Efficient algorithm for estimating the correlation dimension from a set of discrete points

James Theiler
Department of Physics 103-33, California Institute of Technology, Pasadena, California 91125
(Received 27 March 1987)

We present an algorithm which computes the standard Grassberger-Procaccia correlation dimension of a strange attractor from a finite sample of N points on the attractor. The usual algorithm involves measuring the distances between all pairs of points on the attractor, but then discarding those distances greater than some cutoff r_0. Our idea is to avoid computing those larger distances altogether. This is done with a spatial grid of boxes (each of size r_0) into which the points are organized. By computing distances between pairs of points only if those points are in the same or in adjacent boxes, we get all the distances less than r_0 and avoid computing many of the larger distances. The execution time for the algorithm depends on the choice of r_0, the smaller r_0, the fewer distances to calculate, and in general the shorter the run time. The minimum time scales as $O(N\log N)$; this compares with the $O(N^2)$ time that is usually required. Using this algorithm, we have obtained speedup factors of up to a thousand over the usual method.

I. INTRODUCTION

It is known that complex aperiodic behavior can result from deterministic physical systems with few degrees of freedom.[1,2] Further, dissipative dynamical systems which may have *many* degrees of freedom (such as fluids) can—after an initial transient time—settle down to a state in which only a few degrees of freedom are relevant to the dynamics.[3] In this post-transient state, the system's trajectory through phase space is confined to a low-dimensional subset of the available phase space. When the subset is a strange attractor, motion is complex aperiodic and typically chaotic.[4] On the other hand, the motion of a dynamical system may be complicated just because the system itself is complicated; a full description would require many degrees of freedom. Here a stochastic analysis is usually more appropriate.

An experimentalist, observing a system that displays apparently erratic motion, seeks to distinguish between these two kinds of motion, deterministic "chaos" and stochastic "noise."[5] Does the system have a (low-dimensional) strange attractor in its phase space, and if it does, what is its dimension?

The first dimension algorithms were based on a box-counting principle,[6,7] though this was soon judged to be computationally inefficient.[8] Algorithms based on distances between pairs of points were introduced,[9,10] and of these the correlation dimension of Grassberger and Procaccia[11] and Takens[12] is among the most widely used. There are a number of limitations and potential pitfalls with the correlation algorithm and many of these we leave to be discussed elsewhere,[2,3,11,13–15] but one problem has always been that the computation can be very time consuming.

We address the issue of computational efficiency with a new algorithm: box-assisted correlation. By recognizing that it is the shortest distances which are important, and by devising a way of finding those short distances without computing all of them, we have developed an algorithm for estimating correlation dimension which is much faster than the standard algorithm. We hope that this will allow the numerical estimation of dimension to become a more widespread test of experimental data.

In Sec. II we introduce the correlation dimension and its definition in terms of the numerical problem that is to be solved; namely, we estimate the dimension of an attractor $\mathcal{A}$ from a finite sample of discrete points $\{x_1,x_2,\ldots,x_N\}$ on the attractor. In Sec. III the basic ideas behind our algorithm are motivated and introduced. Section IV provides a theoretical evaluation of the algorithm's performance over a range of input parameters (such as box size) and shows that for well-chosen values $O(N\log N)$ computation time can be achieved. Section V discusses the program itself and its performance on a real computer, and demonstrates its speedup with the example of the Hénon attractor. Finally, in the Appendix, we introduce "prism-assisted correlation," a generalization of the "box-assisted correlation" algorithm which is more effective at large embedding dimension.

II. CORRELATION DIMENSION

We wish to estimate the dimension of an attractor $\mathcal{A}$ which is embedded[6,16] in an m-dimensional Euclidean space from a sample of N points on the attractor, that is, from the set $\{x_1,x_2,\ldots,x_N\}$ with $x_i\in\mathcal{A}\subset\mathbb{R}^m$. Grassberger and Procaccia[11] suggest that we measure the distance between every pair of points and then compute the correlation integral

$$C(N,r)=\frac{2}{N(N-1)}\sum_{\substack{i,j\\(1\le i<j\le N)}}H(r-\|x_i-x_j\|)\ , \quad (1)$$

where $H(x)$ is the Heaviside step function. The summation counts the number of pairs (x_i,x_j) for which the

distance $\|x_i - x_j\|$ is less than r.

For a set such as a strange attractor with dimension ν, the correlation integral scales like r^ν for small r; in particular, we expect for large N and small r that

$$C(N,r) \approx (r/R)^\nu , \tag{2}$$

where the constant of proportionality R provides a convenient measure for the "effective radius" of the attractor. To find this ν, Grassberger and Procaccia prescribe that a log-log plot of $C(N,r)$ versus r be constructed and that the dimension be read off as the slope of the curve over some range $r \leq r_0$ (see Fig. 1).

A variant of this method, suggested by Takens,[17] involves computing the logarithm of every distance $r \leq r_0$, taking the average

$$\alpha = \langle \log(r/r_0) \rangle , \tag{3}$$

and finally setting $\nu = -1/\alpha$. This, Takens shows, is a maximum likelihood estimate for the correlation dimension. In both cases, finding α or finding $C(N,r)$ all of the $O(N^2)$ distances between each of the pairs of points must be computed, even though only those for which $r \leq r_0$ are actually used. Because there are so many distances, the operations performed in determining them and in subsequently updating α or $C(N,r)$ dominate the computational workload in estimating correlation dimension.

III. THE BOX-ASSISTED CORRELATION ALGORITHM

In this paper, we propose that only the small distances ought to be computed. We introduce an algorithm which computes *all* distances $r \leq r_0$ and *some* (*but not all*) of the distances $r > r_0$. It is by *not* computing all those extra distances that this algorithm is able to achieve its advantage.

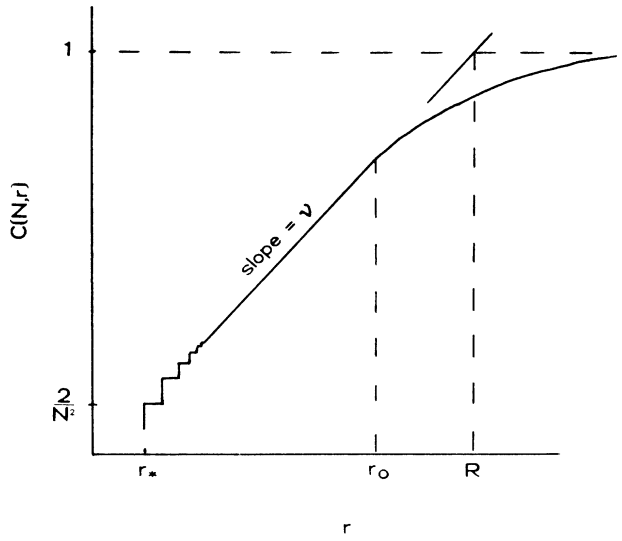

FIG. 1. Correlation integral for a typical strange attractor of dimension ν (axes are logarithmically scaled). Here, r_* is the smallest of the $\frac{1}{2}N^2$ distances, R is the "effective radius" of the attractor, and $r \leq r_0$ is the scaling region, over which $C(N,r) \approx (r/R)^\nu$.

Indeed, we find that for appropriately chosen r_0, we can find the smallest $O(N)$ distances in $O(N \log N)$ time.[18] This can be dramatically faster than the $O(N^2)$ time that is usually required. We present an example below with $N = 64\,000$ points that can be implemented on a personal computer; our algorithm cuts the computation time by a factor of over 1000.

In our procedure, points are distributed into m-dimensional "boxes" of size r_0. Then, rather than compute distances between *every* pair of points, we only compute distances between points which are either in the same box or else are in neighboring boxes. This way, we get all of the distances in the range $0 \leq r \leq r_0$. In the process, we also compute a few extra distances in the range $r_0 < r \leq 2r_0$ (Ref. 19) which are discarded (see Fig. 2).

There is, to be sure, an inefficiency in these discarded distances, but it is no more inefficient than the standard algorithm which computes and discards *all* distances $r > r_0$. On the other hand, there is a certain amount of "overhead" with the box-assisted correlation algorithm: One must keep track of which points are in which box and which boxes are neighbors of each other.

The primary extra work, we find, comes from searching for neighbors of the boxes. When we set up our grid

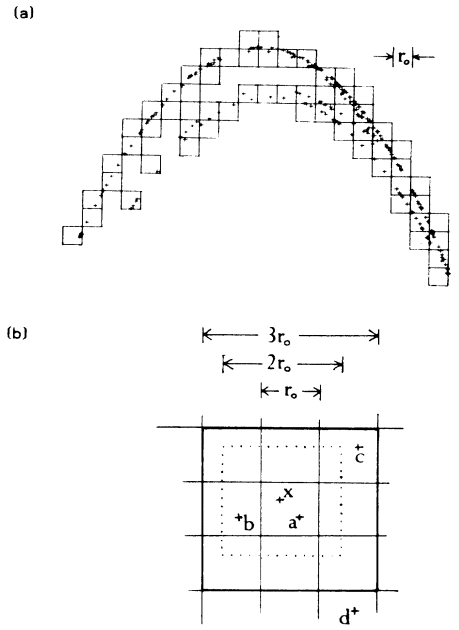

FIG. 2. (a) A square grid of boxes of width r_0 is placed over the points of the attractor. (b) Here, distances are computed from point x to other points on the attractor. Distances are computed to points which are in the same (e.g., point a) or in adjacent boxes (e.g., points b,c). Any of these distances which are greater than r_0 (such as to point c) are discarded. Points (e.g., point d) not in adjacent boxes are not considered. In other words, distances to points (a,b,c, not d) inside the bold box are computed, but only those (a,b, not c) inside the dotted box are used.

of boxes we do not actually provide a separate memory location of each box. If we did, then we would find, for a typical attractor of dimension $\nu < m$, that *most* of the boxes would be empty. Instead, we have what amounts to a list of box positions. From the point of view of computer storage, empty boxes do not exist. Given the position of a box's neighbor, the searching routine must determine whether or not a box exists at that neighboring position, and if it does, which points are inside it.

In our algorithm, we associate a box position with each point. Then the points are sorted lexicographically[20] according to the position of each box. This effectively sorts our list of boxes and at the same time provides a convenient way to determine which points are in a given box. Furthermore, since the boxes are sorted in linear order, a binary search can effectively find which, if any, box is at a given position. The cost of all this extra sorting and searching is only $O(N \log N)$. For small r_0, therefore, the total execution time for computing a correlation dimension can be dramatically reduced.

IV. EVALUATION OF EFFECTIVENESS

In this section, we analyze the efficiency of the box-assisted correlation method and discuss in more detail the various operations which comprise the algorithm's overhead.

A. Run time

We divide time required to execute the box-assisted correlation computation into four components: a reading time, a sorting time, a distance-computation time, and a neighbor-searching time. We write

$$T_{\text{box-assisted}} = T_{\text{read}} + T_{\text{sort}} + T_{\text{dist}} + T_{\text{search}} \ . \tag{4}$$

These correspond more or less to the chronological operations of the algorithm. First, it reads all the points; next, it sorts the points; and finally, for the rest of the time it switches back and forth between searching for neighboring boxes and computing distances between points in those boxes.

The reading time can be written directly

$$T_{\text{read}} = \tau_{\text{read}} N \ , \tag{5}$$

where τ_{read} is the time to read a single point and associate a box location with it. Sorting N points can be done in $O(N \log N)$ time, so we write

$$T_{\text{sort}} = \tau_{\text{sort}} N \log_2 N \ . \tag{6}$$

The time spent computing distances is proportional to the number of distances D that are ultimately computed, that is,

$$T_{\text{dist}} = \tau_{\text{dist}} D \ , \tag{7}$$

which still leaves us to estimate D for this algorithm. Just as the searching routine is the most complicated part of the algorithm, estimation of T_{search} is the most difficult. For now we will write

$$T_{\text{search}} = \tau_{\text{search}} SB \ , \tag{8}$$

where B is the number of nonempty boxes and the variable S—into which all of the complication is incorporated—is the average number of "search steps" per box.

In practice it is these last two terms, the distance computations and the neighbor searches, which take all the time. Except for very-low-dimensional systems ($m \leq 2$) and/or very small boxes $r_0 \to 0$, the reading and sorting times are comparatively negligible.

In the standard algorithm, by contrast, there is no searching and sorting, but *all* of the distances are computed, so that

$$T_{\text{standard}} = \tau_{\text{read}} N + \tau_{\text{dist}} \tfrac{1}{2} N^2 \ . \tag{9}$$

As long as $D \ll \frac{1}{2} N^2$ and the searching term does not dominate [the sorting time *cannot* dominate, since it is manifestly $O(N \log N)$], the running time for the box-assisted correlation algorithm will be much less than that for the standard algorithm.

The coefficients τ_{read}, τ_{sort}, τ_{dist}, and τ_{search} are machine dependent,[21] though they are all of essentially the same magnitude. τ_{dist} increases directly with embedding dimension m; τ_{sort} and τ_{search} increase more or less linearly with m but level off for large m. For our program, points are read in directly as a time series and then embedded[16] into $\mathbb{R}^m$, so that τ_{read} is independent of m.

B. Choice of box size

The box size r_0 is the only parameter over which the user has full control in the box-assisted correlation algorithm. How D, B, and especially S depend upon r_0 is nontrivial, though we will make some estimates below.

Two attitudes can be taken toward optimum choice of box size r_0. We might for instance specify beforehand that we want all distances less than that value of r_0 which determines the scaling region. Arguing that the more distances the better the statistics,[22] we say that we want to compute as many distances as possible and exclude only those beyond which the r^ν scaling fails.

The second approach chooses r_0 so that $O(N)$ of the $\frac{1}{2} N^2$ distances is less than r_0. The standard algorithm provides the full $O(N^2)$ range of $C(N,r)$ but it takes a time which is also $O(N^2)$. Since it is a log-log plot of $C(N,r)$ versus r that will ultimately be constructed, we may want to optimize the logarithmic range obtained per unit of computing time, that is, $(\log D_*)/T$, where D_* is the number of distances less than r_0 and T is the time it takes to do the computation. We will later see that

$$T = O(D_*) + O(N \log N) \ . \tag{10}$$

Thus, choosing $D_* = O(N)$ optimizes $(\log D_*)/T$.

Now we can estimate $C(N,r)$ with, for instance, $C(\sqrt{N}, r)$; this will cost $O(N)$ and give a range in the correlation integral of $O(N)$—in particular, this estimates $C(N,r)$ for the larger distances. If we can compute the smallest $O(N)$ distances cheaply, in $O(N \log N)$ time, say, then we will have obtained an $O(N)$ range for the $C(N,r)$ curve which is distinct from the large-

distance $O(N)$ range. These two ranges may then be "pasted together" on logarithmic axes, providing $O(N^2)$ range with significantly less than $O(N^2)$ work.

To find r_0 so that N distances are less than r_0, we invoke the definition of the correlation integral to get the implicit equation

$$N=\tfrac{1}{2}N^2C(N,r_0)\ , \tag{11}$$

which, when we apply the approximation $C(N,r_0)\approx(r_0/R)^\nu$, we can invert for r_0 to obtain

$$r_0=R\,(2/N)^{1/\nu}\ . \tag{12}$$

This may seem a bit circular, defining r_0 in terms of ν, the quantity we are ultimately after, but rough estimates of ν and R are usually available, and at any rate can be estimated from $C(\sqrt{N},r)$.

C. Number of distances

Of the D distances we compute, the "desirable" distances are those less than r_0. We can write this number down exactly, in terms of the correlation integral

$$D_*=\tfrac{1}{2}N^2C(N,r_0)\approx\tfrac{1}{2}N^2(r_0/R)^\nu\ . \tag{13}$$

Now the actual D includes some distances that are greater than r_0, and so will be larger than this. We estimate the ratio of desirable distances to the total D by considering distances from a single (typical) point (see Fig. 2). The number of desirable distances measured *from* this particular point will be proportional to $(2r_0)^\nu$ since a hypercube with diameter $2r_0$ centered on the particular point will contain all those points to which the distance is less than r_0. By the same token, a hypercube of diameter $3r_0$ centered not at the particular point but at the center of the box in which the particular point resides will contain all the points *to* which distances are measured. Hence the ratio of desirable distances to total distances computed will be $(2/3)^\nu$, and

$$D=\tfrac{1}{2}N^2(3r_0/2R)^\nu\ . \tag{14}$$

For $r_0\gtrsim 2R/3$, this equation overestimates D; the number of distances calculated is never larger than $\frac{1}{2}N^2$. It follows that the time spent computing distances is

$$T_{\rm dist}=\tau_{\rm dist}D=(3/2)^\nu D_{r\le r_0}\ , \tag{15}$$

which for fixed ν varies linearly with the number of desirable distances. Should be desire $O(N)$ distances, $T_{\rm dist}$ will be $O(N)$ as long as the coefficient $(3/2)^\nu\ll N$. That is,

$$\nu<\log_{3/2}N=1.7\log_2 N\ . \tag{16}$$

In fact, as we will later see, it is not the extra distances but the increased search time that limits how large a dimension we are able to efficiently compute with the box-assisted correlation algorithm.

D. Number of boxes

If R is the "radius" of the attractor, then $(2R/r_0)$ estimates the number of boxes along any one dimension, and $(2R/r_0)^\nu$ (Ref. 23) is the number of boxes expected to cover the attractor. Now if r_0 is sufficiently large that there are many points *per* box, then $(2R/r_0)^\nu$ is a reasonable estimate of B. On the other hand, since the number of points N is finite, we expect that as r_0 decreases more and more of these covering boxes will be empty. In particular, as the boxes become even tinier, the points will eventually come to be individually wrapped—a separate box for each point. That is,

$$\lim_{r_0\to 0}B=N\ . \tag{17}$$

If we model the distribution of points among the available boxes with the Poisson formula,[24] then the number of nonempty boxes is

$$B=(2R/r_0)^\nu(1-e^{-N(r_0/2R)^\nu})\ . \tag{18}$$

E. Number of neighbor searches

Unlike D and B, which depend on the geometry of the attractor S the average number of searching steps from each box depends on the cleverness of our search strategy. Although this makes it difficult to give a good general estimate for S in terms of the other parameters, we can provide some upper bounds. We will provide two upper bounds in particular, each associated with a different strategy for neighbor searching. Our algorithm uses a hybrid of these two strategies, so both bounds apply. In many cases, it turns out that the actual S is much less than both bounds.

From every box, we can search each of the neighboring positions to see if there is a nonempty box at that position. Since the boxes are sorted, each search can be done in $\log_2 B$ steps. If $(b_1,\ldots,b_m)$ is the position of the "from" box, then the positions of the "to" boxes will be of the form $(b_1+\Delta b_1,\ldots,b_m+\Delta b_m)$, where $\Delta b_i\in\{-1,0,1\}$ for $i=1,\ldots,m$. Thus, there are 3^m "to" positions. We can write

$$S\le\frac{3^m-1}{2}\log_2 B\ , \tag{19}$$

where the -1 is to exclude the case $\Delta b_1=\Delta b_2=\cdots=\Delta b_m=0$ (the "to" box is the same as the "from" box) and the factor of 2 stems from the symmetry of distance: $d(A,B)\equiv d(B,A)$. Having found all the distances from points in box no. 1 to those in box no. 2, we need not compute distances from points in box no. 2 to those in box no. 1.

For intermediate numbers of boxes, $3^m<B\ll N$, and with $\nu\sim m$ Eq. (19) not only bounds but reasonably estimates S. For low-dimensional attractors $\nu\ll m$, a box typically will have many empty neighbors and the effective S will be much lower than this bound.

The alternative strategy is to go through the list of boxes one at a time and determine whether or not each is a neighbor. There are $B/2$ such candidates (with the factor of 2 arising as above), and the binary search is avoided, so

$$S\le\tfrac{1}{2}B\ . \tag{20}$$

This second bound provides a reasonable estimate of S only when B is very small or when v is very large (see Appendix).

We can use these bounds on S to bound the search time. Using $B \leq N$ and Eq. (19), we have

$$T_{\text{search}} \leq \tau_{\text{search}} \frac{3^m - 1}{2} N \log_2 N \ , \tag{21}$$

which formally is $O(N \log N)$. We note, however, that this "order" is sensible only if the coefficient is not too large; this is for $3^m \ll N$ or

$$m \ll \log_3 N = 1.6 \log_2 N \ . \tag{22}$$

In practice, we find that the search time begins to overrun the total execution time of the standard algorithm at $m \approx 0.75 \log_2 N$. From the second bound in Eq. (20), we have

$$T_{\text{search}} \leq \tau_{\text{search}} \tfrac{1}{2} N^2 \ , \tag{23}$$

which shows that in the worst case the search time is $O(N^2)$. In this case, the total execution time for the box-assisted correlation method may exceed that of the standard method—this certainly is a case to be avoided. But at least the search time is never any worse than $O(N^2)$. No matter how large m or how poorly chosen r_0, the search time will never be atrociously longer than the total execution time for the standard algorithm.

F. Large and small box-size limits

Our bound in Eq. (20) tells us that the search time is negligible if r_0 is so large that all the points fall into a single box. In that case all of the distances are computed, and all of the computation goes into computing distances. In other words, the $r_0 \gg R$ limit of box-assisted correlation does exactly what the standard algorithm does and does it for essentially the same computational cost.

We consider also the limit $r_0 \to 0$. In this case, $B \to N$, and although formally both bounds on the search time are at their maximum here, the boxes are becoming sparser and more isolated from each other. Eventually none of the boxes have any neighbors and the actual search time plummets to as low as $\tau_{\text{search}} N$. In this limit we also have $D \to 0$, so the total execution time is very small. Of course, with no distances computed, not much is learned about the attractor (this much is learned: that the smallest distance is greater than r_0) so that there is not much practical benefit in this limit.

V. IMPLEMENTATION

A program[25] to implement this algorithm has been written in the *C* language and tested on an IBM PC running at 4.77 MHz with 640K RAM. We find that memory limitations[26] (not time constraints) prevent us from processing a time series of more than 64 000 points.

As an example, we compute the dimension of the Hénon attractor[27] from a sample of $N = 64\,000$ points. Using $r_0 = 0.0005$ and $m = 2$, the whole computation takes about 36 min. Only four of those minutes are actually spent computing distances (and of the 1.88×10^5 distances that computed, only 1.15×10^5 are actually used); the rest of the time is spent "setting up." It takes 7 min to read in and box the points, 15 min to sort the points, and 10 min to search for neighboring boxes. However inefficient this seems at first, it is still dramatically faster than the standard approach of computing all $\frac{1}{2}N^2 = 2.05 \times 10^9$ distances, which on our PC would take over 30 days.

This choice of $r_0 = 0.0005$ is much smaller than what might conventionally be considered the scaling regime, but it enables us to get the shortest $O(N)$ distances computed and tabulated. As a separate computation, we can take a smaller sample of $O(\sqrt{N})$ points and get an estimate for what the "rest" of the correlation curve looks like, and again this will only take $O(N)$ time. What we end up with, in this case, is the full $O(N^2)$ dynamic range in $C(N,r)$ computed with $O(N \log N)$ work (see Fig. 3).

ACKNOWLEDGMENTS

The author gratefully acknowledges Mark Muldoon and Dr. Noel R. Corngold for their helpful suggestions and wise counsel.

APPENDIX: "PRISM-ASSISTED CORRELATION"

Because there are so many potential neighbors ($\sim 3^m$) in systems with large embedding dimension, we find that the search time increases very rapidly [though not quite exponentially—it is limited ultimately by the bound in Eq. (22)] with increasing m. Empirically, we find with random data[28] that the box-assisted algorithm becomes

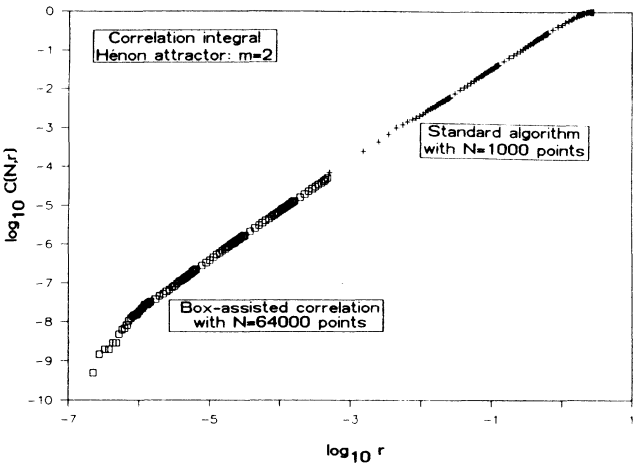

FIG. 3. Log-log plot of the correlation integral for the Hénon attractor. The small distances in the lower half (□) of the curve were computed with $N = 64\,000$ points using a box size of $r_0 = 0.0005$. The upper half (+) was computed with a much smaller sample of $N = 1000$ points using a box size so large ($r_0 = 3.0$) that all distances are computed. Both computations together took less than an hour on a personal computer. To compute the entire curve in one piece with the standard algorithm would have taken over a month.

worse than the standard algorithm for embedding dimension larger than $m \approx 0.75 \log_2 N$.

Recently, however, we have devised a variant of this algorithm which gets around this large-m limitation. In the m-dimensional space we impose a b-dimensional grid where b is less than m. The "boxes" in this space are m-dimensional rectangular prisms with b short sides of length r_0 and $m-b$ long sides which extend the entire length of the attractor. As before, we compute distances between pairs of points only if those points are in the same or in adjoining prisms. Note that $b=m$ is just the regular box-assisted algorithm and $b=0$ corresponds to the standard algorithm.

With $b<m$, there are fewer neighbors and the coefficient of the search time looks like 3^b instead of 3^m. We can take m as large as we like and the search time will not increase. On the other hand, for fixed m, a smaller value of b means more distances are computed. This is because of the distances we have to compute between points at opposite "ends" of these long prisms. Following Eq. (13), we have

$$D_{\text{prism}} = \tfrac{1}{2} N^2 (3r_0/2R)^b \tag{24}$$

distances to compute. Let us take r_0, as usual, so that $O(N)$ desirable distances are computed. We substitute Eq. (11) into (24) to get

$$D_{\text{prism}} = \tfrac{1}{2} N^2 [\tfrac{2}{3}(N/2)^{1/\nu}]^{-b} \tag{25}$$

In choosing the best value for b we have these two competing effects. The number of distances computed *decreases* with increasing b, and the number of neighbors the program has to search for *increases*. The sum of the two times we can estimate as

$$T_{\text{prism}} = \tau_{\text{dist}} \tfrac{1}{2} N^2 [\tfrac{2}{3}(N/2)^{1/\nu}]^{-b} + \tau_{\text{search}} \frac{3^b - 1}{2} B \log_2 B \ , \tag{26}$$

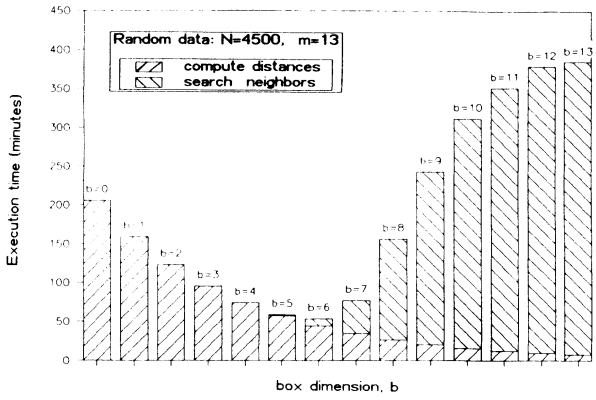

FIG. 4. Prism-assisted correlation for $N=4500$ points of random data (uniformly distributed over $[-1,1]^m \subset \mathbb{R}^m$) embedded in $m=13$ space with $r_0=0.66$ chosen so that ~ 4500 "desirable" distances are computed. Taking $b=m$ corresponds to the usual box-assisted algorithm and $b=0$ is the standard unassisted algorithm. Although this is a case where the standard ($b=0$) algorithm is better than the box-assisted ($b=m$) method, we find that we do get improvement from the "prism-assisted" correlation algorithm with $0<b\leq 8$ and that the best performance is achieved at $b=6\approx 0.5\log_2 4500$. The contribution to total execution time due to initial reading and sorting is about 1 min.

noting that the term for search time is valid only for $b<\nu$. Formally, we can optimize by setting $\partial T/\partial b = 0$. The resulting expression is quite unwieldy, but in the limit of large N and large ν ($\gg \log N$), we have

$$b = \frac{\log N}{\log(9/2)} \approx 0.5 \log_2 N \ . \tag{27}$$

Numerical experiments with random data confirm this estimate (see Fig. 4).

[1]This was first shown by E. N. Lorenz, J. Atmos. Sci. **20**, 130 (1963).

[2]For a comprehensive review see J.-P. Eckmann and D. Ruelle, Rev. Mod. Phys. **57**, 617 (1985).

[3]For a recent example see A. Brandstater and H. L. Swinney, Phys. Rev. A **35**, 2207 (1987).

[4]"Chaotic" is defined by the property of nearby trajectories to diverge. E. Ott, Rev. Mod. Phys. **53**, 655 (1981).

[5]A method of distinguishing stochastic and (infinitely differentiable) deterministic time series directly from the power spectrum is discussed by D. Sigeti and W. Horsthemke, Phys. Rev. A **35**, 2276 (1987). However, it involves the asymptotic high-frequency behavior and this may not be available in a discrete time series. Also, it does not measure the number of degrees of freedom in a deterministic system.

[6]N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, Phys. Rev. Lett. **45**, 712 (1980).

[7]D. A. Russel, J. D. Hanson, and E. Ott, Phys. Rev. Lett. **45**, 1175 (1980).

[8]H. S. Greenside, A. Wolf, J. Swift, and T. Pignataro, Phys. Rev. A **25**, 3453 (1982).

[9]Y. Termonia and Z. Alexandrowicz, Phys. Rev. Lett. **51**, 1265 (1983).

[10]P. Grassberger and I. Procaccia, Phys. Rev. Lett. **50**, 346 (1983).

[11]A fuller exposition is given by P. Grassberger and I. Procaccia, Physica **9D**, 189 (1983).

[12]The correlation dimension was discussed independently by F. Takens, Invariants Related to Dimension and Entropy, in Atas do 13°. Colóqkio Brasiliero de Matemática, Rio de Janeiro, 1983 (unpublished).

[13]J. Theiler, Phys. Rev. A **34**, 2427 (1986).

[14]J. W. Havstad and C. L. Ehlers (unpublished).

[15]J. Theiler, Ph.D. thesis, California Institute of Technology, 1987.

[16]F. Takens, in *Detecting Strange Attractors in Turbulence,* Vol. 898 of *Lecture Notes in Mathematics,* edited by D. A. Rand

and L.-S. Young (Springer-Verlag, Berlin, 1981).

[17]F. Takens, in *On the Numerical Determination of the Dimension of an Attractor,* Vol. 1125 of *Lecture Notes in Mathematics,* edited by B. L. J. Braaksma, H. W. Broer, and F. Takens (Springer-Verlag, Berlin, 1985).

[18]A general discussion of related problems from a computer science viewpoint can be found in J. L. Bentley, Commun. ACM **23**, 214 (1980). We are indebted to J. D. Farmer for this reference.

[19]This assumes the L_∞ or "maximum" metric. For the L_1 or "taxicab" metric, the appropriate inequality is $r \leq 2mr_0$; for the L_2 or Euclidean metric, it is $r \leq 2\sqrt{m}\, r_0$.

[20]In the "lexicographic" ordering, we say that $\mathbf{a} < \mathbf{b}$, where in coordinate notation $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_m)$, if and only if $a_k < b_k$ for some $k \leq m$, and $a_i = b_i \forall i < k$.

[21]On our IBM PC, we find $\tau_{\text{dist}} = (0.5 + 0.2m)$ msec. In the Takens method, we have also to take a logarithm of all distances less than r_0; these cost 4.3 msec each (with an 8087 floating point co-processor). Also $\tau_{\text{search}} \approx (1.5 + 0.6m)$ msec and $\tau_{\text{sort}} = (0.44 + 0.09m)$ msec, with some leveling off at large m. Finally, $\tau_{\text{read}} = 6.7$ msec. There is a small memory compiler option which cuts these times in half, but it can only be used for $N < 5000$.

[22]Takens (in Ref. 17) says the error bar on ν will scale as $1/\sqrt{D_*}$, where D_* is the number of distances less than r_0.

[23]Properly we should write $B \approx (2R/r_0)^d$, where d is the "capacity" of the set. Though capacity and correlation dimension are not precisely the same thing, our approximations do not distinguish between them.

[24]What we really assume is that points are distributed uniformly among those boxes which cover the attractor; this presumes not only that the distribution of points is uniform over the attractor, but that the intersection of the attractor with boxes is uniform—in fact, there are often a lot of "clipped edges." For details of this effect in another context, see W. E. Caswell and J. A. Yorke, in *Dimensions and Entropies in Chaotic Systems,* Vol. 32 of *Springer Series in Synergetics,* edited by G. Mayer-Kess (Springer-Verlag, Berlin, 1986), p. 123.

[25]The source code, further documentation, and executable files which run on an IBM PC (or compatible) are available from the author.

[26]In our Grassberger-Procaccia routine, we use $8N + 4xr_0$ bytes, where x is the "expansion factor" which is multiplied by the floating point input before discretizing into integers. In our Takens maximum likelihood routine, we use $14N$ bytes since the floating point input is stored in double precision.

[27]M. Hénon, Commun. Math. Phys. **50**, 69 (1976). The mapping is $x_{i+1} = y_i + 1 - ax_i^2$; $y_{i+1} = bx_i$. Following Hénon, we use $a = 1.4$, $b = 0.3$.

[28]For low-dimensional attractor data, we find that the search time increases so slowly with m that $O(N)$ distances can *always* be computed faster with the box-assisted algorithm than with the standard algorithm. But even in these cases, we find that the "prism-assisted" variant provides still further improvement.