. Lorenz, *J. Atmos. Sci.* **20**, 130 (1963).
18. M. C. Mackey and L. Glass, *Science* **197**, 287 (1977).

19. A. Fraser, Ph.D. dissertation, University of Texas at Austin, May 1988.

20. A. Fraser and H. Swinney, *Phys. Rev. A* **33**, 1134 (1986).

21. The author has since extended this work to box counting dimension calculations. After the completion of this work, the author became aware of a paper, L. S. Liebovitch and T. Toth, *Phys. Lett. A* **141**, 386 (1989), which uses a scheme similar to the partitioning scheme developed in Section III.A to calculate the box counting dimension, although it does not develop a generalized compactification scheme as is presented in Section III.B of this paper. See also the paper by Hunt and Sullivan in the conference proceedings edited by G. Mayer-Kress, *Dimensions and Entropies in Chaotic Systems* (Springer-Verlag, New York, 1986), where they use Monte Carlo integration to calculate fractal dimensions efficiently.

22. K. M. Short, Vitro Technical Report 08281-1/1990.

23. K. M. Short, Vitro Technical Report 08281-2/1991.