ELSEVIER

Physica A 271 (1999) 427–447

PHYSICA A

www.elsevier.com/locate/physa

The use of generalized information dimension in measuring fractal dimension of time series

Y. Ashkenazy

Department of Physics, Bar-Ilan University, Ramat-Gan 52900, Israel

Received 5 May 1999

Abstract

An algorithm for calculating generalized fractal dimension of a time series using the general information function is presented. The algorithm is based on a strings sort technique and requires $O(N \log_2 N)$ computations. A rough estimate for the number of points needed for the fractal dimension calculation is given. The algorithm was tested on analytic example as well as well-known examples, such as, the Lorenz attractor, the Rossler attractor, the van der Pol oscillator, and the Mackey–Glass equation, and compared, successfully, with previous results published in the literature. The computation time for the algorithm suggested in this paper is much less than the computation time according to other methods.

PACS: 05.45.+b; 47.52.+j; 47.53.+n

Keywords: Fractal dimension; Time series; Information dimension; Correlation dimension

1. Background

In the recent decades the study of chaos theory has gathered momentum. The complexity that can be found in many physical and biological systems has been analyzed by the tools of chaos theory. Characteristic properties, such as the Lyapunov exponent, Kolmogorov entropy and the fractal dimension (FD), have been measured in experimental systems. It is fairly easy to calculate signs of chaos if the system can be represented by a set of non-linear ordinary differential equations. In many cases it is very difficult to build a mathematical model that can represent sharply the experimental system. It is essential, for this purpose, to reconstruct a new phase space based on the

E-mail address: ashkenaz@alon.cc.biu.ac.il (Y. Ashkenazy)

0378-4371/99/$ - see front matter
PII: S0378-4371(99)00192-2

information that one can produce from the system. A global value that is relatively simple to compute is the FD. The FD can give an indication of the dimensionality and complexity of the system. Since actual living biological systems are not stable and the system complexity varies with time, one can distinguish between different states of the system by the FD. The FD can also determine whether a particular system is more complex than other systems. However, since biological systems are very complex, it is better to use all the information and details the system can provide. In this paper we will present an algorithm for calculating FD based on the geometrical structure of the system. The method can provide important information, in addition, on the geometrical of the system (as reconstructed from a time series).

The most common way of calculating FD is through the correlation function, $C_q(r)$ (Eq. (6)). There is also another method of FD calculation based on Lyapunov exponents and the Kaplan–Yorke conjecture [1] (Eq. (27)). However, the computation of the Lyapunov exponent spectrum from a time series is very difficult and people usually try to avoid this method [1]. The algorithm which is presented in this paper is important, since it gives a comparative method for calculating FD according to the correlation function. The need for an additional method of FD calculation is critical in some types of time series, such as EEG series (which are produced by brain activity), since several different estimations for FD have been published in the literature [2–7]. A comparative algorithm can help to reach final conclusions about the FD estimate for the signal.

A very simple way to reconstruct a phase space from single time series was suggested by Takens [8]. Giving a time series x_i, $i = 1,\ldots,N_p$, we build a new n-dimensional phase-space in the following way:

$$\begin{aligned}
&\vec{y}_0 = \{x(t_0), x(t_0+\tau), x(t_0+2\tau), \ldots, x(t_0+(n-1)\tau)\}\,,\\
&\vec{y}_1 = \{x(t_1), x(t_1+\tau), x(t_1+2\tau), \ldots, x(t_1+(n-1)\tau)\}\,,\\
&\vdots\\
&\vec{y}_N = \{x(t_N), x(t_N+\tau), x(t_N+2\tau), \ldots, x(t_N+(n-1)\tau)\}\,,
\end{aligned}$$

$$t_i = t_0 + \mathrm{i}\Delta t, \quad \tau = m\Delta t, \quad N = N_p - (n-1)m, \quad m = integer\,, \tag{1}$$

where Δt is the sampling rate, τ corresponds to the interval on the time series that creates the reconstructed phase-space (it is usually chosen to be the first zero of the autocorrelation function (discussions about the choose of τ see for example [9–11], or the first minimum of mutual information [12]; in this work we will use m instead of τ as an index), and N is number of reconstructed vectors. For ideal systems (an infinite number of points without external noise) any τ can be chosen. Takens proved that such systems converge to the real dimension if $n \geqslant 2D+1$, where D is the FD of the real system. According to Ding et al. [13] if $n \geqslant D$ then the FD of the reconstructed phase-space is equal to the actual FD.

[1] The basic difficulty is that for high FD there are some exponents which are very close to zero; one can easily add an extra unnecessary exponent that can increase the dimensionality by one; this difficulty is most dominant in Lyapunov exponents which have been calculated from a time series.

In this paper we will first (Section 2) present regular analytic methods for calculating the FD of a system (generalized correlation method and generalized information method). An efficient algorithm for calculating the FD from a time series based on string sorting will be described in the next section (Section 3). The following step is to check and compare the general correlation methods with the general information method (see Eq. (2)) using known examples (Section 4). Finally, we summarize in Section 5.

2. Generalized information dimension and generalized correlation dimension

2.1. Generalized information dimension

The basic way to calculate an FD is with the Shannon entropy, $I_1(\varepsilon)$[2] [14], of the system. The entropy is just a particular case of the general information which is defined in the following way [15]:

$$I_q(\varepsilon) = \frac{1}{1-q} \ln \left(\sum_{i=1}^{M(\varepsilon)} p_i^q \right) , \tag{2}$$

where we divide the phase-space to $M(\varepsilon)$ hypercubes of edge ε. The probability to find a point in the ith hypercube is denoted by p_i. The generalization, q, can be any real number; we usually use just an integer q. When we increase q, we give more weight to more populated boxes, and when we decrease q the less occupied boxes become dominant. For $q = 1$, by applying the l'Hospital rule, Eq. (2) becomes

$$I_1(\varepsilon) = - \sum_{i=1}^{M(\varepsilon)} p_i \ln p_i , \tag{3}$$

where $I_1(\varepsilon)$ is referred to also as the Shannon entropy [11] of the system. The definition of the general information dimension (GID) is

$$D_q = - \lim_{\varepsilon \to 0} \frac{I_q(\varepsilon)}{\ln \varepsilon}, \tag{4}$$

for $N \to \infty$ (N is the number of points). However, in practice, the requirement $\varepsilon \to 0$ is not achieved, and an average over a range of ε is required.

In some cases, this average is not sufficient because several values of $I_q(\varepsilon)$ can be computed for the same ε. To illustrate this we use a $2D$ Cantor set, presented in Fig. 1. We start from a square. From 9 sub-squares we erase the 5 internal squares. We continue with this evolution for each remainder square. This procedure is continued, in principle, to infinity. The FD, D_0, of the $2D$ Cantor set is $\ln 4/\ln 3$ (when $q=0$, I_0 is just the number of nonempty squares and D_0 is the logarithmic ratio between nonempty squares and ε, where the square edge is normalized to one) as shown in Fig. 1.

[2] Generally, one has to add a superscript, n, the embedding dimension in which the general information is calculated. At this stage we assume that the embedding dimension is known.

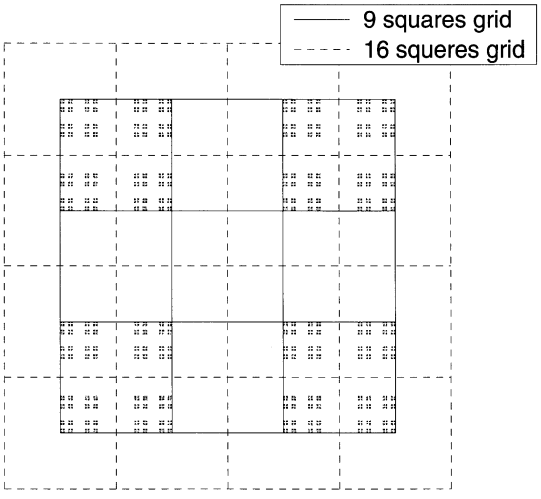

Fig. 1. 2D Cantor set. Two different positions of the two-dimensional grid, give completely different FD.

However, it is possible to locate the 2D grid in such a way that there are 16 nonempty squares, giving rise to $D_0 = 2\ln 4/\ln 3$, twice the time of the real FD of the system. This illustration shows that when ε is not small, different positions of the grid can lead to different FDs. In this case it is clear that we have to locate the grid in such a way that minimum squares will be nonempty (it is easy to show that this claim for the minimum is true for every q). In general, we can say that one must locate the hyper-grid so that the general information is minimum:

$$\tilde{I}_q(\varepsilon) \equiv \min \frac{1}{1-q} \ln \left(\sum_{i=1}^{M(\varepsilon)} p_i^q \right) , \tag{5}$$

This proper location of the hyper-grid reduces the influence of surface boxes that are partly contained in the attractor. The requirement of a minimum gives a good estimate for the GID when ε is not small.

2.2. Generalized correlation dimension

In 1983 Grassberger and Procaccia presented a new method for calculating D_2 [16]. According to this method, it is possible to calculate the dimension just from the correlation between different points, without direct connection to the phase space, and therefore easy to use. Some years later, a more general correlation function was suggested by Pawelzik and Schuster [17]:

$$C_q(r) = \left[\frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{N} \sum_{j=1}^{N} \Theta \left(r - |\vec{x}_i - \vec{x}_j| \right) \right]^{q-1} \right]^{1/(q-1)} , \tag{6}$$

where N is the number of points, $\vec{x}_i$ is a point of the system and q is the generalization parameter. $\Theta(x)$ is the Heaviside step function

$$\Theta(x) = \begin{cases} 0 \text{ when } x \leqslant 0\,, \\ 1 \text{ when } x > 0\,. \end{cases} \tag{7}$$

According to this method we have to calculate the generalized probability to find any two points within a distance r. This method is some kind of integration on Eq. (2). It is not necessary to compute the real distance (e.g. $\|\vec{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}$ where n is the phase-space dimension); it is equivalent to calculating the probability to find any two points in a hyper-box where one of them is located in the middle (e.g. $\|\vec{x}\| = \max_{1 \leqslant i \leqslant n} |x_i|$ [18]). It is easier to compute the last possibility. For the special case of $q = 1$, Eq. (6) can be written (by applying the l'Hospital's rule [12]) as

$$\ln C_1(r) = \frac{1}{N} \sum_{i=1}^{N} \ln \left[\frac{1}{N} \sum_{j=1}^{N} \Theta(r - |\vec{x}_i - \vec{x}_j|) \right] . \tag{8}$$

The generalized correlation dimension (GCD) has a similar definition to the GID (Eq. (4)):

$$D_q = \lim_{N \to \infty; r \to 0} \frac{\ln C_q(r)}{\ln r} . \tag{9}$$

Both GID and GCD methods should give identical results. The GCD method is easy use and gives smooth curves. On the other hand, the method requires $O(N^2)$ computations [3]. Also, the smooth curves due to averaging over all distances are associated with a loss in information based on the attractor structure.

As we pointed out earlier, we usually have a limited number of points, forcing us to calculate dimension at large r. Thus, an error enters the calculation of FD. The minimum number of points needed for the FD calculation has been discussed in several papers and different constraints suggested (for example [20,21]). In this paper we will use the $N_{\min}$ of the Eckmann and Ruelle [21] constraint:

$$D_2 < \frac{2 \log_{10} N}{\log_{10}(\frac{1}{\rho})} , \tag{10}$$

under the following conditions:

$$\rho = \frac{r}{r_0} \ll 1, \quad \tfrac{1}{2} N^2 \rho^D \gg 1 \tag{11}$$

with reference to the Grassberger and Procaccia method. Here, r_0 is the attractor diameter, and r is the maximum distance that gives reliable results in Eq. (6) when $q = 2$. The normalized distance, ρ, must be small because of the misleading influence of the hyper-surface. A ρ too large (close to 1) can cause incorrect results since we take

[3] If one looks just at small r values, the method requires just $O(N)$ computations [18].

into account also hyper-boxes that are not well occupied because the major volume is outside the attractor.

However, if one has a long time series, then it is not necessary to compute all N^2 relations in Eq. (6). One can compute Eq. (6) for certain reference points such that the conditions in Eq. (11) will still hold. Then, Eq. (6) can be written as follows:

$$C_q(r) = \left[\frac{1}{N_{ref}} \sum_{i=1}^{N_{ref}} \left[\frac{1}{N_{data} - 2w - 1} \sum_{j=1}^{N_{data}} \Theta(r - |\vec{x}_{i\cdot s} - \vec{x}_j|) \right]^{q-1} \right]^{1/(q-1)} , \qquad (12)$$

where

$$|i \cdot s - j| > w, \quad s = \left\lfloor \frac{N_{data}}{N_{ref}} \right\rfloor .$$

For each reference point we calculate the correlation function over all data points. The step for the time series is s. To neglect short time correlation one must also introduce a cutoff, w. Usually $w \approx m$ (m corresponds to the τ from (1)) [22]. The number of distances $|\vec{x}_i - \vec{x}_j|$ will be $P = N_{ref}(N_{data} - 2w - 1)$ instead of N^2. The new form of Eq. (10) is

$$D < \frac{\log_{10} P}{\log_{10}(\frac{1}{\rho})} , \qquad (13)$$

and the conditions (11) become

$$\rho = \frac{r}{r_0} \ll 1, \quad \tfrac{1}{2} P \rho^D \gg 1 . \qquad (14)$$

Although we discussed here a minimum number of data points needed for the dimension calculation, one must choose ρ such that the influence from the surface is negligible (the growth of the surface is exponential to the embedding dimension). That gives us an upper limit for ρ. On the other hand, in order to find the FD we must average over a range of r, giving us a lower limit for ρ; therefore $N_{\min}$ is determined according to the lower limit, giving rise to larger amount of points.

3. Algorithm

3.1. Background

In this section we describe an algorithm for GID method[4]. The algorithm is based on a string sort and can be useful both for GID and GCD methods.

In previous works there have been several suggestions to compute DIG [23–28]. The key idea of some of those methods [25–27] is to rescale the coordinates of each point and to express them in binary form. Then, the edge-box size can be initialized by the lowest value possible and then it doubles on each step. Those methods require

[4] Computer programs are available at *http://faculty.biu.ac.il/∼ ashkenaz/FDprog/*.

$O(N \log N)$ operations. Another method [24] uses a recursive algorithm to find the FD. The algorithm starts from a d dimensional box which contains all data points. Then, this box is divided into 2^d sub-boxes, and so on. The empty boxes are not considered, and those which contain only one point are marked. This procedure requires only $O(N)$ operations (for a reasonable series length this fact does not make a significant difference [24]).

As pointed out in Ref. [24], in spite of the efficiency of the above algorithm (speed and resolution), it is quite difficult to converge to the FD of a high dimensional system. I was aware to this difficulty, and suggested a *smoothing* term to solve it (as will be explained in this section). Basically, we allow any choice of edge length (in contrast to power of 2 edge size of the above methods) and optimal location of the grid is searched. This procedure leads to convergence to the FD.

One of the works that calculates GID was done by Caswell and Yorke [23]. They calculate FD of 2D maps. They have proved that it is very efficient to divide the phase space into circles instead of squares in spite of the neglected area between the circles. However, this approach is not suitable for higher phase-space dimensions. The reason might be that the volume of the hyper-balls compared to the entire volume of the attractor decreases according to the attractor dimension. In this way we lose most of the information that is included in the attractor, since the space between the hyper-balls is not taken into account. It is easy to show that the ratio between the volume of the hyper-sphere with radius R, V_S, and the volume of the hyper-box with edge size of $2R$, V_B (the sphere is in the box), is

$$\frac{V_S(2n)}{V_B(2n)} = \frac{1}{2 \cdot 4 \cdot 6 \cdots 2n} \left(\frac{\pi}{2}\right)^n , \tag{15}$$

for even phase space dimension, and,

$$\frac{V_S(2n-1)}{V_B(2n-1)} = \frac{1}{1 \cdot 3 \cdot 5 \cdots (2n-1)} \left(\frac{\pi}{2}\right)^{n-1} , \tag{16}$$

for odd phase space dimension. It is clear that the ratios (15) and (16) tend to zero for large n.

3.2. Algorithm

Let us call the time series $x(i)$. The form of the reconstructed vector, Eq. (1), depends on the jumping, m, that creates the phase space. We order the time series in the following way:

$$\begin{aligned}
&x(0), x(m), x(2m), \ldots, x((l-1)m) , \\
&x(1), x(1+m), x(1+2m), \ldots, x(1+(l-1)m) , \\
&\vdots \\
&x(m-1), x(m-1+m), x(m-1+2m), \ldots, x(m-1+(l-1)m) ,
\end{aligned} \tag{17}$$

where $l = \lfloor N_p/m \rfloor$. Let us denote the new series as $\tilde{x}_i$ ($i = 0, \ldots, (lm - 1)$; we lose $N_p - lm$ data points). If we take n consecutive numbers in each row, we create a reconstructed vector in the n dimensional embedding dimension.

At this stage, we fit a string (for each ε) to the number series Eq. (17). One character is actually a number between $0, \ldots, 255$, and thus, it is possible to divide one edge of the hyper-box to 255 parts (we need one of the 256 characters for sign). The correspondence is applied in the following way. We search for the minimum value in $\tilde{x}_i$ series, and denote it as $\tilde{x}_{\min}$ (similarly, we denote the maximum values as $\tilde{x}_{\max}$). The corresponding character to $\tilde{x}_i$ is

$$y_i = \left\lfloor \frac{\tilde{x}_i - \tilde{x}_{\min}}{\varepsilon} \right\rfloor \quad . \tag{18}$$

The value of ε is in the range

$$\frac{(\tilde{x}_{\max} - \tilde{x}_{\min})}{255} \leqslant \varepsilon \leqslant (\tilde{x}_{\max} - \tilde{x}_{\min}) \, . \tag{19}$$

If we take now n consecutive characters (string), we have the "address" of the hyper-cube in which the corresponding vector is found.

Let us represent the string as follows:

$$\begin{aligned}
&s_0 = y_0 y_1 \ldots y_{n-1}, \quad s_1 = y_1 \ldots y_n, \ldots, s_{l-n} = y_{l-n} \ldots y_{l-1} \, , \\
&s_{l-n+1} = y_l \ldots y_{l+n-1}, \ldots, s_{2(l-n)+1} = y_{2l-n} \ldots y_{2l-1} \, , \\
&\vdots \\
&s_{(m-1)(l-n+1)} = y_{(m-1)l} \ldots y_{(m-1)l+n-1}, \ldots, s_{m(l-n+1)-1} = y_{ml-n} \ldots y_{ml-1} \, .
\end{aligned} \tag{20}$$

Obviously, we do not have to keep each string s_i in the memory; we can keep just the pointers to the beginning of the strings in the character series, y_i. The number of vectors/strings is $m(l - n + 1)$.

As mentioned, each string is actually the address of a hyper-box (for which the vector is contained) in a hyper-grid that covers the attractor. The first character is the position of the first coordinate of the hyper-box on the first edge of the hyper-grid. The second character denotes the location on the second edge, and so on. We actually grid the attractor with a hyper-grid so that any edge in this grid can be divided into 255 parts. Thus the maximal number of boxes is 255^n, where n is the embedding dimension (there is no limit for n). Most of those boxes are empty, and we keep just the occupied boxes in the hyper-grid.

The next step is to check how many vectors fall in the same box, or, in other words, how many identical "addresses" there are. For this we have to sort the vector that points to strings, s_i, in increasing order (one string is less than the other when the first character that is not equal is less than the parallel character; e.g., '*abcce*' < '*abcda*' since the fourth character in the first string, *c*, is less then the fourth character in the second string, *d*). The above process is illustrated in Fig. 2.

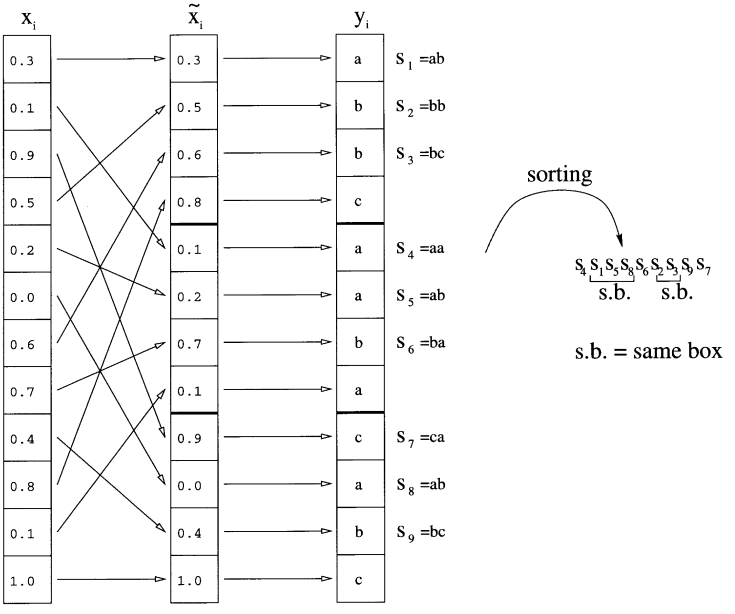

Fig. 2. Illustration of the GID algorithm. We take series containing 12 data points between zero and one. The parameter values are: $m = 3$, $\varepsilon = 0.4$ and $n = 2$. For simplicity, we assume that the lowest character is 'a'.

The most efficient way to sort N elements is the "quick sort" [29] (There is other sort algorithms that require even less computation (O(N)) such as radix sorting [30,31]). It requires just O($N \log_2 N$) computations. However, this sort algorithm is not suitable for our propose, since after the vector is sorted, and we slightly increase the size of the edge of the hyper-box, ε, there are just a few changes in the vector, and most of it remains sorted. Thus, one has to use another method which requires less computations for this kind of situation, because the quick sort requires O($N \log_2 N$) computations independently of the initial state. The sort that we used was a "shell sort" [29], which requires O($N^{3/2}$) computations in worst case, around O($N^{5/4}$) computations for random series, and O(N) operations for an almost sorted vector. Thus, if we compute the information function for m different ε values O($N \log_2 N + mN$) operations will be needed.

After sorting we count how many identical vectors there are in each box. Suppose that we want to calculate GID for embedding dimensions $n_{\min} \dots n_{\max}$, and generalizations $q_{\min} \dots q_{\max}$. We count in the sorted pointers vector identical strings containing $n_{\max}$ characters. Once we detect a difference we know that the coordinate of the box has changed. We detect the first different location. A first difference in the last character of the string means that it is possible to observe the difference just in the $n_{\max}^{\text{th}}$ embedding dimension, and in lower embedding dimension it is impossible to observe the difference. If the first mismatch is in the nth character in the string, then the box changes for embedding dimensions higher than or equal to n. We hold a counter vector

for the different embedding dimensions, and we will assign zero for those embedding dimensions in which we observed a change. At this stage, we have to add to the results matrix the probability to fall in a box according to Eqs. (2), (3). We continue with this method to the end of the pointer vector and then print the results matrix for the current ε and continue to the next ε.

It is easy to show that one does not have to worry about the range of the generalization, q, because for large time series with 10^6 data points the range of computational q's is from -100 to 100.

The method of string sorting can be used also for calculating GCD. Our algorithm is actually a generalization of the method that was suggested by Grassberger [18]. The algorithm is specially efficient for small r ($r < (1/10)D$ where D is the attractor diameter) and it requires $O(N \log_2 N)$ computations instead of $O(N^2)$ computations (according to the Grassberger algorithm it requires $O(N)$ computations instead $O(N^2)$ computations).

The basic idea of Grassberger was that for small $r_{\max}$ values, where $r_{\max}$ is the maximum distance for which the correlation function is computed, one does not have to consider distances larger than $r_{\max}$, distances which require most of the computation time. Thus, it is enough to consider just the neighboring boxes (we use the definition distance $\|\vec{x}\| = \max_{1 \leqslant i \leqslant n} |x_i|$ in Eq. (6)), for which their edge is equal to $r_{\max}$, since distances greater than $r_{\max}$ are not considered in Eq. (6). If, for example, one wants to calculate the correlation of $r \leqslant r_{\max} = (1/10)D$ (in accordance with conditions (10) (11) (13) (14)), then (in $2D$ projection) it is enough to calculate distances in 9 squares instead 100 squares (actually, it is enough to consider just 5 squares since Eq. (6) is symmetric). According to Grassberger, one has to build a matrix in which any cell points to the list of data points located in it. It is possible to generalize this method to a $3D$ projection matrix.

The generalization to higher dimensions can be done very easily according to the string sort method. As a first step one has to prepare a string y_i (18), which is the same operation as gridding the attractor by a hyper-grid of size r. The next step is to sort strings s_i (20). For each reference point in Eq. (12) the boxes adjoining the box with the reference point in it should be found. Now, it is left to find the distances between the reference point to other points in neighboring boxes (we keep the pointer vector from string y_i to the initial series and vice versa), and calculate the correlation according to Eq. (12).

The algorithm described above is especially efficient for very complex systems with high FD. For example, for EEG series produced by brain activity, it is well known (except for very special cases such as epilepsy [19]) that the FD is, at least, four. In this case $n_{\min} = 4$, and instead of calculating distances in l^4 boxes (l is the ratio between the attractor diameter and r) it is sufficient to calculate distances in $3^4 = 81$ boxes. If, for example, $l = 9$, just $3^4/9^4 = 1/81$ distances must be computed (or even less if one takes into account also the symmetry of Eq. (12)). In this way, despite the $O(N \log_2 N)$ operations needed for the initial sort, one can reduce significantly the computation time.

3.3. Testing of GID algorithm and number of data points needed for calculating GID

Let us test the algorithm of GID on random numbers. We create a random series between 0 to 1 (uniform distribution). We then reconstruct the phase-space according to Takens theory (1). For any embedding dimension that we would like to reconstruct, the phase space we will get a dimension that is the same as the embedding dimension, since the new reconstructed vectors are random in the new phase space, and hence fill the space (in our case, a hypercube of edge 1).

For embedding dimension 1 (a simple line), if the edge length is ε, then the probability to fall on any edge is also ε. The number of edges ε that covers the series range $[0,1)$ is $\lfloor 1/\varepsilon \rfloor$. The probability to fall on the last edge is $(1 \bmod \varepsilon)$ (the residue of the division). For generalization q one gets

$$\sum_{i=1}^{M(\varepsilon)} p_i^q = \lfloor 1/\varepsilon \rfloor \varepsilon^q + (1 \bmod \varepsilon)^q \,. \tag{21}$$

For embedding dimension 2, one has to square (21) since the probability distribution on different sides is equal. For embedding dimension n Eq. (2) becomes

$$I_q(\varepsilon) = \frac{1}{1-q} \ln[\lfloor 1/\varepsilon \rfloor \varepsilon^q + (1 \bmod \varepsilon)^q]^n \,. \tag{22}$$

Opening Eq. (22) according to the binomial formula will give all different combinations for the partly contained boxes. Notice that this grid location fulfills requirement (5) for the minimum information function.

In Fig. 3 we present $-I_2(\varepsilon)$, both, according to the GID algorithm, and according to Eq. (22). There is full correspondence between the curves, and, as we find, it is possible to see that the slope (dimension) in the embedding dimension n is equal to the dimension itself. When $-I_2(\varepsilon) \approx -10$, the curves separate. There is lower convergence (when $-I_2(\varepsilon) \approx -12.5$) since the edge is too small, and the number of reconstructed vectors is equal to the number of nonempty boxes.

The "saw tooth" that is seen in Fig. 3 is caused by partial boxes that fall on the edge of the hyper-box. The "saw tooth" appears when there is an integer number of edges that covers the big edge. The slope in the beginning of every saw tooth is equal to zero (that comes from the extremum condition of (22)). The "saw teeth" become smaller when ε decrease, since the relative number of boxes that fall on the surface in small compared to the entire number of boxes.

It is possible to calculate roughly the number of points needed for the dimension calculation in a homogeneous system. The separation point of Fig. 3 is approximately around $-I_2(\varepsilon) \approx -10$, for every embedding dimension. Thus, the number of hyper-boxes at this point is e^{10} (since $-10 = \ln(M(\varepsilon)p^2)$ where $p \sim 1/M(\varepsilon)$ in our example). The graph comes to a saturation around $-I_2(\varepsilon) \approx -12.5$, giving rise to $e^{12.5}$ data points. The number of points per box is just, $e^{2.5} \approx 12$. Thus, generally, when there are less then 12 points per box, the value of $I_q(\varepsilon)$ is unreliable. Let us denote the

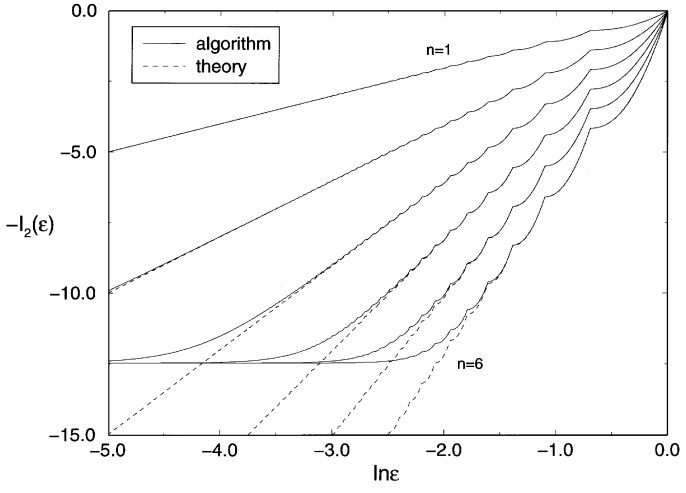

Fig. 3. Dimension calculation for random series by GID algorithm and by the theory.

minimum edges needed for the calculation as $m_{\min}$ ($m_{\min}$ is just $\lfloor 1/\varepsilon \rfloor$, and thus one can define a lower value for ε). The number of hyper-boxes in embedding dimension n is $m_{\min}^n$. Thus, the minimum number of points needed for computing the FD of an attractor with an FD D, is

$$N = 12 m_{\min}^D \,. \tag{23}$$

This estimate is less than what was required by Smith [20] (42^D) and larger then the requirement of Eckmann and Ruelle [21] (if we take, for example, $m_{\min} = 1/\rho = 10$ then according to Eq. (10), $10^{D/2}$ points is needed and according to Eq. (23), 12×10^D points is needed). However, as we pointed out earlier, Eq. (23) is a rough estimation, and one can converge to a desired FD even with less points.

4. Examples

In this section we will present the results of both GID and GCD on well-known systems, such as the Lorenz attractor and the Rossler attractor. The FD of those results are well known by various methods, and we will present almost identical results achieved by our methods. In the following examples, we choose one of the coordinates to be the time series; we normalize the time series to be in the range of $0, \ldots, 1$.

4.1. Lorenz attractor

The Lorenz attractor [32] is represented by a set of three first-order non-linear differential equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x) \,,$$

$$\frac{dy}{dt} = -xz + rx - y\,,$$

$$\frac{dz}{dt} = xy - bz\,. \tag{24}$$

Although it is much simpler to find the FD from the real phase-space (x, y, z), we prefer the difficult case of finding the FD just from one coordinate to show the efficiency of Takens theorem. We choose the zth component to be the time series from which we reconstruct the phase space (the parameter values are: $\sigma = 16$, $r = 45.92$, $b = 4$). We took 32768 data points and $m = 6$ (the jumping on the time series which creates the reconstructed phase space; the time step is $\Delta t = 0.02$). The generalization that we used was $q = 2$. The embedding dimensions were $n = 1 \ldots 7$. The FD of the Lorenz attractor is 2.07 ([33] and others).

In Fig. 4 we show the results of GCD and GID methods. As expected from the GCD curves, in Fig. 4a we see very smooth curves, for which the slopes (in the center parts of the curves) converge approximately to dimension 2.08. In Fig. 4b we see some jumping in the GID curves. It can be clearly seen that for small ε, one cannot see the jumping while for large ε the non-monotonicity is much stronger. This is typical behavior for GID curves; large ε (or larger box edge) reflects more variability of $-I_2(\varepsilon)$ to the hyper-grid location. To smooth the curves in order to get reliable results, it is necessary to find a proper location of the hyper-grid that gives minimum general information $\tilde{I}_2(\varepsilon)$ (or maximum $-\tilde{I}_2(\varepsilon)$). We have performed seven comparisons to find that proper location. As for GCD, we also fitted approximate linear curves (dashed lines) which lead to dimension 2.09 (Fig. 4c), and thus agree well with previous results. To make sure of the validity of the dimension results we add here (Fig. 4d) graphs of successive slopes of curves in Fig. 4c, and conclude with dimension results versus embedding dimension according to several generalizations ($q = 2, \ldots, 5$, Fig. 4e). In both graphs, the approximate dimension is 2.03.

4.2. Rossler attractor

Let us examine briefly our second example, the Rossler attractor [34]. The differential equations which generate the trajectory are

$$\frac{dx}{dt} = -z - y\,,$$

$$\frac{dy}{dt} = x + ay\,, \tag{25}$$

$$\frac{dz}{dt} = b + z(x - c)\,.$$

The parameter values are: $a = 0.15$, $b = 0.2$, $c = 10$. The time step is $t = 0.2$ and $m = 8$. We use the y direction to be the time series. The number of data points is $N = 16384$. In Fig. 5 we show the results of GID method of embedding dimension $n = 1, \ldots, 7$. There are many more discontinuities in the unsmoothed curves (Fig. 5a) compared to

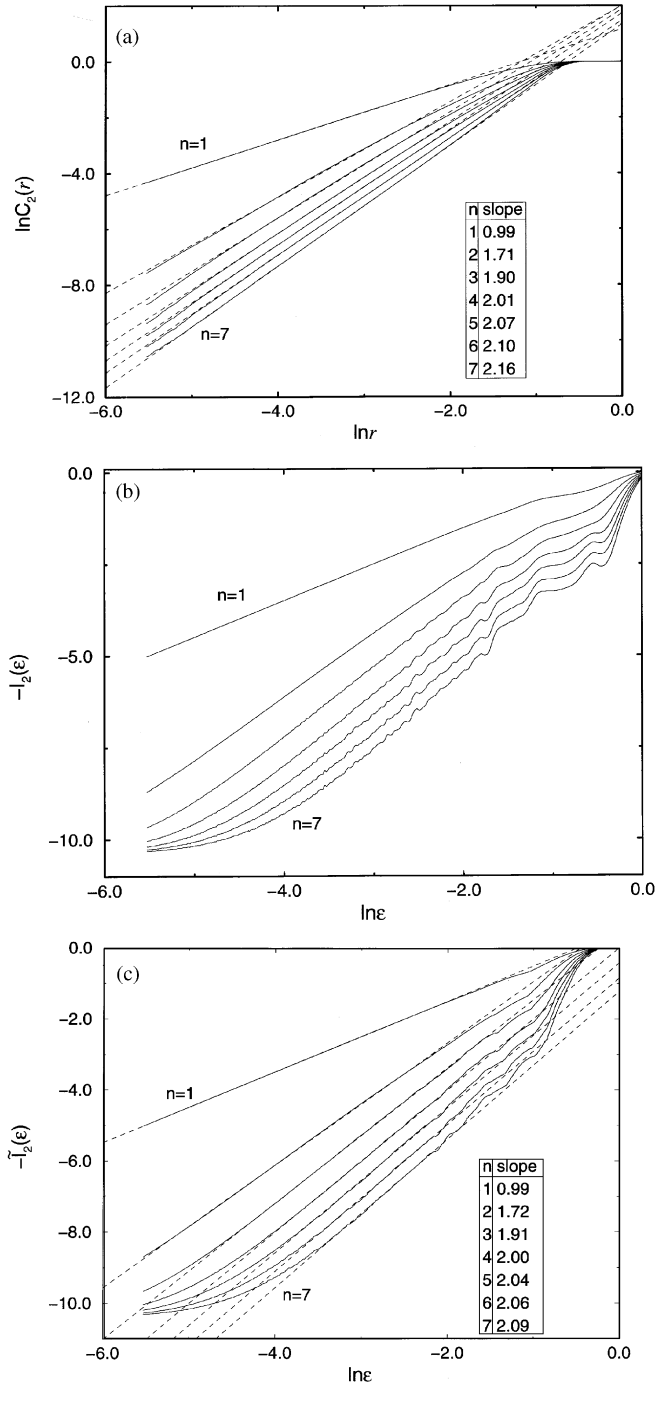

Fig. 4. Correlation and information curves of reconstructed Lorenz attractor. (a) Correlation curves; the estimate slopes are added. (b) Unsmoothed information curves. (c) Smoothed information curves; the estimated linear curves (dashed line) and their slope is added. (d) The slopes of $-\tilde{I}_2(\varepsilon)$ (Fig. 4c). The dashed line represents the estimated FD. (e) FD of Lorenz attractor for different embedding dimensions, n, and for different generalizations, q.

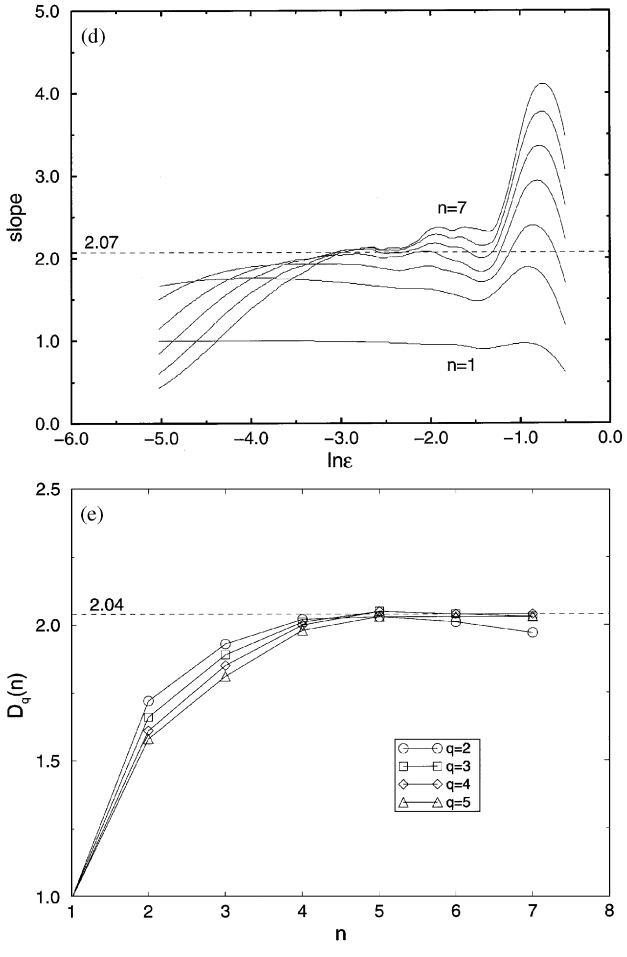

Fig. 4. Continued.

the Lorenz attractor, reflecting a sensitivity to the grid location. The smooth curves in Fig. 5b are produced after 20 comparisons. Again, we approximate the slopes of central linear part of the curves in two different ways and find dimension 2.05, which is in agreement with dimension 2.03 calculated by the use of the GCD method, and through the Lyapunov exponent [33].

4.3. Van der Pol oscillator

Another example that will be tested is the van der Pol oscillator [35,36]. This system was investigated in the beginning of the century in order to find a model for the heart beat variability (among other uses of this equation). The behavior of the system (with parameter values that will be used) is chaotic although it looks quite periodic. Thus, we expect a smooth behavior of the geometrical structure of the attractor. The equation

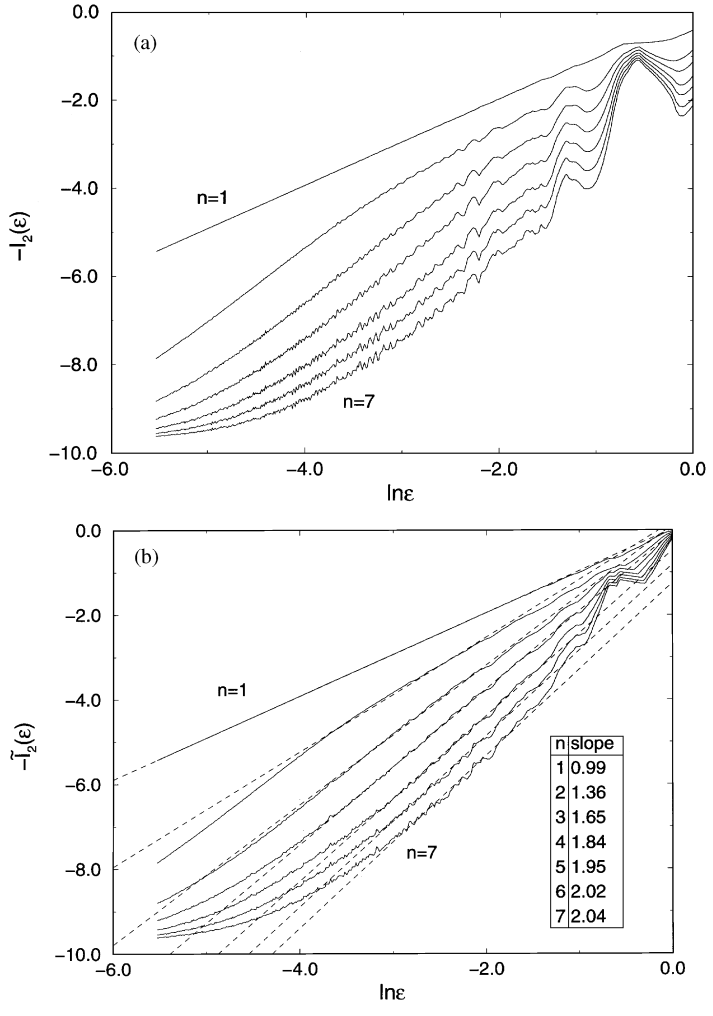

Fig. 5. (a) Unsmoothed graphs of Rossler attractor. (b) Smoothed information graphs; the FD estimation is added.

of motion is

$$\ddot{x} - \alpha(1 - x^2)\dot{x} + kx = f \cos \Omega t \ . \tag{26}$$

The parameter values are: $\Omega = 2.446$, $\alpha = 5$, $k = 1$ and $f = 1$. Habib and Ryne [37] and others [38] have found that the Lyapunov exponents of the system are $\lambda_1 \approx 0.098$, $\lambda_2 = 0$ and $\lambda_3 \approx -6.84$, and thus, according to the Kaplan and Yorke formula [1],

$$D_L = j + \frac{\sum_{i=1}^{j} \lambda_i}{|\lambda_{j+1}|} \approx 2 + \frac{0.098}{6.84} \approx 2.014 \ , \tag{27}$$

where j is defined by the condition that $\sum_{i=1}^{j} \lambda_i > 0$ and $\sum_{i=1}^{j+1} \lambda_i < 0$. The fact that the system has a "periodic" nature is reflected in this very low FD (as known, the minimal FD of a chaotic system is 2).

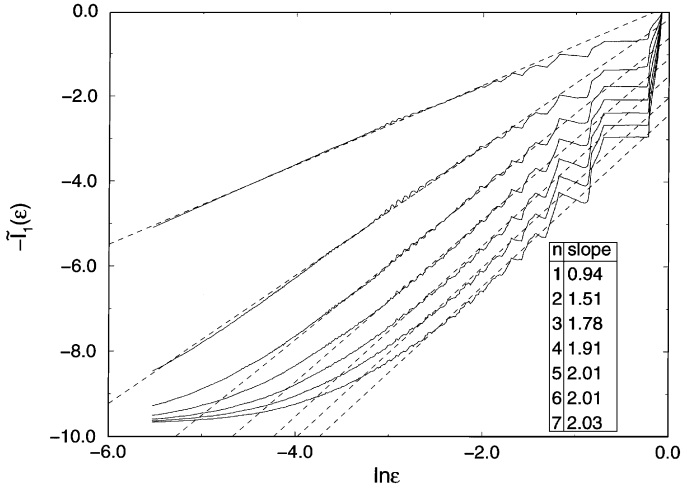

Fig. 6. van der Pol oscillator: the smoothed information graphs, $-\tilde{I}_1(\varepsilon)$; the estimated FD is added (dashed lines with their corresponding slope).

In Fig. 6 we present the calculation of D_1 which we suppose to be close to D_L. The approximate slope leads to an FD of $D_1 \sim 2.02$, which is in good agreement with D_L. Notice that minimization of $I_1(\varepsilon)$ did not succeed for large ε values although we used 7 grid comparisons; that fact can cause a small error in the slope estimation, since the slope is usually determined from the central part of the curves.

4.4. Mackey–Glass equation

Up to now we examined the relation between GID and GCD on systems with finite dimension, for which their FD can be calculated by using methods based on the true integrated vector of the system. A class of systems which is more close to reality are systems with infinite dimensional dynamics. A delay differential equation of the type

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = F(x(t), x(t-\tau)) \tag{28}$$

belongs to this class. In order to solve this kind of equation, one needs to initiate $x(t)$ in the range $0 \leqslant t \leqslant \tau$, and then, by assuming that the system can be represented by a finite set of discrete values of $x(t)$, it is possible to evaluate the system dynamics step by step. The solution is considered to be accurate if one converges to the same global properties, such as Lyapunov exponents and FD, by different methods [39], and without dependence on the number of intervals to which the function is divided [16][5]. Delay

[5] Note that the dynamics which is calculated by the use of different methods, or, equivalently, by a different number of discrete integration intervals, does not have to behave identically. In fact, one can expect similar dynamics if the system is not chaotic, but, if the system is chaotic, it is sensitive to initial conditions and to the integration method, and even moreover, to integration accuracy [40,41]. However, the global properties of a system converge to the same values, by using different integration methods and using different integration accuracy [41].

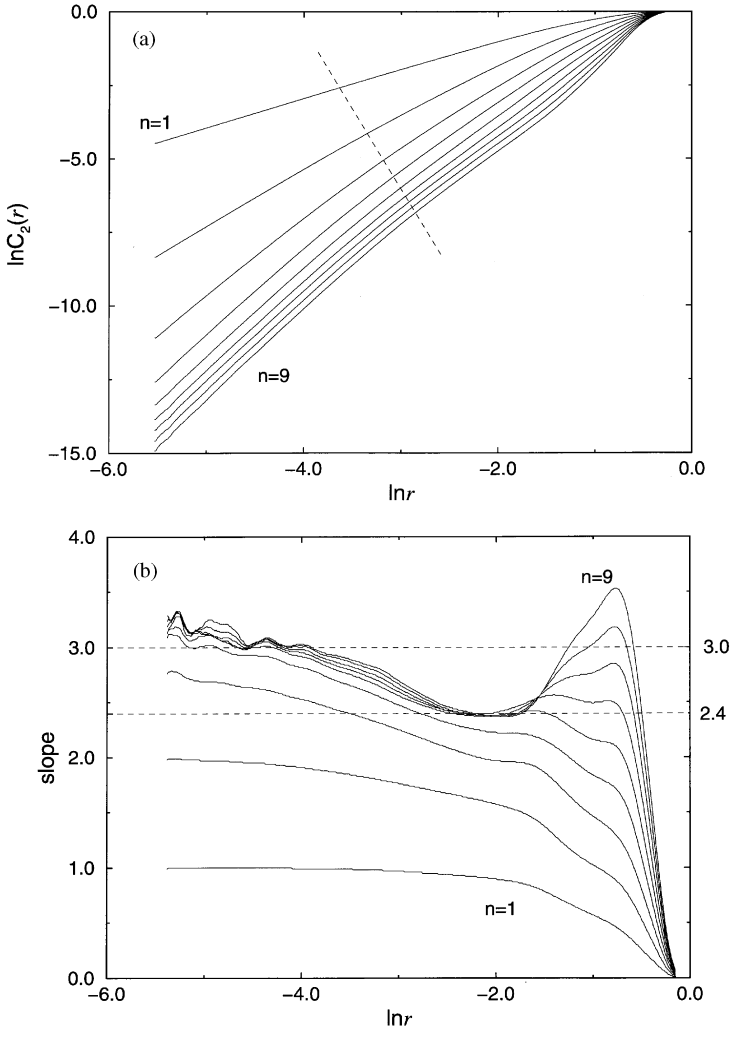

Fig. 7. Mackey–Glass equation: (a) The GCD graphs. One notices two regions of parallel curves, which lead to different FDs. (b) The successive slopes of a. (c) The GID graphs. (d) The successive slopes of c.

equations, such as Eq. (28), describe systems in which a stimulus has a delay response. There are many practical examples from control theory, economics, population biology, and other fields.

One of the known examples is the model of blood cell production in patients with leukemia, formulated by Mackey and Glass [42]:

$$\dot{x}(t) = \frac{ax(t-\tau)}{1+[x(t-\tau)]^c} - bx(t)\ . \tag{29}$$

Following Refs. [16,39], the parameter values are: a=0.1, b=0.2, and c=10. We confine ourselves to $\tau = 30.0$. As in Ref. [16], we choose the time series to be $\{x(t), x(t+\tau),$

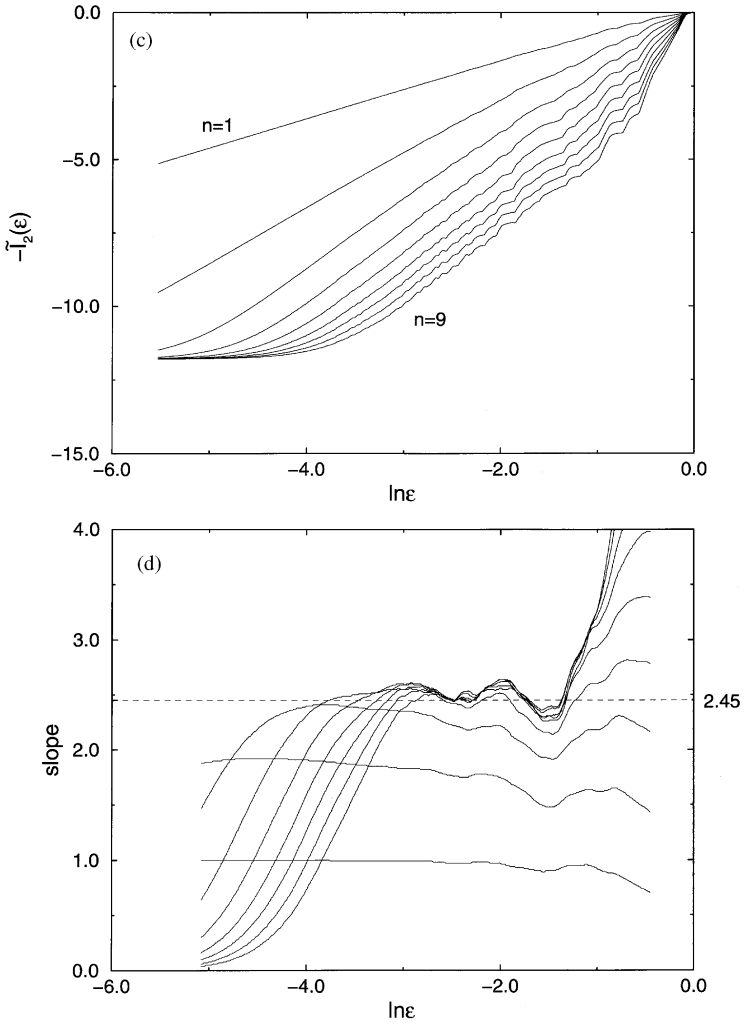

Fig. 7. Continued.

$x(t+2\tau),\ldots\}$,[6] as well as the integration method which is described in this reference. The length of the time series is $N = 131072$.

The FD (D_2) calculation of the Mackey–Glass equation is presented in Fig. 7. The embedding dimensions are, $n = 1,\ldots,9$. In Fig. 7a, the correlation function, $C_2(r)$ is shown. One notices that there are two regions in each one of which there is convergence to a certain slope. These regions are separated by a dashed line. In Fig. 7b, the average local slope of Fig. 7a is shown. One can identify easily two convergences, the first

[6] In fact, the common procedure of reconstructing a phase space from a time series is described in Eq. (1). According to this method, one has to build the reconstructed vectors by the use of a jumping choice m which is determined by the first zero of the autocorrelation function, or the first minimum of the mutual information function [8]. However, we used the same series as in Ref. [16] in order to compare results.

in the neighborhood of ~ 3.0, and the second around ~ 2.4. Thus, there are two approximations for the FD, D_2, pointing to two different scales. The first approximation is similar to the FD that was calculated in Ref. [16]. However, the GID graphs which are presented in Fig. 7c and 7d lead to an FD, $D_2 \sim 2.45$, which seems to be very close to the second convergence of Fig. 7b. Notice that the convergence to $D_2 \sim 2.4$ appears, in both methods, in the neighborhood of the same box size (~ 2.1).

5. Summary

In this work we develop a new algorithm for calculation of a general type of information, $I_q(n)$, which is based on string sorting (the method of string sort can be used also to calculate the conventional GCD method). According to our algorithm, one can divide the phase space into 255 parts in each hyper-box edge. The algorithm requires $O(N \log_2 N)$ computations, where N is the number of reconstructed vectors. A rough estimate for the number of points needed for the FD calculation was given. The algorithm, which can be used in a regular system with known equations of motion, was tested on a reconstructed phase space (which was built according to Takens theorem). The general information graphs have non-monotonic curves, which can be smoothed by the requirement for minimum general information. We examine our algorithm on some well known examples, such as, the Lorenz attractor, the Rossler attractor, the van der Pol oscillator and others, and show that the FD that was computed by the GID method is almost identical to the well-known FDs of those systems.

In practice, the computation time of an FD using the GID method, was much less than for the GCD method. For a typical time series with 32768 data points, the computation time needed for the GCD method was about nine times greater than the computation time of GID method (when we do not restrict ourselves to small r values and we compute all N^2 relations). Thus, in addition to the fact that the algorithm developed in this paper enables the use of comparative methods (which is crucial in some cases), the algorithm is generally faster.

Acknowledgements

The author wishes to thank J. Levitan, M. Lewkowicz and L.P. Horwitz for very useful discussions.

References

[1] J.L. Kaplan, J.A. Yorke, in: H.-O. Peitgen, H.-O. Walther (Eds.), Functional Differential Equations and Approximations of Fixed Points, Lecture Notes in Mathematics 730, Springer, Berlin, 1979.
[2] A. Babloyantz, C. Nicolis, M. Salazar, Phys. Rev. A 111 (1985) 152.
[3] A. Babloyantz, A. Destexhe, Proc. Nat. Acad. Sci. USA 83 (1986) 3513.
[4] E. Basar (Ed.), Chaos in the Brain, Springer, Berlin, 1990.

[5] G. Mayer-Kress, F.E. Yates, L. Benton, M. Keidel, W.S. Tirsch, S.J. Poepple, Math. Biosci. 90 (1988) 155.
[6] K. Saermark, J. Lebech, C.K. Bak, A. Sabers, in: E. Basar, T.H. Bullock (Eds.), Springer Series in Brain Dynamics, Springer, Berlin, 1989, p. 149.
[7] K. Saermark, Y. Ashkenazy, J. Levitan, M. Lewkowicz, Physica A 236 (1997) 363.
[8] F. Takens, in: D. Rand, L.S. Young (Eds.), Dynamical Systems and Turbulence, Springer, Berlin, 1981.
[9] H.G. Schuster, Deterministic Chaos, Physik Verlag, Weinheim, 1989.
[10] A.M. Fraser, H.L. Swinney, Phys. Rev. A 33 (1986) 1134.
[11] J.-C. Roux, R.H. Simoyi, H.L. Swinney, Physica D 8 (1983) 257.
[12] H.D.I. Abarbanel, R. Brown, J.J. Sidorowich, L.S. Tsimring, Rev. Mod. Phys. 65 (1993) 1331.
[13] M. Ding, C. Grebobi, E. Ott, T. Sauer, J.A. Yorke, Physica D 8 (1993) 257.
[14] C.E. Shannon, W. Weaver, The Mathematical Theory of Information, University of ll Press, Urbana, 1949.
[15] J. Balatoni, A. Reny, in: P. Turan (Ed.), Selected Papers of A. Renyi, Akademiai K. Budapest, 1976, 588 p.
[16] P. Grassberger, I. Procaccia, Physica D 9 (1983) 189.
[17] K. Pawelzik, H.G. Schuster, Phys. Rev. A 35 (1987) 481.
[18] P. Grassberger, Phys. Lett. A 148 (1990) 63.
[19] A. Babloyantz, A. Destexhe, in: M. Markus, S. Muller, G. Nicolis (Eds.), From Chemical to Biological Organization, Springer, Berlin, 1988.
[20] L.A. Smith, Phys. Lett. A 133 (1988) 283.
[21] J.-P. Eckmann, D. Ruelle, Physica D 56 (1992) 185.
[22] J. Theiler, Phys. Rev. A 34 (1986) 2427.
[23] W.E. Caswell, J.A. York, in: G. Mayer-Kress (Ed.), Dimensions and Entropies in Chaotic Systems, Springer, Berlin, 1986.
[24] T.C.A. Molteno, Phys. Rev. E 48 (1993) 3263.
[25] X.-J. Hou, R. Gilmore, G. Mindlin, H. Solari, Phys. Lett. A 151 (1990) 43.
[26] A. Block, W. von Bloh, H. Schnellnhuber, Phys. Rev. A 42 (1990) 1869.
[27] L. Liebovitch, T. Toth, Phys. Lett. A 141 (1989) 386.
[28] A. Kruger, Comp. Phys. Commun. 98 (1996) 224.
[29] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, 2nd Edition, Cambridge University Press, Cambridge, 1995.
[30] D.E. Knooth, The Art of Computer Programming, Vol. 3 – Sorting and Searching, Addison-Wesley, Reading, MA, 1975.
[31] A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley, MA, 1983.
[32] E.N. Lorenz, J. Atmos. Sci. 20 (1963) 130.
[33] A. Wolf, J.B. Swift, H.L. Swinney, J.A. Vastano, Physica D 16 (1985) 285.
[34] O.E. Rossler, Phys. Lett. 57A (1976) 397.
[35] B. van der Pol, Philos. Mag. (7) 2 (1926) 978.
[36] B. van der Pol, van der Mark, Philos. Mag. (7) 6 (1928) 763.
[37] S. Habib, R.D. Ryne, Phys. Rev. Lett. 74 (1995) 70.
[38] K. Geist, U. Parlitz, W. Lauterborn, Prog. Theor. Phys. 83 (1990) 875.
[39] J.D. Farmer, Physica D 4 (1981) 366.
[40] J.M. Greene, J. Math. Phys. 20 (1979) 1183.
[41] Y. Ashkenazy, C. Goren, L.P. Horwitz, Phys. Lett. A 243 (1998) 195.
[42] M.C. Mackey, L. Glass, Science 197 (1977) 287.