Adaptive box-assisted algorithm for correlation-dimension estimation

Angelo Corana
Institute for Electronic Circuits, National Research Council, Via De Marini 6, 16149 Genova, Italy
(Received 27 October 1999; revised manuscript received 16 June 2000)

An algorithm is presented for efficient computation of the correlation dimension from a time series. The main feature of the algorithm is the use of a variable number of points in order to keep the number of close pairs approximately constant at the various scales and at the various embedding dimensions. The procedure consists of a number of steps with decreasing cutoff distance; at each step only neighboring pairs are considered, using a box-assisted approach. The algorithm is tested by performing some trials on time series from known model attractors. With respect to the standard algorithm, the one proposed here yields more uniform precision in the various correlation integral values, improving the statistics at the smallest distances. Moreover, it gives a substantial reduction in computation time, allowing execution of trials with a very large number of points, and exploitation of shorter length scales. The algorithm can be easily adapted for the computation of q-order generalized dimensions.

PACS number(s): 05.45.Tp, 07.05.Kf, 89.80.+h

I. INTRODUCTION

The correlation dimension (D_2), obtained through the computation of the correlation integral, has been since its introduction in 1983 [1,2] a widely used parameter in nonlinear time series analysis. The procedure needed for the estimation is simple, since it consists in computing and histogramming distances within a set of points embedded in R^m, but there are a number of aspects that must be considered in order to obtain a reliable estimate [3].

Errors can be divided into two classes [4]: statistical errors, which become more important at the smallest scales, and systematic errors (e.g., the edge effects). Related issues are the influence of the number of points, the influence of noise, the need for reliability indices, and the need for fully automatic procedures.

The difficulty of obtaining a good D_2 estimate increases as the D_2 value itself increases. Specifically, most authors believe that it is essentially impossible to estimate D_2 in the case of high-dimensional systems (i.e., when $D_2 \geqslant 5-6$). Another important aspect is that the naive algorithm, which takes into account all distances, is computationally heavy. In fact, the computation of the correlation integral from a set of N points is $O(N^2)$ for each embedding dimension m, and several correlation integrals, for increasing m values, must be evaluated.

The first and more straightforward improvement is the computation of the correlation integrals for all the embeddings $m \leqslant m_{\max}$ as a by-product of the computation with $m = m_{\max}$ [5,6]. In addition, the basic version has been optimized for various computer architectures, including superscalar (pipelined) processors [6,7] and parallel systems [8,9]. Another way to reduce the time complexity of the algorithm is to compute not all distances but only those less than a given cutoff value using various techniques for the fast search of neighboring points, like m-dimensional trees [10] or meshes of boxes [11,5]. However, such algorithms do not yield the complete correlation integral, which can be useful when dealing with experimental signals. Moreover, in some cases the choice of the cutoff distance is not straightforward. In any case, with the usual approach the correlation integrals (for all the length scales or for those less than the cutoff distance) are computed with a fixed number of points, so the statistics worsen at the shortest scales.

To obtain the whole correlation integral keeping computation time low, Theiler proposed a double computation: one using the whole point set with a small cutoff distance and one using a small subset of points and considering all distances [11]. In this work we propose an adaptive box-assisted (ABA) algorithm for fast computation of the correlation integral, which is performed using a different number of points for each length scale and for each embedding m. The procedure consists of a number of steps, with a decreasing cutoff distance, starting from the maximum value and reaching the minimum scale at which we intend to operate. At each step the box-assisted approach allows an efficient search of candidate pairs for distance computation. As the search becomes more local and therefore faster, the number of points increases to assure good statistics in any situation.

An estimation of the computational cost is given in terms of the number of candidate pairs for the various values of the cutoff distance. Some trials have been performed on time series from known model attractors to compare the proposed algorithm with the standard ones, taking into account both the accuracy of correlation integral values and the computational cost.

II. THE CORRELATION DIMENSION

A. The basic procedure

Let us consider a set of N points ($\mathbf{x}_i, i=1, \ldots, N$), reconstructed in R^m with the time-delay method [2,6] from a scalar time series. The correlation integral is defined as

$$C_m(\epsilon) = \lim_{N \to \infty} C_m(N, \epsilon), \tag{1}$$

where

$$C_m(N, \epsilon) = \frac{\mathcal{N}_m(N, \epsilon)}{\mathcal{N}_t(N)}, \tag{2}$$

 ©2000 The American Physical Society

$$\mathcal{N}_m(N,\epsilon)=\{\text{no. of pairs } (i,j), i>j : \|\mathbf{x}_i-\mathbf{x}_j\|<\epsilon\} \quad (3)$$

and

$$\mathcal{N}_t(N)=\frac{N(N-1)}{2}=\{\text{total number of pairs}\}. \quad (4)$$

$\|\cdot\|$ denotes a norm in R^m. $C_m(\epsilon)$ gives the probability of a randomly chosen pair (with respect to the natural measure of the attractor) in the m-dimensional space having a separation less than ϵ.

It is advisable to consider in the evaluation of the correlation integral only pairs that are close in space but not in time [4,3]. It suffices to require $i-j>h$ in Eq. (3), h being a suitable integer >1, and to modify $\mathcal{N}_t(N)$ accordingly.

Provided that m is sufficiently high to allow a good reconstruction of the attractor, if $\epsilon\to 0$, $C_m(\epsilon)$ varies as a power of ϵ, the exponent being the correlation dimension [1,2],

$$C_m(\epsilon)\propto\epsilon^{D_2}. \quad (5)$$

Having N points available, we compute the sample correlation integral $C_m(N,\epsilon)$, usually by chosing a suitable number of ϵ values and building the cumulative histogram of distances between all pairs of points. $C_m(N,\epsilon)$ increases monotonically from $2/N(N-1)$ up to the saturation value 1. The error with respect to the ''true'' correlation integral increases as ϵ decreases and m increases, since the statistics worsen [i.e., $\mathcal{N}_m(N,\epsilon)$ becomes too low].

Since with a limited data set we cannot explore very small scales, the usual procedure searches for a range of intermediate ϵ values $\epsilon'\leqslant\epsilon\leqslant\epsilon''$ (*scaling region*) for which relation (5) approximately holds. $\log C_m(N,\epsilon)$ versus $\log\epsilon$ is plotted for increasing embeddings (e.g., from 2 up to $m_{\max}$) and the slope (smoothed local derivative) in the scaling region is fitted. If the slope increases with m up to a saturation value (plateau), this value is assumed as the estimation of the true D_2 [12]. Various criteria can be employed to extract the best D_2 estimate and its degree of reliability from the log-log plot [13,14].

The scaling behavior disappears at short values of ϵ owing to the poor statistics and to the noise that may affect the data, and at high values (saturation region) owing to the edge effects, i.e., the effects of the finite size and global shape of the attractor. The pattern of $\log C_m(\epsilon)$ versus $\log\epsilon$ is sometimes more involved, owing, for example, to the presence of multiple scaling regions.

When m increases interpoint distances also increase. So ϵ' increases, owing to depopulation at the shortest scales, and ϵ'' decreases, since the boundary effects become more important, resulting in a shorter and shorter scaling region [15,16]. This also accounts for failure in D_2 estimation for high-dimensional systems [17,18].

The condition of having a quite well-defined scaling region of sufficient width (e.g., one or half a decade on the logarithmic scale) yields the minimum number of points N, which increases with the D_2 value itself, needed for a reliable computation [19]. To improve the statistics at the smallest scales we are forced to use a high number of points, but in this way computation time increases and for the highest ϵ values we process more data than necessary.

B. Methods with cutoff distance

These methods are based on the observation that it is useless to compute all interpoint distances, since D_2 is obtained in the scaling region, i.e., for distances less than ϵ'' [11,5]. They consider only interpoint distances less than a cutoff value ϵ_c, which must be chosen small enough to achieve a significant reduction in computation time, but well above ϵ'. Neighboring points are found in an efficient way using an additional data structure (e.g., an m-dimensional tree or mesh of boxes).

These methods are interesting, but present some drawbacks: (1) only a portion of the log-log plot for $\epsilon\leqslant\epsilon_c$ is obtained, while in some circumstances the whole log-log plot gives us more insight into the dynamics [20,7]; (2) the proper choice of ϵ_c requires some *a priori* information about the attractor or must be performed by trials; (3) if the time series is noisy, we cannot choose too small a cutoff ϵ_c, thus limiting the efficacy of the method. For these reasons most researchers continue to prefer the naive algorithm, which computes all distances.

In [11] it is suggested to perform two separate computations: the first with the whole set of N points and a small cutoff distance ϵ_c; the second to estimate the correlation integral for $\epsilon>\epsilon_c$, using a lower number of points (e.g., $\sqrt{N}$) and computing all distances. The whole correlation integral is then obtained by joining the two portions.

We point out that using cutoff methods, for a given N, the accuracy of the various values $C_m(N,\epsilon)$ decreases as ϵ decreases and m increases, just as in the standard method. This also happens in each correlation integral portion obtained with the two-computation scheme suggested in [11].

III. THE ADAPTIVE BOX-ASSISTED ALGORITHM

A. The proposed approach

Supposing we can consider, for each ϵ and m, $\mathcal{N}_m(N,\epsilon)$ as a binomial (n,p) random variable with $n=\mathcal{N}_t(N)$ (number of trials) and $p=C_m(\epsilon)$ (probability that the distance between a randomly chosen pair is less than ϵ), we obtain, after $\mathcal{N}_t$ trials, the estimate $p(n)$ of p,

$$p(n)=C_m(N,\epsilon)=\frac{\mathcal{N}_m(N,\epsilon)}{\mathcal{N}_t(N)}, \quad (6)$$

with an actual error $\Delta p=|p(n)-p|$ and a standard error

$$\sigma_p=\left(\frac{p(1-p)}{n}\right)^{1/2}. \quad (7)$$

Since in order to estimate D_2 we plot $\log C_m(\epsilon)$ versus $\log\epsilon$, and since for small Δp

$$\log(p\pm\Delta p)\simeq\log p\pm\frac{\Delta p}{p}, \quad (8)$$

we see that the error on $\log C_m(\epsilon)$ is the relative error on $C_m(\epsilon)$.

Equation (7) shows that with the standard methods, which use a constant number of trials, the relative standard error

σ_p/p increases as p decreases, i.e., as we move toward smaller ϵ values and higher embeddings. On the contrary, we must have

$$n \geqslant \frac{1-p}{\varepsilon^2 p} \tag{9}$$

to keep σ_p/p less than or equal to a prefixed value ε, as p varies. Observing that $(1-p)/p < 1/p$, setting

$$\mathcal{N}_{th} = \frac{1}{\varepsilon^2}, \tag{10}$$

and getting back to the original quantities, we see that the above condition is satisfied if

$$\mathcal{N}_t(N) C_m(\epsilon) = E(\mathcal{N}_m(N,\epsilon)) = \text{const} = \mathcal{N}_{th}, \tag{11}$$

i.e., if the number of trials is inversely proportional to C_m, so that the expected number of successes $\mathcal{N}_m$ is kept constant. Since $\mathcal{N}_t(N) \simeq N^2/2$, Eq. (11) yields

$$N \simeq \left(\frac{2\mathcal{N}_{th}}{C_m(\epsilon)} \right)^{1/2}, \tag{12}$$

showing how the number of points is expected to vary with $C_m(\epsilon)$ if we require that $E(\mathcal{N}_m(N,\epsilon)) = \mathcal{N}_{th}$. Since the variation of $C_m(\epsilon)$ with ϵ is very high [see Eq. (5)], it follows that $\mathcal{N}_t(N)$ or equivalently N must vary greatly in order to satisfy Eqs. (11) and (12).

On the basis of the above considerations, we propose a different procedure, in which each $C_m(\epsilon)$ element is computed with a different number of points in order to keep $\mathcal{N}_m(N,\epsilon)$ approximately equal to $\mathcal{N}_{th}$, ensuring good statistics for each ϵ and m. In this way we obtain a quite uniform relative precision for all the $C_m(\epsilon)$ values. The $\mathcal{N}_{th}$ value determines the accuracy level.

B. Algorithm description

This procedure can be implemented in a very efficient manner using an adaptive box-assisted algorithm with a number of decreasing cutoff distances, which coincide with the considered ϵ values. In the following we describe the algorithm, whose pseudocode is reported in the Appendix. We choose $N_{\max}$= the maximum number of available points; $m_{\max}$= the maximum embedding dimension; $\mathcal{N}_{th}$= the threshold value for the number of pairs $\mathcal{N}_m(N,\epsilon)$ [see Eq. (11)]; and the sequence of ϵ values as $\epsilon_l = a^{-l}$, $1 < a \leqslant 2$, $l = 0, \ldots, L-1$. $\epsilon_{\min} = a^{-(L-1)}$ is the minimum value of ϵ. L gives the number of length scales considered; a and L must be suitably chosen in order to obtain a good resolution and a reasonable $\epsilon_{\min}$ value. To allow maximum generality, we also accept noninteger values of a, resulting in noninteger $\epsilon/\epsilon_{\min}$ ratios. The number of correlation integral values to compute is $n_{el} = L(m_{\max}-1)$.

Point coordinates are normalized to the unit interval and preprocessed by dividing them by $\epsilon_{\min}$. To save memory we use the time series values directly, with the correspondence

$$x_{iv} = x_{i+(v-1)u}, \quad i = 1, \ldots, N_{\max}, \quad v = 1, \ldots m, \tag{13}$$

where the integer u ($\geqslant 1$) gives the time delay. Distances are computed using the maximum norm.

1. The mesh of boxes

As in [5], we use an m'-dimensional mesh of boxes and linked lists for a fast search of neighboring points. Typically $m' = 2$; in some circumstances, especially if $m_{\max}$ is very high, a greater value (e.g., $m' = 3$) can speed up the computation but causes the loss of C_m for $m < m'$ and an increased demand on memory. For this work we chose $m' = 2$.

The two-dimensional (2D) subspace is divided into nonoverlapped boxes of edge $\epsilon_{\min}$; there are $n_b = 1/\epsilon_{\min}$ boxes along each side, and $M = n_b^2 = a^{2(L-1)}$ is the total number of boxes. The array B, of size M, corresponds to the mesh of boxes, and is used to build in the array LL the linked lists of indices of points falling into the same box. If $\epsilon_{\min}$ is very small, we can save memory by employing the wrapping [5].

At the beginning, all the elements of both B and LL are set to zero. During the procedure, each nonempty location of B points to the head of the corresponding linked list.

2. Computing and histogramming distances at the kth step

The algorithm comprises L steps. At the generic kth step ($k = 0, \ldots, L-1$) the cutoff distance is $\epsilon_k = a^{-k}$ and the locality of the search is expressed by

$$\rho_k = \text{Ceil}\left(\frac{\epsilon_k}{\epsilon_{\min}} \right), \tag{14}$$

where Ceil(x) denotes the smallest integer greater than or equal to x. In the first step ($k = 0$ and $\epsilon_0 = 1$) no distances are discarded.

For every new point $\mathbf{x}_i$, picked out randomly among the set of $N_{\max}$ points, the proper location of B is addressed with the integer parts of the first two coordinates. The linked lists corresponding to the boxes in the square of side $(2\rho_k+1)^2$, centered around the box containing point i, are sequentially scanned; the pairs formed by the current point and previous points in these linked lists are considered as candidate pairs ($\mathcal{N}_c$) for distance computation, and a subset of them has distance $< \epsilon_k$ in the R^2 subspace (see Fig. 1).

For these points the distances $d^{(m)}$ less than ϵ_k (actually less than $\epsilon_k/\epsilon_{\min}$ owing to preprocessing of point coordinates) are recursively computed for the various embeddings and binned in the usual way, simultaneously updating the histograms of distances $[H(m,l), l = k, \ldots, L-1]$. Precisely, $H(m,l)$ contains the number of pairs with

$$\frac{a^{-l-1}}{\epsilon_{\min}} < d^{(m)} \leqslant \frac{a^{-l}}{\epsilon_{\min}}. \tag{15}$$

Then the current point is inserted at the beginning of its list and the location of B is updated.

3. Testing phase

Every N_{test} points we compute the cumulative histograms $[H_c(m,l), l = k, \ldots, L-1]$ and we test the various elements corresponding to not-yet-computed correlation integral val-

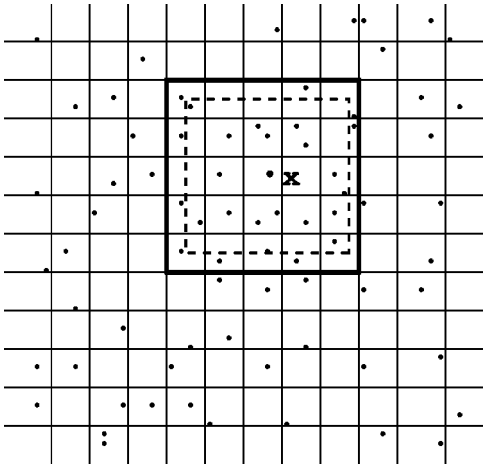

FIG. 1. Example with $\epsilon = 2\epsilon_{\min}$, i.e., $\rho = 2$ and $2\rho + 1 = 5$. The small boxes have side $\epsilon_{\min}$; points in the dashed square of side 4ϵ are the close points (maximum norm) to point **x** in the 2D subspace; points in the square of 5×5 boxes form with point **x** the candidate pairs.

ues to find the ones that have reached the $\mathcal{N}_{th}$ threshold. For such values the correlation integral is computed as

$$C(m,l) = \frac{H_c(m,l)}{\mathcal{N}_t(N_{m,l})}, \tag{16}$$

$N_{m,l}$ being the number of points processed so far; this value is saved in $n_p(m,l)$. We note that $H_c(m,l)$ contains the quantity $\mathcal{N}_m(N_{m,l}, \epsilon_l)$ and $C(m,l)$ corresponds to $C_m(N_{m,l}, \epsilon_l)$.

The $H_c(m,l)$ values grow more slowly as m and l increase, according to the correlation integral values. Hence, the $C(m,l)$ values are computed in decreasing order.

In order to limit the testing overhead, various strategies can be adopted to vary the test step during computation. In accordance with Eq. (12), we find it useful to increase N_{test} as $1/(C_{\min})^s$, $C_{\min}$ being the smallest $C(m,l)$ computed so far, and $s \leqslant \frac{1}{2}$.

4. Cutoff distance reduction

When all the $C(m,k)$ for the current k are computed, we begin the next step ($k \leftarrow k+1$), reducing the current cutoff value by dividing it by a. Correspondingly, the square of boxes (Fig. 1) for searching for the neighboring points is reduced. The columns of H from 0 to $k-1$, which correspond to distances greater than the current ϵ_k, are updated no further.

Let N_k be the number of points processed when all the values in the kth column of C have been computed. For all the $C(m,l)$ values computed during step k, we have $N_{k-1} < N_{m,l} \leqslant N_k$.

As the algorithm proceeds, the number of points increases, but the search becomes more local and therefore faster. The box-assisted approach ensures an efficient search of neighboring points at each step.

5. Termination phase

The algorithm proceeds toward the smallest ϵ value ($\epsilon_{\min}$) and terminates when all C values have been computed with the required statistics $\mathcal{N}_{th}$, or when all points have been processed. In this latter case the remaining C values are in any case computed with the available statistics, and the corresponding elements in n_p are set to 1.

The D_2 can then be estimated using the same techniques that apply with the standard procedure (manual or automatic determination of the scaling region, best fitting of slope in the scaling region, test of reliability, etc.). We can choose to use all the available C values, or only those computed that fulfill the threshold criterion.

C. ABA vs BA and BA-2 algorithms

We point out that the ABA algorithm is similar in some aspects to the box-assisted (BA) algorithm presented in [5], but there are important differences, as shown in Table I.

For comparison we also implemented, using the BA algorithm, a two-computation scheme (BA-2 algorithm) derived from the idea proposed in [11]. In the first phase we process N_0 ($\ll N$) points with $\epsilon_0 = 1$ (no distances are discarded) and we find the minimum ϵ_c for the second phase as the minimum ϵ value for which all $C_m(\epsilon)$ have been evaluated. In the second phase we process the whole point set with a cutoff distance $\epsilon_1 \geqslant \epsilon_c$ (of course, a larger ϵ_1 improves accuracy but increases computation time). This procedure assures that the whole correlation integral is obtained.

TABLE I. Comparison of the proposed algorithm (ABA) with the algorithm presented in [5] (BA); $m' = 2$ and the maximum norm is used. Corr. Int. is the correlation integral.

ABA algorithm	BA algorithm
L cutoff distances ($\epsilon_k = a^{-k}, k = 0, \ldots, L-1$) decreasing from 1 up to $\epsilon_{\min}$	one cutoff distance (ϵ_c)
coordinates of points are divided by $\epsilon_{\min}$	coordinates of points are divided by ϵ_c
all Corr. Int. values are computed	only Corr. Int. values for $\epsilon \leqslant \epsilon_c$ are computed
at the kth step, candidate pairs are searched in the $[2\ \text{Ceil}(\epsilon_k/\epsilon_{\min})+1]^2$ neighboring boxes; the search becomes more local as k increases	candidate pairs are searched in the $(2+1)^2$ neighboring boxes
N is variable	N is fixed
$\mathcal{N}_m(N,\epsilon)$ is approximately constant	$\mathcal{N}_m(N,\epsilon)$ is variable

D. ABA algorithm and the fixed-mass approach

In the ABA algorithm we require the number of neighboring pairs $\mathcal{N}_m(N,\epsilon)$ to remain approximately constant at the various scales and at the various embedding dimensions. Therefore the ABA algorithm presents some analogies with the fixed-mass approach [21], proposed for the computation of generalized fractal dimensions, and based on the scaling of the size ϵ (averaged over a set of reference points) of a neighborhood containing K points, as a function of the total number of points. Indeed, both methods rely on the relationship between the number of points N and the neighborhood size ϵ, when the number of neighboring points (or pairs in the case of D_2) is kept constant. However, in the ABA algorithm the independent variable is ϵ, and for each ϵ we find the minimum number of points N needed to have $\mathcal{N}_{th}$ close pairs. In the fixed-mass method the independent variable is N, and we find the average distance ϵ to the K nearest neighbor as N varies. Also, the implementation is different, since the ABA algorithm is a correlation integral algorithm, although box assisted and modified to manage a variable number of points, whereas the fixed-mass algorithm is a K nearest neighbor algorithm.

E. Using the algorithm

The proposed algorithm requires two input parameters, namely, $N_{\max}$ and $\mathcal{N}_{th}$ (the minimum number of pairs with distance less than ϵ), whereas the standard methods require N only. The ABA algorithm must be used with $\mathcal{N}_{th}<\mathcal{N}_t(N_{\max})$. It yields results that are essentially identical to those of the standard method with $N=N_{\max}$, but it saves computing time since for each ϵ value it uses only the minimum number of points needed to achieve the required statistics. The computation time and the precision of the results obviously increase with $\mathcal{N}_{th}$ (typical values are $1\times10^3-1\times10^7$).

The algorithm makes it possible to run trials with a very high number of points (i.e., $N_{\max}\geqslant1\times10^6$), which cannot be carried out with the standard method since computation time would be too high. In this way we can reach lower scales, resulting in wider and flatter scaling regions.

If $\mathcal{N}_{m_{\max}}(N_{\max},\epsilon_{\min})>\mathcal{N}_{th}$ all C elements fulfill the threshold criterion, not all the available points are used, and results depend only on $\mathcal{N}_{th}$ and not on $N_{\max}$. Otherwise, at the smallest scales the whole set of points is used, possibly without reaching the threshold $\mathcal{N}_{th}$ for some $H_c(m,l)$ values.

If $\mathcal{N}_{th}\geqslant\mathcal{N}_t(N_{\max})$ the algorithm behaves like the standard one and all C elements are computed using the whole set of points. It is obviously useless to employ the algorithm in this way since it performs worse than the standard method owing to the overhead arising from managing linked lists and testing the cumulative histogram.

F. Memory demand and computational cost

The proposed algorithm requires, like the BA algorithm [5], an additional storage of $N_{\max}$ words for the linked lists and of n_b^2 words for the array B. Furthermore, $(m_{\max}-1)L$ words are needed for the array n_p. The total memory demand (in words) is

$$W_{\text{tot}}\simeq 2N_{\max}+n_b^2+4(m_{\max}-1)L, \tag{17}$$

which is roughly twice that of the standard algorithm and about the same as the BA algorithm.

A precise computational analysis of the algorithm is quite involved, since the computational complexity depends on the kind of time series, on the D_2 value itself, and on a number of other parameters $(N_{\max},m_{\max},\mathcal{N}_{th},L,a)$. A fairly rough evaluation of the computational cost can be done in terms of the ratio between the number of pairs examined (candidate pairs) $\mathcal{N}_c$ and the total number of pairs $\mathcal{N}_t$. At the kth step this ratio is the ratio between the neighboring boxes of each point and the total number of boxes

$$R_k=\min\left(\frac{(2\rho_k+1)^2}{M},1\right). \tag{18}$$

The incremental number of pairs resulting in the kth step, when the number of points increases from N_{k-1} to N_k, is $(N_{-1}\equiv0)$

$$\Delta\mathcal{N}_k=\frac{(N_k-N_{k-1})(N_k-N_{k-1}-1)}{2}+(N_k-N_{k-1})N_{k-1},\quad k=0,\ldots,L-1. \tag{19}$$

We can therefore express the total number of candidate pairs as

$$\mathcal{N}_c=\sum_{k=0}^{L-1}R_k\Delta\mathcal{N}_k, \tag{20}$$

and the ratio between the number of candidate pairs and the total number of pairs is

$$\frac{\mathcal{N}_c}{\mathcal{N}_t(N_{L-1})}=\frac{\sum_{k=0}^{L-1}R_k\Delta\mathcal{N}_k}{N_{L-1}(N_{L-1}-1)/2}. \tag{21}$$

This ratio, unity in the standard algorithm, gives us an indication of the reduction in the computational cost. The N_k values can be approximately obtained from Eq. (12),

$$N_k=\left(\frac{2\mathcal{N}_{th}}{C_{m_{\max}}(\epsilon_k)}\right)^{1/2}. \tag{22}$$

So the $\Delta\mathcal{N}_k$ quantities (and therefore $\mathcal{N}_c$) are proportional to $\mathcal{N}_{th}$ and increase if the $C_{m_{\max}}(\epsilon_k)$ values decrease (i.e., if the D_2 is higher). If the total number of points is insufficient to reach $\mathcal{N}_m(N_{\max},\epsilon_k)=\mathcal{N}_{th}$ for $k>k'$, the summation in Eq. (20) reduces to the first k' terms.

It is interesting to consider also the BA-2 algorithm, for which the ratio between the number of candidate pairs and the total number of pairs can be expressed as a particular case of Eq. (21) with $L=2$,

$$\frac{\mathcal{N}_c}{\mathcal{N}_t(N)}=\frac{R_1\Delta\mathcal{N}_1+\Delta\mathcal{N}_0}{N(N-1)/2}. \tag{23}$$

TABLE II. Comparison of results obtained with the proposed algorithm (ABA) and with the standard one. In all trials $a=1.055$, $L=128$, and $m_{\max}=14$. #el [$\leqslant L(m_{\max}-1)=1664$] denotes the number of evaluated values of the correlation integral. Numbers in square brackets denote powers of 10.

System	$N_{\max}$	Algorithm	$\mathcal{N}_{th}$	T_{CPU} (s)	#el	$\mathrm{err}_{\mathrm{av}}$	$\mathrm{err}_{\max}$
Lorenz	200 000	std		34 081	1664	Ref.	Ref.
	10 000	std		89	1664	3.1[−2]	0.16
	50 000	std		2283	1664	1.2[−2]	8.2[−2]
	200 000	ABA	1[2]	6	1664	4.7[−2]	0.18
	200 000	ABA	1[3]	15	1664	2.2[−2]	9.3[−2]
	200 000	ABA	1[4]	43	1664	6.7[−3]	3.4[−2]
	200 000	ABA	1[5]	116	1664	3.2[−3]	1.9[−2]
	200 000	ABA	1[6]	403	1664	1.7[−3]	9.9[−3]
	200 000	ABA	1[7]	1752	1664	7.8[−4]	7.3[−3]
2-Lorenz	200 000	std		33 443	1589	Ref.	Ref.
	10 000	std		84	1322	0.12	1.14
	50 000	std		2107	1527	4.2[−2]	1.2
	200 000	ABA	1[2]	63	1589	2.8[−2]	0.15
	200 000	ABA	1[3]	124	1589	1.1[−2]	8.1[−2]
	200 000	ABA	1[4]	309	1589	4.9[−3]	2.6[−2]
	200 000	ABA	1[5]	839	1589	2.0[−3]	1.5[−2]
	200 000	ABA	1[6]	2332	1589	1.1[−3]	1.1[−2]
	200 000	ABA	1[7]	6408	1589	5.6[−4]	7.1[−3]
	200 000	ABA	1[8]	14 584	1589	2.4[−4]	3.0[−3]
	2 000 000	ABA	1[7]	86 966	1664		
3-torus	200 000	std		32 487	1603	Ref.	Ref.
	10 000	std		83	1456	0.10	0.93
	50 000	std		2097	1575	6.4[−2]	0.9
	200 000	ABA	1[2]	38	1603	2.4[−2]	0.12
	200 000	ABA	1[3]	46	1603	1.0[−2]	5.7[−2]
	200 000	ABA	1[4]	79	1603	3.7[−3]	2.1[−2]
	200 000	ABA	1[5]	186	1603	1.5[−3]	1.1[−2]
	200 000	ABA	1[6]	588	1603	8.7[−4]	6.0[−3]
	200 000	ABA	1[7]	2200	1603	2.2[−4]	1.9[−3]

Since the ABA algorithm uses the minimum number of points needed to obtain the required accuracy, it is expected to outperform the BA-2 algorithm, which can be viewed as an approximation of the ABA algorithm with only one cutoff distance reduction.

IV. EXPERIMENT RESULTS

To test the algorithm, we performed some trials using time series from known attractors. We chose $a=1.055$ and $L=128$, i.e., $\epsilon_{\min}=0.001$, which give 128 points uniformly distributed over three decades. We fixed $m_{\max}=14$. All trials were carried out on a DIGITAL personal workstation Alpha AU 433 (clock speed 433 MHz).

We considered the following time series: (i) The x variable from the Lorenz system, with parameters $\sigma=10$, $R=28$, $b=8/3$ ($D_2=2.06$). (ii) A time series obtained by adding two time series (x variable) from two different Lorenz systems (2-Lorenz in Table II): the first with $\sigma=10$, $R=28$, $b=8/3$ and the second with $\sigma=40$, $R=16$, $b=4$. The resulting time series is expected to give $D_2\simeq 4.12$. (iii) A three-torus obtained by summing up three sinusoids with mutually incommensurable frequencies. In all cases we performed a reference trial with the standard procedure and $N=200\,000$ points, whose results are assumed as reference values.

The results are presented in Table II. #el is the number ($\leqslant n_{el}$) of correlation integral values evaluated in each trial. For each trial we consider the CPU time (in seconds) needed for the computation, the average absolute error $\mathrm{err}_{\mathrm{av}}$, and the maximum absolute error $\mathrm{err}_{\max}$ between the #el values of $\log C(m,\epsilon)$ and the reference values. We see that the ABA algorithm is considerably faster than the standard one. The error reduces as $\mathcal{N}_{th}$ increases, eventually becoming negligible. #el increases, as expected, with the number of points, and, for a given $N_{\max}$, the ABA algorithm allows the computation of the same number of C values as the standard algorithm with an accuracy that depends on $\mathcal{N}_{th}$.

The 2-Lorenz time series is analyzed in more detail in Fig. 2, where some plots of the derivative of $\log_{10}C_m(\epsilon)$ vs $\log_{10}\epsilon$ are compared. In this case $N=50\,000$ is the minimum number of points needed to obtain a quite well-defined scaling region. However, the scaling range is short and there are high statistical errors at the smallest scales. The situation is much better with $N=200\,000$, but the standard algorithm requires about 9 h of CPU time. The ABA algorithm gives us the same results within a negligible error in a fraction of the

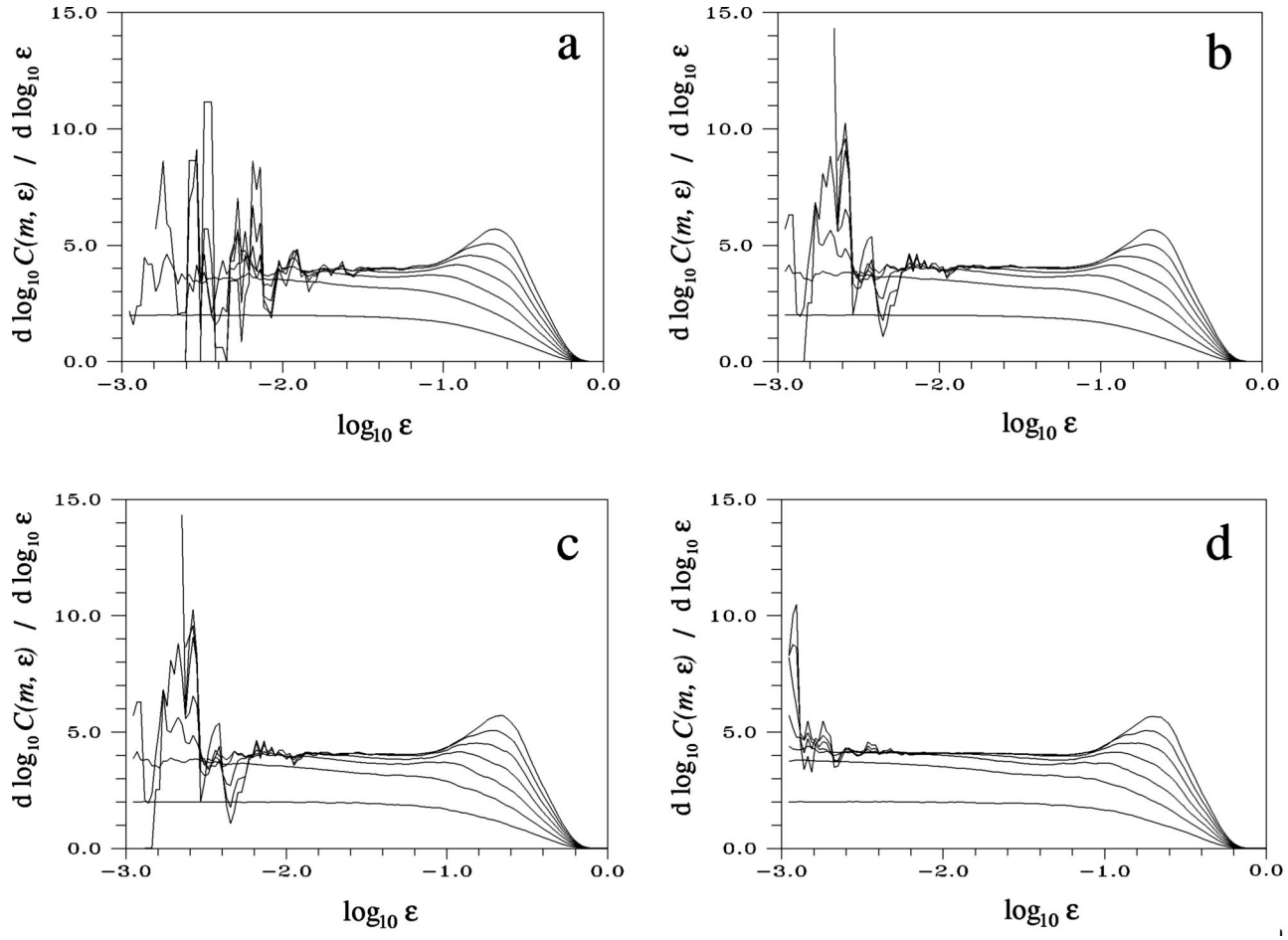

FIG. 2. Examples of plots of the derivative of $\log_{10}C(m,\epsilon)$ vs $\log_{10}\epsilon$ for the 2-Lorenz time series ($m=2,14$ step 2): (a) standard algorithm, $N=50\,000$, $T_{\mathrm{CPU}}=2107$ s; (b) standard algorithm, $N=200\,000$, $T_{\mathrm{CPU}}=33\,443$ s; (c) ABA algorithm, $N=200\,000$, $\mathcal{N}_{th}=1\times10^7$, $T_{\mathrm{CPU}}=6408$ s; (d) ABA algorithm, $N=2\,000\,000$, $\mathcal{N}_{th}=1\times10^7$, $T_{\mathrm{CPU}}=86\,966$ s.

CPU time. Figure 2(d) shows an example of use of the algorithm with a very high number of points: the scaling region remains flat up to $\log_{10}\epsilon\simeq-2.6$. The CPU time is about 25 h, but in this case the standard algorithm would require 38 days.

Table III shows some results obtained with the BA-2 algorithm. N_0 is the subset of points used in phase 1; as expected the number of C elements that we can compute with a fixed N_0 decreases if the system complexity increases and, as a consequence, we must increase the cutoff distance ϵ_1 in phase 2.

Although the BA-2 algorithm greatly reduces computing time, giving quite good results, we see that the ABA algorithm in general performs better. In fact, comparing trials with similar CPU time (see Tables II and III and Fig. 3, which refers to the 2-Lorenz system), we see that the errors obtained with the ABA algorithm are lower. This occurs both for average errors and, to a larger extent, for maximum

TABLE III. Results obtained with the BA-2 algorithm. In all trials $a=1.055$, $L=128$, and $m_{\max}=14$. N_0 is the number of points used in phase 1; ϵ_1 is the cutoff distance in phase 2. Numbers in square brackets denote powers of 10.

System	N	N_0	ϵ_1	T_{CPU} (s)	#el	$\mathrm{err}_{\mathrm{av}}$	$\mathrm{err}_{\max}$
Lorenz	200 000	1000	0.014	136	1664	7.0[−3]	0.11
	200 000	5000	0.003	37	1664	1.2[−2]	0.13
	200 000	20 000	0.002	467	1664	3.2[−3]	1.5[−2]
2-Lorenz	200 000	1000	0.070	1698	1589	5.8[−3]	0.17
	200 000	5000	0.040	626	1589	4.4[−3]	0.10
	200 000	20 000	0.014	502	1589	1.4[−2]	0.41
3-torus	200 000	1000	0.040	370	1603	1.8[−2]	0.28
	200 000	5000	0.018	102	1603	8.2[−3]	0.10
	200 000	20 000	0.006	433	1603	4.0[−2]	0.42

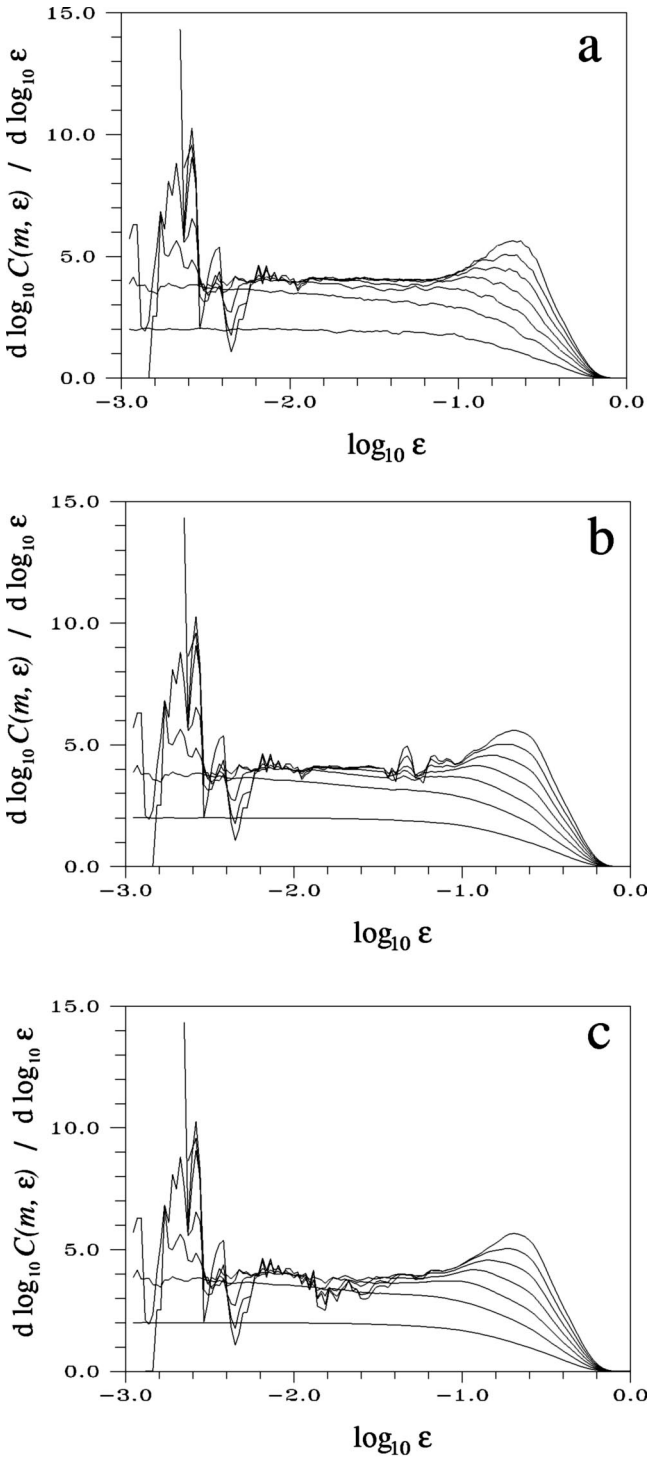

FIG. 3. Plots of the derivative of $\log_{10}C(m,\epsilon)$ vs $\log_{10}\epsilon$ for the 2-Lorenz time series ($N=200\,000$, $m=2{,}14$ step 2): (a) ABA algorithm, $\mathcal{N}_{th}=1\times10^5$, $T_{\mathrm{CPU}}=839$ s; (b) BA-2 algorithm, $N_0=5000$, $\epsilon_1=0.040$, $T_{\mathrm{CPU}}=626$ s; (c) BA-2 algorithm, $N_0=20\,000$, $\epsilon_1=0.014$, $T_{\mathrm{CPU}}=502$ s.

errors. In particular, the ABA algorithm gives a more uniform accuracy and wider and flatter scaling regions (see Fig. 3).

Furthermore, the ABA algorithm is fully adaptive, whereas with the BA-2 algorithm the proper choice of N_0 requires some *a priori* information about the system under investigation.

```
begin procedure corr_int(X, N_max, C)
    input time series X
    ε_min = a^{-(L-1)}
    preprocessing of X (x_i ← (x_i − x_min)/(x_max − x_min))/ε_min)
    set elements of B, LL, n_p to zero
    n_el = L(m_max − 1)
    k = 0
    ε = 1
    ρ = Ceil(ε/ε_min)
    cnt = 0
    do while (n_el > 0 & cnt < N_max)
        i = rand(1, N_max)
        cnt = cnt + 1
        i1 = int(x_i1)
        i2 = int(x_i2)
        j1l = max(i1 − ρ, 0)
        j1h = min(i1 + ρ, n_b)
        j2l = max(i2 − ρ, 0)
        j2h = min(i2 + ρ, n_b)
        for j1 = i1l, i1h
            for j2 = i2l, i2h
                j = B(j1, j2)
                do while(j ≠ 0)
                    if (j < i − h) then
                        dist(x_i, x_j, ε, H)
                    endif
                    j = LL(j)
                enddo
            endfor
        endfor
        LL(i) = B(i1, i2)
        B(i1, i2) = i
        if (cnt mod N_test = 0) check(cnt, H, C, n_el, n_p, k, ε)
    enddo
    if (n_el > 0) check(cnt, H, C, n_el, n_p, k, ε)
    if (n_el > 0) compute the remaining elements of C
end
```

FIG. 4. The main procedure CORR_INT.

V. CONCLUSIONS

The proposed algorithm compares favorably with the standard procedure. With a suitable choice of the parameter $\mathcal{N}_{th}$, we achieve a substantial reduction in computation time, obtaining essentially the same correlation integral values as in the standard procedure. The algorithm also outperforms the two-computation scheme outlined in [11].

A major feature of the algorithm is that it is fully adaptive to the system under consideration, since it does not require any *a priori* knowledge. Moreover, for each ϵ value it uses the minimum number of points necessary to obtain the required statistics. So, for a given accuracy, the computation time increases with the complexity of the system (see, for example, the Lorenz and 2-Lorenz time series in Table II). Varying $\mathcal{N}_{th}$, we can choose the degree of accuracy of results and the related computation time. For intermediate values of N and using quite a low threshold $\mathcal{N}_{th}$, we can reach very fast processing.

Obviously, if we lower $\mathcal{N}_{th}$ the execution becomes faster, but the accuracy of results is reduced. In some circumstances a trial with a very low value of $\mathcal{N}_{th}$ may be useful for obtaining a fairly rough but very fast estimation of the correla-

```
begin procedure dist(x, y, ε, H)
  for m = 2, m_max
    computes d^(m)
    if (d^(m) ≥ ε/ε_min) exit
    if (d^(m) < 1) then
      l = L − 1
    else
      l = int(−log_a d^(m)) + L − 1
    endif
    H(m,l) = H(m,l) + 1
  endfor
end
```

FIG. 5. Distance computation and histogram updating.

```
begin procedure check(cnt, H, C, n_el, n_p, k, ε)
  N_t = (cnt−h)(cnt−h−1)/2
  /* compute the cumulative histogram H_c */
  copy H in H_c
  for m = 2, m_max
    for l = L − 2, k, −1
      H_c(m,l) = H_c(m,l) + H_c(m,l+1)
    endfor
  endfor
  /* test for H_c(m,l) > N_th */
  for m = 2, m_max
    for l = L − 1, k, −1
      if (n_p(m,l) = 0 & H_c(m,l) > N_th) then
        C(m,l) = H_c(m,l)/N_t
        n_p(m,l) = cnt
        n_el = n_el − 1
      endif
    endfor
  endfor
  /* check if ε can be reduced */
  if (all C(:,k) values have been computed) then
    k = k + 1
    ε = ε/a
    ρ = Ceil(ε/ε_min)
  endif
end
```

FIG. 6. The procedure CHECK computes the correlation integral elements that satisfy the threshold $\mathcal{N}_{th}$.

tion integral (and therefore of the correlation dimension), which can be refined later, if necessary.

The algorithm is suitable for carrying out trials with a very high number of points (in our trials we used up to 15 million points). Of course, even if the proposed method allows the efficient computation of correlation integrals, it does not solve all the problems connected to the estimation of the correlation dimension, especially in the case of medium-high dimensions (e.g., $D_2 \geqslant 5$). These intrinsic difficulties are mainly due to a very high depopulation at small scales and higher embeddings, so that a huge number of points is needed to achieve sufficient statistics. Such a huge number of points is not normally available in experimental situations.

In order to increase the speed of the algorithm, especially when dealing with ''difficult'' time series, we can slightly lower the threshold $\mathcal{N}_{th}$ for the smallest ϵ values.

With a few modifications, i.e., introducing a convergence test for each correlation integral value, the proposed method can be used as a tool to perform experimental investigation of the convergence behavior of the correlation integral for a given system, particularly when a very high number of points is available. The algorithm can easily be adapted for the computation of the individual correlation integral with respect to a given reference point, and therefore it can be used to compute the pointwise dimension and the q-order generalized dimensions.

APPENDIX: PSEUDOCODE OF THE ALGORITHM

The algorithm comprises the main procedure CORR_INT (Fig. 4) and the procedures DIST, which computes distances and updates the histograms (Fig. 5), and CHECK, which computes the correlation integral elements that fulfill the threshold $\mathcal{N}_{th}$ and reduces the cutoff distance if appropriate (Fig. 6). ϵ and ρ denote the current values of ϵ_k and ρ_k.

The arrays used are $X(1{:}N_{\max})$ with the time series, $H(2{:}m_{\max},0{:}L-1)$ for the histogram, $C(2{:}m_{\max},0{:}L-1)$ for the correlation integral, $B(0{:}n_b-1,0{:}n_b-1)$ with the mesh of boxes, $LL(1{:}N_{\max})$ where the linked lists are built, and $n_p(2{:}m_{\max},0{:}L-1)$ to check if the C elements have been computed and to store the number of points at which they are computed. Moreover, the procedure CHECK uses the local array $H_c(2{:}m_{\max},0{:}L-1)$ for the cumulative histogram.

The computer program, in FORTRAN language, is available to those who request it.

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

[2] P. Grassberger and I. Procaccia, Physica D **9**, 189 (1983).

[3] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis* (Cambridge University Press, Cambridge, 1997).

[4] J. Theiler, J. Opt. Soc. Am. A **7**, 1055 (1990).

[5] P. Grassberger, Phys. Lett. A **148**, 63 (1990).

[6] A. Corana, A. Casaleggio, C. Rolando, and S. Ridella, Parallel Comput. **17**, 809 (1991).

[7] E. Toledo, S. Toledo, Y. Almog, and S. Akselrod, Phys. Lett. A **229**, 375 (1997).

[8] S. Chirravuri, S.M. Bhandarkar, and D. Whitmiret, Int. J. Supercomput. Appl. **9**, 296 (1995).

[9] A. Corana, Concurrency: Practice and Experience **10**, 737 (1998).

[10] S. Bingham and M. Kot, Phys. Lett. A **140**, 327 (1989).

[11] J. Theiler, Phys. Rev. A **36**, 4456 (1987).

[12] M. Ding, C. Grebogi, E. Ott, T. Sauer, and J.A. Yorke, Physica D **69**, 404 (1993).

[13] H. Isliker, Phys. Lett. A **169**, 313 (1992).

[14] A. Casaleggio and A. Corana, Chaos Solitons Fractals **11**, 2017 (2000).
[15] H. Heng, M. Bauer, and W. Martienssen, Chaos Solitons Fractals **7**, 197 (1996).
[16] M.A.H. Nerenberg and C. Essex, Phys. Rev. A **42**, 7065 (1990).
[17] N.A. Gershenfeld, Physica D **55**, 135 (1992).
[18] A. Jedynak, M. Bach, and J. Timmer, Phys. Rev. E **50**, 1770 (1994).
[19] J.P. Eckmann and D. Ruelle, Physica D **56**, 185 (1992).
[20] A. Galka, T. Maab, and G. Pfister, Physica D **121**, 237 (1998).
[21] R. Badii and A. Politi, J. Stat. Phys. **40**, 725 (1985).