Lyapunov spectrum from time series using moving boxes

N. N. Oiwa and N. Fiedler-Ferrara
Instituto de Física, Universidade de São Paulo, Caixa Postal 66318, 05315-970, São Paulo, Brazil
(Received 29 August 2001; published 11 February 2002)

We present a very fast algorithm (few seconds) for estimating full Lyapunov spectrum from time series. The method requires a smaller number of parameters than other time-average algorithms, and is tested for data with different numerical precision, sampling frequencies, total sampling times, and presence of noise. We report conclusive results for the electron density broadband fluctuations of a plasma at the edge of tokamak TBR-1.

DOI: 10.1103/PhysRevE.65.036702 PACS number(s): 02.70.−c, 05.45.−a

I. INTRODUCTION

Invariant properties of dynamical systems, such as fractal dimensions, Lyapunov exponents and topological characteristics, have been very useful for the characterization of chaotic time series associated with experimental data [1,2]. In recent years, many methods for computing Lyapunov exponents λ_i have been proposed [3–10]. A brief description and comparison of these methods can be found in the literature [1,2,11,12].

The most traditional methods use time-average approaches [7–10] as follows. Consider the points $\vec{x}(j)$, $j=1,2,\ldots,n$, collected from some continuous orbit in a phase space R^d, using a sampling frequency Δ^{-1}. Now, we extract a subsequence by taking every T_mth point, $T_m \in Z$ and T_m is the evolution lag ($\tau_m = T_m\Delta$ is the evolution time). Then, we suppose that this set of points is an application of a map producing the point $\vec{x}(j+T_m)$ as a function of $\vec{F}(\vec{x}(j))$. The evolution of a small difference $\vec{x}(j+T_m)-\vec{x}(j'+T_m)$ from the orbit is given by

$$\vec{x}(j+T_m)-\vec{x}(j'+T_m)=D\vec{F}(j)[\vec{x}(j+T_m)-\vec{x}(j'+T_m)], \tag{1}$$

where $D\vec{F}(j)$ is a $d\times d$ Jacobian matrix. The product of n_T Jacobian matrices in time is given by

$$J_{n_T}=D\vec{F}(n_T)\cdots D\vec{F}(j)\cdots D\vec{F}(2T_m+1) \times D\vec{F}(T_m+1)D\vec{F}(1). \tag{2}$$

Here, $n_T=n/T_m$. According to the Oseledec multiplicative ergodic theorem [13], if the probability distribution function of the map $\vec{F}$ is invariant, λ_i can be calculated from the eigenvalues of the matrix

$$\Lambda=\lim_{n_T\to\infty}(J^{\dagger}_{n_T}J_{n_T})^{1/2n_T}. \tag{3}$$

Methods that use Eq. (3) have been called time-average algorithms.

On the other hand, attempts to estimate λ_i using spatial averages were made by Kim and Hsu [14] and Boyarsky [15]. More recently, Froyland *et al.* [16] proposed a method using the probability density function associated with the attractor. The invariant measure is approximated on the basis of the deterministic system that is taken as a random dynamical system governed by a finite-state Markov chain. In order to define λ_i of the Markov chain, a random matrix is associated with each state. The obtained invariant density assigns a weight to each portion of the piecewise linear map. The triangularization method of Delauncy [17] is used to provide an approximation of the true system. On each simplex the map is approximately linear, and exponents are spatially averaged by weighting areas of high density more heavily than areas of low density.

The available methods to estimate λ_i require huge computer processing times and a subtle adjustment of parameters. The method proposed in this paper adapts phase space-average techniques, permitting an easier and faster estimate of λ_i from an experimental time series. Although the phase space is partitioned as in the space-average methods, our algorithm is still a time-average method, since the Lyapunov spectrum is calculated from Eq. (3). The present algorithm is to λ_i estimation what Crutchfield and McNamara's altas method [18] is to local prediction techniques.

In this paper the algorithm is described in Sec. II. The procedure for its application is given in Sec. III. Comparisons with traditional time-average methods are presented in Sec. IV, where we also report numerical tests concerning the sensitivity of the algorithm to the variation of the parameters and characteristics of the analyzed data. Furthermore, the algorithm is applied to experimental data of edge electron density broadband fluctuations in the TBR-1 tokamak, and the results are compared with those provided by usual time-average methods. Finally, conclusions are given in Sec. V. The algorithm of moving boxes, a procedure for the division of the phase space, is explained in detail in the Appendix.

II. THE ALGORITHM

Assume the time series $\{x_j\}$ given by integers $x_j=x(t_0+j\Delta)$, $j=1,\ldots,n$, corresponding to a total sampling time $t=n\Delta$. The reason for using integers is that the computation is faster than with the real format. Moreover, the definition of boxes and subspaces edges are much more precise using integers. Also, outputs of analog-to-digital converters, which usually measure the data in nonlinear experiments, are binary numbers converted into integers.

Takens' time delay reconstruction method [19] is applied to the time series, such that

 ©2002 The American Physical Society

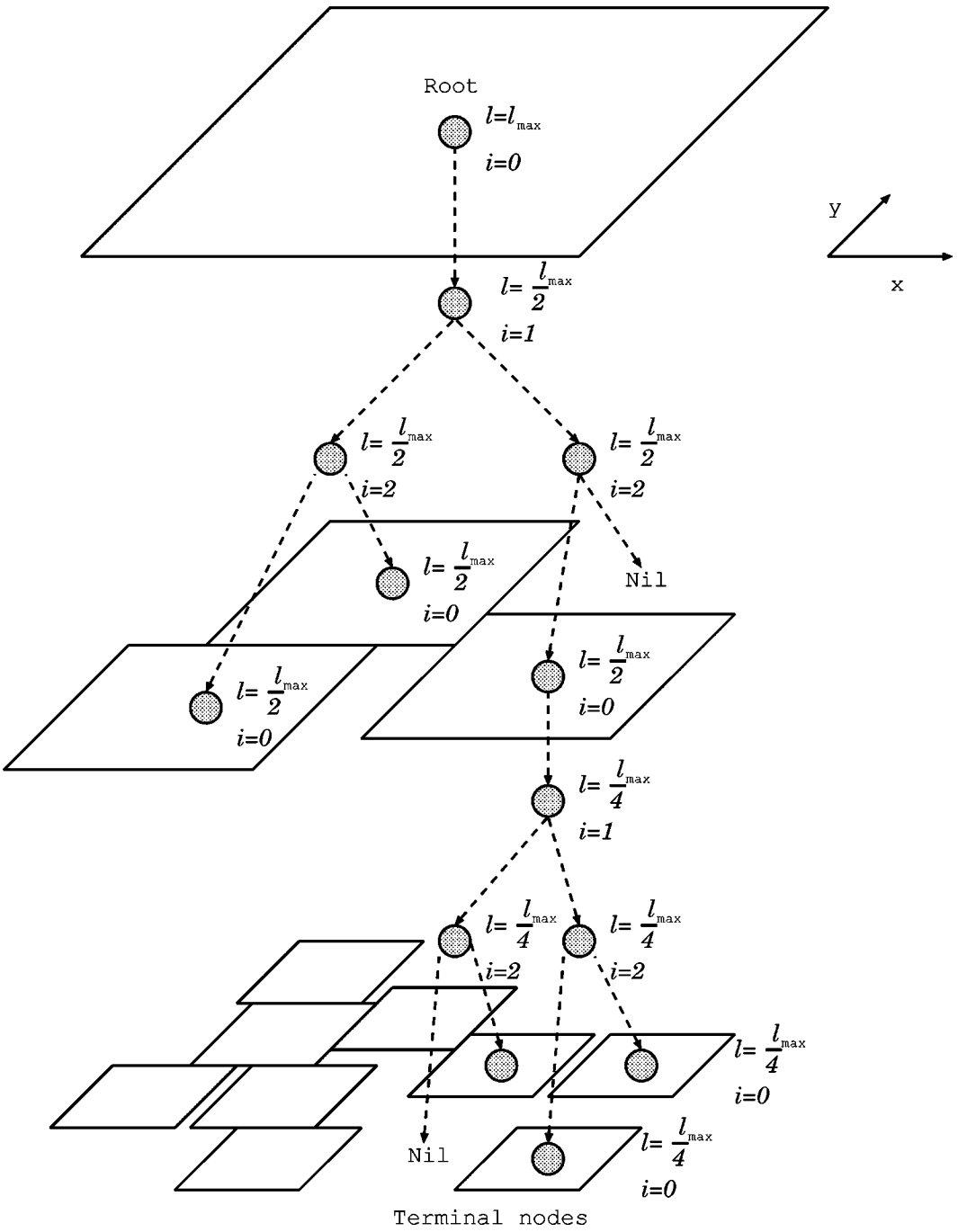

FIG. 1. Binary tree for a two-dimensional attractor. The smallest boxes' value $\ell_{min}=\ell_{max}/4$. In this case, the depth of the tree is defined by ℓ and i, $i=0,1,2$.

$$\vec{\xi}_j=(x_j,x_{j+T_d},\ldots,x_{j+(d-1)T_d}), \tag{4}$$

where T_d is the reconstruction lag ($\tau_d=T_d\Delta$ is the reconstruction time), and d is the embedding dimension.

An important step in our method is an efficient procedure for defining neighborhoods over the attractor. Neighborhoods are a set of boxes or hypercubes B_s, $s=1,\ldots,N$, with size $\ell_{min}\in Z$, covering the attractor (Fig. 1). Boxes are disjoint, and the whole attractor is represented by X, where $X=\cup_s B_s$. Although the attractor is usually defined as a set of points (time average), we can also represent it as a set of boxes (spatial average), since we suppose that the signal is ergodic. A particular box B_s of the attractor $\{\vec{\xi}_j\}$ will be localized by the tables $\pi(\vec{\xi}_j)$. $\pi(\vec{\xi}_j)$ is an array with length n, where each element of the array localizes the box B_s giving a point $\vec{\xi}_j$ of the time series, $B_s=\pi(\vec{\xi}_j)$. On the other hand, we can find a set of points of box B_s, using $\pi^{-1}(B_s)$. $\pi^{-1}(B_s)$ is an index with length N, where each element is a table with the $n_P(B_s)$ points that belongs to this box B_s. Here, $\pi^{-1}(B_s)=\{\vec{\xi}_j\in B_s\}$. The algorithm for the division of

the phase space in boxes and the construction of $\pi(\vec{\xi}_j)$ and $\pi^{-1}(B_s)$, which is called moving boxes, is presented in the Appendix. While the traditional methods [7,9,10] find neighboring points using searching routines, the proposed method does not require a searching procedure, since neighboring points are given by $\pi^{-1}(B_s)$. In the previous methods, the neighborhood is a hypercircle or cube centered in a point of the orbit of the attractor. In our algorithm, the neighbors are a group of points defined by the box B_s. These indexes have the same role as π and π^{-1} of the method of Eckmann *et al.* [9], but they have a different meaning. In our case π^{-1} may indicate more than one point x_j, while for Eckmann *et al.* π^{-1} points just for one. We use a computer technique called ''dynamic storage allocation'' for building $\pi(\vec{\xi}_j)$ and $\pi^{-1}(B_s)$ [20].

In order to increase the performance of the computer code, the Jacobian matrices associated with the box B_s are approximated by a unique matrix $D\vec{F}(B_s)$,

$$D\vec{F}(j) \approx D\vec{F}(B_s), \quad \vec{\xi}_j \in B_s. \tag{5}$$

Now we can rewrite the noncommutative product in Eq. (2) as

$$J_{n_T} = \Pi_{j=1}^{n_T} D\vec{F}(B_s), \quad \vec{\xi}_j \in B_s. \tag{6}$$

In contrast with Eq. (3), where $n_T \to \infty$, in this last equation n_T is finite because the analyzed data $\{x_j\}$ is finite.

Another problem faced by using a naive method for estimating Lyapunov exponents is the overflow when matrices are multiplied in Eq. (6) [22]. This is avoided introducing the triangularization

$$D\vec{F}(B_s) Q_{j-T_m} = Q_j R_j, \quad \vec{\xi}_j \in B_s, \tag{7}$$

where Q_j and Q_{j-T_m} are orthogonal matrices, and R_j is an upper triangular matrix. Although Eckmann and Ruelle [22] suggest the Householder reorthogonalization process at the Jacobian triangularization due to its precision for the eigenvalue estimation [23], we prefer the algorithm of Gram-Schmidt, since it is four times faster than Householder [23]. So,

$$J_{n_T} = Q_{n_T} P_{n_T}, \quad P_{n_T} = R_{n_T} \cdots R_{2T_m+1} R_{T_m+1} R_1. \tag{8}$$

The product of triangular matrices R_j is an upper triangular matrix P_{n_T}, whose elements of the main diagonal are the product of the R_j main diagonal elements. Then, we can rewrite Eq. (3) as $\Lambda = (P_{n_T}^{\dagger} P_{n_T})^{1/2n_T}$, and the associated Lyapunov exponents are

$$\lambda_i = \frac{1}{n_T \tau_m} \sum_{j=1}^{n_T} \ln r_{ii}(j), \tag{9}$$

where $r_{ii}(j)$ are diagonal elements of the matrix $R(j)$.

Now, we introduce $n_P(B_s)$, which gives how many times the box B_s is visited by the orbit of the attractor as well as the number of Jacobian matrices with the same box B_s. Since $X = \cup_s B_s$, $n_P(X)$ gives how many times all boxes are visited. The probability density function $P(B_s)$ can be also estimated as $P(B_s) = n_P(B_s)/n_P(X)$.

Since there are boxes with $n_v < d+1$, we may not estimate all Jacobian matrices $D\vec{F}(j)$ over the whole attractor. Consequently, the sum in Eq. (9) has $n_T \leqslant n/T_m$ elements. On the other hand, $n_P(X)$ is also the number of used Jacobian matrices, because each box B_s is visited $n_P(B_s)$ times. So, $n_T = n_P(X)$. Moreover, there are $n_P(B_s)$ visited elements in $\pi^{-1}(B_s)$. Thus, the sum $\Sigma_{s=1}^{N} \Sigma_{\vec{\xi}_j \in \pi^{-1}(B_s)}$ has $n_P(X)$ elements, and Eq. (9) can be rewritten as $\lambda_i = [1/n_P(X)\tau_m] \Sigma_{s=1}^{N} \Sigma_{\vec{\xi}_j \in B_s} \ln r_{ii}(j)$. When we introduce the visitation number $n_P(B_s)$ of box B_s, the Lyapunov exponents are given by

$$\lambda_i = \sum_{s=1}^{N} P(B_s) \overline{\lambda}_i, \quad \overline{\lambda}_i(B_s) = \frac{1}{n_P(B_s)\tau_m} \sum_{\vec{\xi}_j \in B_s} \ln r_{ii}(j), \tag{10}$$

where $\overline{\lambda}_i(B_s)$ measures the local divergency of the attractor inside the box B_s.

In order to estimate $r_{ii}(j)$ in Eq. (10), we need to calculate the Jacobian matrices $D\vec{F}(B_s)$. Consider a set $\{\vec{\xi}_j\}$ of points belonging to the box B_s,

$$\{\vec{z}(j)\} = \{\vec{\xi}_j - \langle \vec{\xi}_j \rangle : \vec{\xi}_j \in B_s\}, \tag{11}$$

where $\langle \vec{\xi}_j \rangle$ is the average position of the points inside B_s, and $\vec{z}(j)$ is the distance between $\vec{\xi}_j$ and $\langle \vec{\xi}_j \rangle$. After an evolution lag T_m, all points into the box B_s from $\{\vec{\xi}_j\}$ to $\{\vec{\xi}_{j+T_m}\}$ and the point $\langle \vec{\xi}_j \rangle$ evolves to $\langle \vec{\xi}_{j+T_m} \rangle$. Therefore, the set $\{\vec{z}(j)\}$ can be mapped into

$$\{\vec{z}(j+T_m)\} = \{\vec{\xi}_{j+T_m} - \langle \vec{\xi}_{j+T_m} \rangle : \vec{\xi}_j \in B_s\}. \tag{12}$$

If the hypercube is small enough ($\ell_{\min} \ll \ell_{\max}$) the spatial evolution from $\{\vec{z}(j)\}$ to $\{\vec{z}(j+T_m)\}$ will be mapped by

$$\vec{z}(j+T_m) \approx D\vec{F}(B_s)\vec{z}(j). \tag{13}$$

We do not take higher-order terms in the Taylor expansion as in the algorithm of Brown *et al.* [10] because they do not provide better results. The matrix $D\vec{F}(B_s)$ in Eq. (13) is estimated using the least squares method [24,25]. If $D\vec{F}(B_s)$ is nondegenerated, Eq. (13) has a solution for $n_v \geqslant d+1$. Despite many criteria for n_v in the literature [7–10], the systematic application of our algorithm reveals that the values of n_v do not affect significantly the performance of the computer code. As a consequence, we take all n_v points into B_s for estimating $D\vec{F}(B_s)$.

The algorithm for the phase space division is based on the *k-d* tree algorithm proposed by Friedman, Bently, and Finkel [26], where the attractor is divided into boxes with different sizes [27,28]. The method of Brown *et al.* searches neighboring points by making use of this algorithm. It is a recursive algorithm, and the data structure is a binary tree [20], where

each node is associated with one box or one particular region of the reconstructed phase space. We adapt the algorithm of Friedman *et al.* to cover the attractor with hypercubes of the same length and replace their searching procedure by $\pi(\vec{\xi}_j)$ and $\pi^{-1}(B_s)$ (see the Appendix). The root of the tree is a single hypercube with size $\ell_{\max}=2^{n_b}$ covering the whole attractor (n_b is the number of bits defining integers of the time series). Starting from the root, every box splits into a pair of boxes of the next generation. When there is free space, boxes might move into the reconstructed phase space after each division minimizing the number of boxes covering the attractor. If there are not points in a particular box, the empty box and the associated branch of the binary tree are eliminated, Fig. 1. This algorithm can be used for other analyses where an attractor is covered by a set of boxes. For example, in the case of multifractal analysis, it significantly diminishes the errors [21]. Finally, the boxes of the last generation with size $\ell_{\min}$ list all points ξ_j inside of B_s.

Along general lines, the structure of the algorithm can be summarized as follows: (i) Takens's reconstruction from the time series; (ii) covering of the attractor using moving boxes; (iii) construction of the index tables $\pi(j)$ and $\pi^{-1}(s)$; (iv) estimation of Jacobian matrices through the least squares method, with subsequent triangularization using the Gram-Schmidt reorthogonalization; and (v) estimation of the Lyapunov spectrum from Eq. (10).

III. ADJUSTMENT OF THE PARAMETERS

A first estimate of the adjustable parameters $\ell_{\min}$, τ_m, and τ_d is straightforward. The size $\ell_{\min}$ of the boxes should be 1% of the length L of the attractor [8,9], defined as the standard deviation of the mean associated with the signal $\{x_j\}$. The times τ_m and τ_d should be taken of order of the autocorrelation time τ_c, the time lag such that the linear autocorrelation function is half of its maximum value. Another suitable choice for Takens' reconstruction time is when the autocorrelation function first crosses zero [2,12,29], in spite of other more elaborated prescriptions [30–32]. The embedding dimension for the reconstruction should be $d>2d_A$, where d_A is the dimension of the attractor. Nevertheless, this dimension d is not always necessary [2] to provide an Euclidean space where the set of points can be unfolded without ambiguity. Other criteria, like false nearest neighbors [33], may be used. Anyway, the largest positive Lyapunov exponent does not change using higher embedding dimensions, despite the presence of spurious exponents [9,12]. Concerning the number n of points in the time series, there are some prescriptions in the literature [7–10,34]. However, we would prefer to apply the simplest of them: the total sampling time should be such that the non-negative exponents of the spectrum at the $\lambda_i \times t$ graph converge. It is an important remark that frequently the negative exponents do not exhibit convergence. But, according to the Osedelec theorem [13], the convergence is expected only for non-negative exponents. Moreover, the negative exponent is usually far from the expected value, reflecting the usual difficult to estimate numerically λ_i along contracting directions.

Nevertheless, Lyapunov exponents usually demand a more elaborated procedure than take $\tau_m=\tau_d=\tau_c$, and $\ell_{\min}/\ell_{\max}\approx 0.01$. If these values are taken, we will have illusory small errors. On the other hand, when graphs varying $\ell_{\min}$, τ_m, and τ_d are constructed, plateaus can be identified for non-negative exponents. These stable regions are more pronounced for larger exponents. If the signal is strongly contaminated by noise or other perturbation, these flat regions disappear, and the estimate of Lyapunov exponents is impossible.

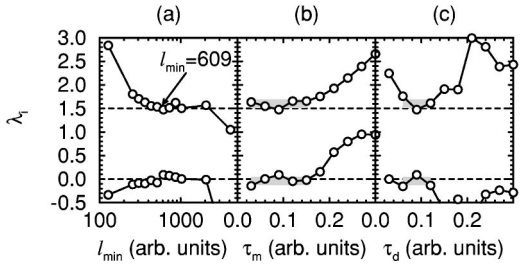

FIG. 2. Lyapunov spectra for the Lorenz flux with $R=45.92$, $\sigma=16.0$, $b=4.0$, $\Delta=0.03$, $n=64\,000$, $\tau_c=0.09$, $\tau_p=0.30$, and $L=13\,468$. The graphs show λ_i varying (a) the total sampling time t, $n=t\Delta^{-1}$; (b) the box size $\ell_{\min}$; (c) the evolution time τ_m; and (d) the reconstruction time τ_d. The dashed lines indicate the expected values. The arrow in (a) indicates the best box size, $\ell_{\min}=609$, used to estimate λ_i in (b) and (c). Dotted lines show the expected values. The gray rectangles indicate the flat regions of spectra where values were used to estimate the λ_i in Table I.

We recommend adjusting the size of the box $\ell_{\min}$ first. Our numerical tests show that the best $\ell_{\min}$ corresponds to the smallest plateau value in the $\lambda_i \times \ell_{\min}$ graph. In case such a stable region does not exist, the most suitable $\ell_{\min}$ is the one that minimizes the largest Lyapunov exponent λ_1. According to Albano *et al.* [29] and our numerical tests, a suitable embedding window for τ_d is the first minimum of the autocorrelation function τ_p. We adopt this prescription for the $\lambda_i \times \tau_m$ graph, too, because we always find the plateaus in this interval of time. However, the best choice for $\ell_{\min}$, τ_m, and τ_d together are the values which maximize these plateaus. The value and the error for λ_i are obtained taking the mean and the standard deviation of the mean of the plateau values.

By way of illustration, take the Lorenz flow [35]

$$(\dot{x},\dot{y},\dot{z})=\sigma(y-x),-xz+Rx-y,xy-bz, \qquad (14)$$

with $R=45.92$, $\sigma=16.0$, and $b=4.0$. Parameters of the time series are $\Delta=0.03$, $n=64\,000$, $\tau_c=0.09$, $\tau_p=0.30$, and $L=13468$. Figure 2 shows a typical graph set for the Lyapunov spectrum analysis. Expected values for the Lyapunov exponents according to the literature are represented by dashed lines. We use $\ell_{\min}=609$, Fig. 2(a), to obtain Figs. 2(b) and 2(c). In these figures, τ_m and τ_d are varied around the autocorrelation time $\tau_c=0.09$. Figure 2(b) is obtained using $\tau_d=0.09$ and Fig. 2(c) makes use of $\tau_m=0.09$. Plateaus are identified in $0.03\leqslant\tau_m\leqslant 0.15$ and $0.06\leqslant\tau_d\leqslant 0.12$, indicated in gray in Figs. 2(b) and 2(c). Average values and standard deviations of this spectrum are in Table I.

TABLE I. Lyapunov exponents spectra using methods of Wolf *et al.* (WSSV), Sano-Sawada (SS), Eckmann *et al.* (ERKC), Brown *et al.* (BBA), and moving boxes.

	Expected value	WSSV	SS	ERKC	BBA	Moving boxes
$x_{j+1}=1-2x_j^2$	$\ln 2=0.6931\ldots$ [a]	0.99 ± 0.01	0.6919 ± 0.0016	0.689 ± 0.007	0.694 ± 0.008	0.698 ± 0.005
Hénon	0.4185 ± 0.0005	0.579 ± 0.004	0.404 ± 0.009	0.418 ± 0.006	0.420 ± 0.002	0.420 ± 0.002
$a=1.4,b=0.3$	-1.6225 ± 0.0005 [b]		-1.57 ± 0.02	-1.48 ± 0.04	-1.61 ± 0.02	-1.63 ± 0.08
Lorenz	1.37 ± 0.01	2.25 ± 0.12	1.32 ± 0.05	1.50 ± 0.08	1.45 ± 0.04	1.47 ± 0.09
$R=40.00,\sigma=16.0$	-0.011 ± 0.005		-0.02 ± 0.07	0.00 ± 0.13	0.00 ± 0.05	-0.11 ± 0.05
$b=4,\Delta=0.03$	-22.359 ± 0.006 [b]		-13.9 ± 3.7	-17.3 ± 7.1	-14.6 ± 9.8	-14.7 ± 2.7
Lorenz	1.507 ± 0.003	2.0 ± 0.1	1.42 ± 0.06	1.66 ± 0.08	1.53 ± 0.02	1.60 ± 0.09
$R=45.92,\sigma=16.0$	-0.001 ± 0.001		-0.01 ± 0.12	0.01 ± 0.18	0.01 ± 0.03	-0.04 ± 0.09
$b=4,\Delta=0.03$	-22.499 ± 0.002 [b]		-14.7 ± 4.5	-19.9 ± 6.3	-19.1 ± 8.4	-14.6 ± 2.5
Rössler	0.0641 ± 0.0009	0.103 ± 0.007	0.067 ± 0.004	0.067 ± 0.002	0.069 ± 0.004	0.069 ± 0.005
$a=b=0.2,c=5.7,$	-0.0003 ± 0.0004		0.005 ± 0.006	0.001 ± 0.005	0.003 ± 0.004	0.001 ± 0.008
$\Delta=0.12$	-4.982 ± 0.003 [b]		-1.09 ± 0.44	-1.2 ± 0.8	-2.36 ± 0.04	-1.9 ± 0.9
Mackey-Glass	0.0074 ± 0.0007	0.012 ± 0.002	0.009 ± 0.002	0.0054 ± 0.0007	0.006 ± 0.002	0.009 ± 0.001
$a=0.2,b=0.1,$	0.0038 ± 0.0007		0.0025 ± 0.0010	0.0019 ± 0.0012	0.001 ± 0.002	0.0038 ± 0.0008
$c=10,T=30,$	-0.0015 ± 0.0008		-0.003 ± 0.001	-0.0011 ± 0.0014	-0.004 ± 0.002	-0.004 ± 0.001
$\Delta=0.3$	-0.017 ± 0.003		-0.014 ± 0.002	-0.020 ± 0.004	-0.017 ± 0.003	-0.016 ± 0.003
	-0.048 ± 0.001 [c]		-0.049 ± 0.007	-0.039 ± 0.008	-0.042 ± 0.004	-0.044 ± 0.008

[a]Exact value.
[b]Semianalytical method [22].
[c]Reference [8].

IV. RESULTS IN MAPS, FLUX, AND EXPERIMENTAL DATA

We compare our method with some well-known algorithms [7–10] and use the following maps and flows: (i) quadratic map [36], $x_{j+1}=1-2x_j^2$, with $n=1024$ and 14 bits of numerical precision; (ii) Hénon map [37] ($x_{j+1}=1-ax_j^2+y_j, y_{j+1}=bx_j$) with $a=1.4$, $b=0.3$, $n=10\,000$, and 12 bits; (iii) Lorenz flux [35], Eq. (14), using $R=45.92$ or $R=40.00$, $\sigma=16.0$, $b=4.0$, $n=64\,000$, $\Delta=0.03$, and 14 bits; (iv) Rössler flux [38] [$\dot{x}=-y-z, \dot{y}=x+ay, \dot{z}=b+z(x-c)$] taking $a=b=0.2$, $c=5.7$, $n=40000$, $\Delta=0.12$, and 14 bits; and (v) Mackey-Glass flux [39], $\dot{x}=(ax(t+s)/\{1+[x(t+s)]^c\})-bx(t)$, with $a=0.2$, $b=0.1$, $c=10.0$, $s=30$, $n=64\,000$, $\Delta=0.3$, and 16 bits. Differential equations have been integrated using the fourth order Runge-Kutta's method [25]. One particular variable has been taken, and the attractor has been reconstructed using Takens' time delay reconstruction method. For time-average methods we also consider other parameters not present in our algorithm, like the number n_T of Jacobian matrices [40].

Results with our method in Table I agree with those obtained using the algorithms of Sano-Sawada [8] and Eckmann *et al.* [9]. However, its performance is slightly inferior to the method of Brown *et al.* [10]. The algorithm of Wolf *et al.* [7] has many limitations for an efficient estimate of the largest Lyapunov exponent.

We also evaluate the sensitivity of our algorithm concerning characteristics of the time series: the number of points in the data set, the level of noise, the numerical precision and the sampling frequency. Figure 3 shows results obtained varying τ_m and τ_d and averaging over the plateaus values for the Lorenz flux with $R=45.92$, $\sigma=16.0$, $b=4.0$, $n=64\,000$, $\Delta=0.03$, and 14 bits has been used simulating an experimental data.

Figures 3(a)–3(c) give values for λ_i while the total sampling time t is varied. When we vary parameters in search of plateaus, our algorithm converges slowly. We need for the Lorenz flow a number of points of the order of 10 000 for the convergence of λ_1 and λ_2. This value should be confronted with those obtained by Karantonis and Pagitsas [12], who compared performances of algorithms of Sano-Sawada and Eckmann *et al.* using the Lorenz flux and keeping all parameters constant, except the number of data points. Their results indicate a minimum of the order of 10 000 points for both methods using fixed parameters. Besides, when averages on the flat regions of the spectrum is used, the quickness and the noise robustness of our algorithm pay for the slow convergence as we will show along this section.

Effects of noise can be seen in Figs. 3(d) and 3(f). We call noise pseudorandom numbers [24] introduced during the numerical integration of the flow. We define the level of noise as the ratio of the standard deviation of the mean of noise by the typical length of the attractor L. The largest Lyapunov exponent using our algorithm is very robust, supporting until 10% of noise. In spite of the large error, the second Lyapunov exponent λ_2 becomes more negative as the level of noise is increased. The same type of behavior has been observed applying the algorithm of Brown *et al.* to the Lorenz flow with noise [2]. In a previous work [40], we compared some traditional time-average algorithms and concluded the method of Eckmann *et al.* is the most noise robust, followed by the algorithm of Brown *et al.* and Sano-Sawada. While the algorithm of Eckmann *et al.* supports up

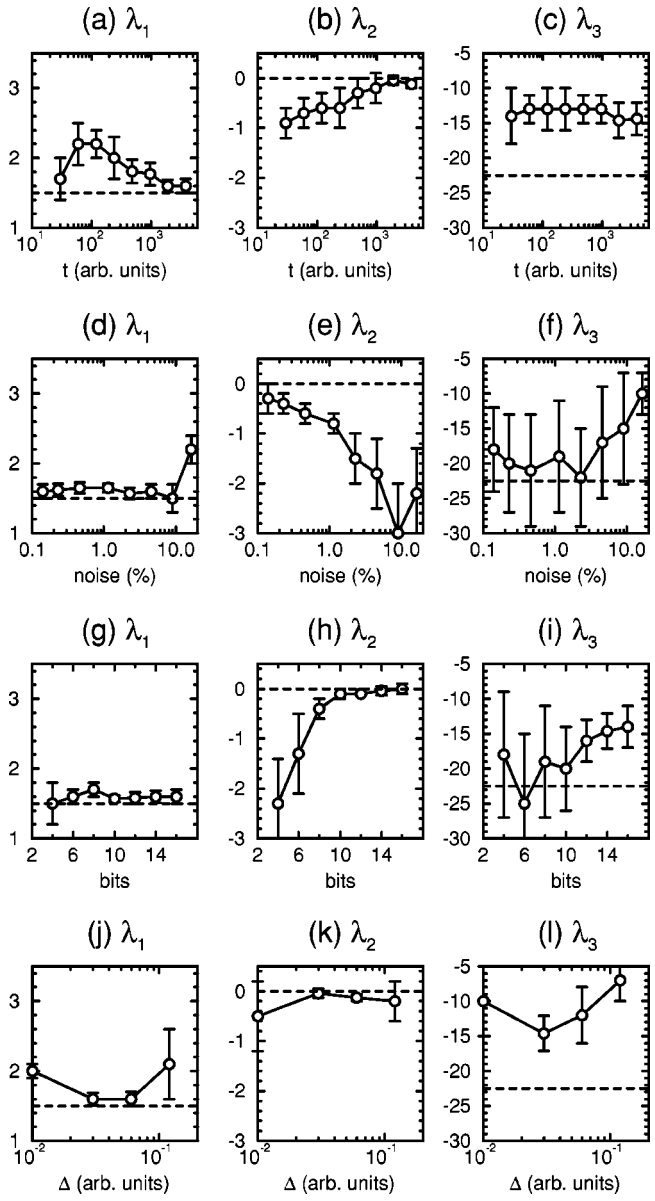

FIG. 3. λ_i varying the total sampling time t (a)–(c), the level of noise (d)–(f), the numerical precision (g)–(i), and the sampling frequency (j)–(l) for the Lorenz flux with $R=45.92$, $\sigma=16.0$, $b=4.0$, $\Delta=0.03$, and 14 bits. λ_i were estimated taking averages of the gray area in Fig. 2. Dashed lines show the expected values. The number of points in the time series is $n=t\Delta^{-1}$.

to 5% of noise, the method of Sano and Sawada gives bad results for signals with more than 1% of noise.

The numerical precision of the data is varied in Figs. 3(g)–3(i). Our algorithm needs data with at least 10 bits of precision, a similar performance compared with the method of Eckmann *et al.* and superior to the algorithms of Brown *et al.* and Sano-Sawada, which need 12 bits [40,42].

We also analyze the effects of the sampling frequency Δ^{-1}. As can be seen in Figs. 3(j)–3(l), $0.03\leqslant\Delta\leqslant0.06$ is the best range for the estimates, which is consistent with an adequate sampling of the signal.

We choose the method of Sano-Sawada to compare the performance of the proposed algorithm because both methods use the Gram-Schmidt procedure as the triangularization process, and both estimate the Jacobian through the least squares method. Although Sano and Sawada do not suggest any algorithm to search neighbor points, we have implemented the quicksort algorithm [24] as in the work of Eckmann *et al.* [9]. According to our experience [40], the methods of Eckmann *et al.*, Bryant *et al.* and Wolf *et al.* need about two, three, and five times more than the CPU time of Sano-Sawada, respectively. We run the computer codes, written in language C, into workstations SUN SPARCstation 5. As can be seen in Fig. 4, our method is between 10 and 100 times faster than the algorithm of Sano-Sawada, depending on the number of points, with a much less pronounced rate of growing. Finally, we also tested the method in a notebook Compaq Presario 1200-XL111, and our algorithm takes only 1.45 sec to estimate λ_i.

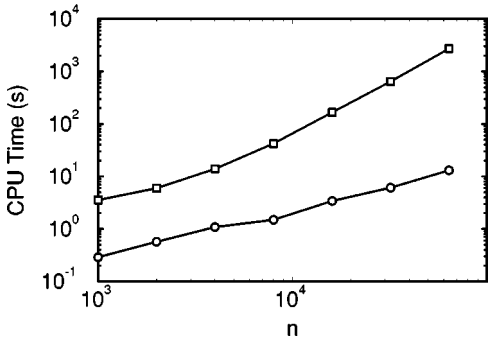

FIG. 4. CPU times for estimating the Lyapunov spectra associated with the Lorenz flux varying the number of points n. The used parameters are $\tau_d=\tau_m=0.09$, $d=3$, and $n_v\geqslant3$, as well as ○, the proposed algorithm and □, the Sano-Sawada's method.

The good performance of the proposed algorithm can be understood. In effect, we have implemented many routines optimizing the data processing. Our procedure does not need search routines, since we use indexes π and π^{-1} to find neighborhoods. Furthermore, we approach all Jacobian matrices associated with a box B_s by a unique average Jacobian matrix, Eq. (5), thus reducing the number of estimated Jacobian matrices. This economy can be exemplified when our method is applied to the Lorenz flux with $R=45.92$, $\sigma=16.0$, $b=4.0$, $\Delta=0.03$, $n=64\,000$, and an evolution lag $T_m=3$. Since $n_T\approx n/T_m$, the method of Eckmann *et al.* uses 21 123 Jacobian matrices. On the other hand, our algorithm computes exponents from 21 213 points covering 949 boxes and using 804 Jacobian matrices. The number of boxes is bigger than the number of Jacobian matrices because $D\vec{F}$ is estimated only from those matrices which do not have degeneracy, Eq. (13). The method of Sano-Sawada needs an even smaller number of Jacobian matrices, since it is fixed in 1000 before processing. However, the performance of our algorithm is faster than the method of Sano and Sawada because it is more efficient to search neighbor points. When we compare to the time-average methods, the number of lines at the computer code has been also optimized in our algorithm through the dynamic storage allocation, the recursive function, and the binary tree.

Finally, we applied the algorithm to experimental data

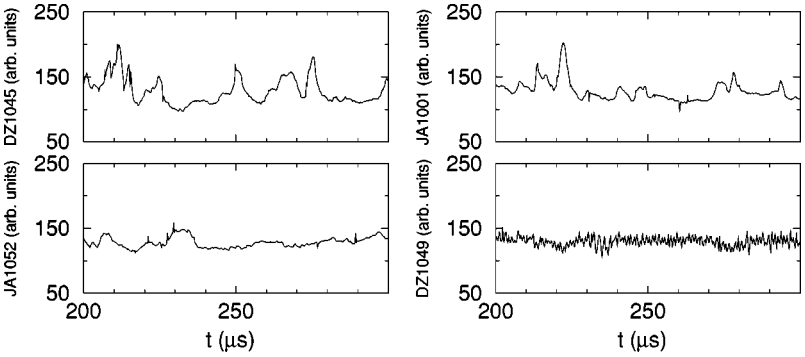

FIG. 5. Set of four analyzed signals: $n=15\,000$, $t=3.00$ ms, $\tau_c=5.00$ μs for DZ1045; $n=12\,000$, $t=3.75$ ms, $\tau_c=5.25$ μs for JA1001; $n=11\,600$, $t=2.90$ ms, $\tau_c=6.50$ μs for JA1052; and $n=12\,000$, $t=3.00$ ms, $\tau_c=0.50$ μs for DZ1049. The signals have been sampling at frequency 4 MHz, $\Delta=0.25$ μs. Each signal is proportional to density broadband fluctuations at the edge of the tokamak TBR-1. The length of the signal is represented by arbitrary units.

from TBR-1. TBR-1 is a small circular cross-section tokamak with a minor radius of 0.11 m and a major radius of 0.30 m, operating with Ohmically-heated plasma with the following typical parameters: toroidal magnetic field $B=4-4.5$ kG, plasma current $4-10$ kA, electron density $(2-10)\times10^{12}$ cm^{-3}, and electron temperature on axis 100 eV. The data consist of local fluctuating signals of the ion saturation current in the triple Langmuir probes placed in the scape-off layer of the tokamak TBR-1 [41]. The signals are proportional to density broadband fluctuations at the edge of the plasma. The associated frequency spectra are wide, covering the range 10–500 kHz. The set of four signals we have analyzed in Fig. 5 represents typical data chosen among the experimental results available.

We have estimated for the largest Lyapunov exponents associated with three of the four signals using our algorithm as well as the methods of Sano-Sawada, Eckmann *et al.*, and Brown *et al.* The Lyapunov exponent analysis for the signal DZ1049 is not conclusive due to the presence of strong signatures of noise [40,42], corroborated by the nonsaturation of the correlation dimension using the algorithm of Grassberger and Procaccia [43].

Figure 6 shows the values for the largest Lyapunov exponents using our method and varying the embedding dimension around the correlation dimension. Each λ_i was calculated considering variations of τ_m and τ_d, as explained in Sec. III. The values indicated in Table II have been obtained taking the mean and the error of the values inside the gray area in Fig. 6.

There is a significant spread in the values for the largest Lyapunov exponents obtained for each signal using different algorithms. The smallest λ_1 we obtain using the proposed algorithm is consistent with its noise robustness. Since noise usually increases the largest Lyapunov exponent, values using other methods should be overestimated.

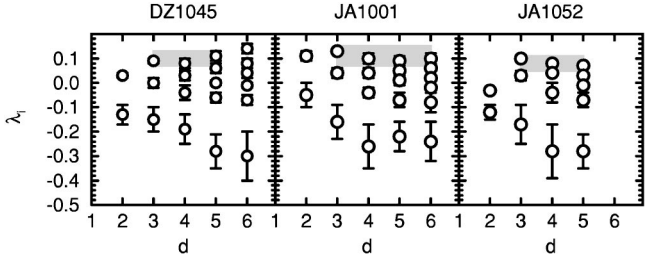

FIG. 6. Lyapunov spectra λ_i in 10^6 s^{-1} using our method for signals DZ1045, JA1001, and JA1052 as a function of the embedding dimension *d*. The values used to estimate the largest Lyapunov exponents are in gray.

V. CONCLUSIONS

We have proposed and tested an algorithm for estimating the Lyapunov exponents from time series. Compared to the traditional time-average methods, the proposed algorithm is faster, more noise robust and uses a smaller number of parameters, simplifying substantially its utilization and giving results comparable to those obtained using the best algorithms. In order to optimize the phase space division into boxes, we constructed a recursive algorithm, where boxes move adapting to the geometry of the attractor. This procedure permits a better and more efficient estimate of the probability density function and the Jacobian matrices inside each box. We also tested estimating Lyapunov exponents from experimental signals of density broadband fluctuations at the edge of the plasma of the tokamak TBR-1, with conclusive results for the largest exponent.

ACKNOWLEDGMENTS

We would like to thank W. P. de Sá, D. F. da Cruz, and R. M. O. Galvão for providing the experimental data analyzed in this paper. The author would like to thank S. R. A. Salinas for the critical reading of the manuscript. We thank Conselho Nacional de Desenvolvimento Tecnológico e Científico (CNPq, Brazil) for financial support.

APPENDIX: THE ALGORITHM OF MOVING BOXES

The binary tree in our algorithm is built using a recursive procedure we call axes_divide. The input variables are asso-

TABLE II. The largest Lyapunov exponent λ_1 in 10^6 s^{-1} for signals DZ1045, JA1001, and JA1052. The exponents were obtained using the algorithms of Sano-Sawada (SS), Eckmann *et al.* (ERKC), Brown *et al.* (BBA), and moving boxes.

Method	DZ1045	JA1001	JA1052
SS	0.19 ± 0.05	0.19 ± 0.05	0.11 ± 0.02
ERKC	0.20 ± 0.05	0.18 ± 0.04	0.07 ± 0.03
BBA	0.15 ± 0.04	0.21 ± 0.06	0.16 ± 0.03
Moving boxes	0.10 ± 0.03	0.11 ± 0.04	0.10 ± 0.03

ciated with the nodes of the binary tree, and these variables control the recursive nature of axes_divide. The description of this variable is given by $\{\vec{\xi}_j\}$, which represents the whole reconstructed attractor or a part of it; the vector $\vec{p}$, $\vec{p} \in Z^d$ represents the center of the box; $\vec{a}$ and $\vec{b}$ are the inferior and superior limits of the reconstructed phase space (at the root) or a subset (at the nth generation) of it, respectively, and they are always redefined along the recursive application of axes_divide. Furthermore, there are the variables controlling the depth of the binary tree: the scalar $\ell \in Z$, which gives the side of the box, and i, $0 \leqslant i \leqslant d$, a variable that indicates the beginning of the attractor division ($i=0$) or the axis which will be divided ($i \neq 0$). We need d generation to divide a hypercube in 2^d boxes, and the tree will have $(k+1)d$ generations from the root to terminal nodes.

The root of the binary tree is one hypercube with size $\ell_{\max}=2^{n_b}$ centered at the middle of the attractor, $\vec{p}=\vec{u}+(\vec{v}-\vec{u})/2$, where $\vec{u}=\min\{\vec{\xi}_j\}$ and $\vec{v}=\max\{\vec{\xi}_j\}$. We compute all $\vec{u}$ and $\vec{v}$ by scanning $\{\vec{\xi}_j\}$ and taking the minimal and maximal values in the most simple way. The limits of the phase space $\vec{a}$ and $\vec{b}$, are simply the numerical precision of the computer, or else these limits are those values sufficient to cover the whole attractor. We take $i=0$ to indicate the beginning of the hypercube division. The input variables of axes_divide, which are associated with the root of the tree, are $\{\vec{\xi}_j\}$, $\vec{p}$, $\vec{a}$, $\vec{b}$, $\ell_{\max}$, and i.

The recursive function axes_divide can be described as follows:

(a) If $\{\vec{\xi}\}=\varnothing$ or $\ell=\ell_{\min}$, nothing will happen, and the procedure axes_divide will return to the calling routine or caller. The conditional expression $\{\vec{\xi}\}=\varnothing$ is a test for the branch elimination. All empty boxes are eliminated using this test. Furthermore, this item has a second function: when the terminal node is the root, we do not need to build the tree, and this situation is identified through the test $\ell=\ell_{\min}$.

(b) If $i=d$ and $\ell \neq \ell_{\min}$, take $i=0$ and call axes_divide making use of $\{\vec{\xi}_j\}$, $\vec{p}$, $\vec{a}$, $\vec{b}$, $\ell/2$ and i as input variables. After this, the procedure is finished. When the division of the box with side ℓ along its d Cartesian axes is finished, we will check in this item, if the box is divided again in 2^d parts or not using the conditional expression $(i=d)\wedge(\ell \neq \ell_{\min})$.

(c) Increment i, $i=i+1$, identifying the axis which will be divided.

(d) Divide the hypercube into two boxes with the same size using a bisector plane in the i axis. Then, we have

$$\{\vec{\xi}_j\}_L=\{\vec{\xi}_j : \xi_{ij}=x_{j+(i-1)T_d} \leqslant p_i\},$$

$$\{\vec{\xi}_j\}_R=\{\vec{\xi}_j : \xi_{ij}=x_{j+(i-1)T_d} > p_i\}, \tag{A1}$$

where ξ_{ij} and p_i are, respectively, the components i of vectors $\vec{\xi}_j$ and $\vec{p}$. These equations define the left and right nodes of the binary tree. If $\{\vec{\xi}_j\}_L$ or $\{\vec{\xi}_j\}_R$ is empty, the branch of the empty set will be eliminated using item (a).

(i) If $\ell=\ell_{\min}$, $i=d$ and $\{\vec{\xi}_j\}_L \neq \varnothing$ or $\{\vec{\xi}_j\}_R \neq \varnothing$, $\{\vec{\xi}_j\}_L$ and $\{\vec{\xi}_j\}_R$ will be joined in a list with terminal nodes of the binary tree, and the function goes back to the caller. This list of terminal nodes is given by $\pi^{-1}(B_s)$. Each element of $\pi^{-1}(B_s)$ is a terminal node and also a table with the positions of each element of the sets $\{\vec{\xi}_j\}_L$ and $\{\vec{\xi}_j\}_R$, $\pi^{-1}(B_s)=\{\vec{\xi}_j\}_L$ and $\pi^{-1}(B_{s+1})=\{\vec{\xi}_j\}_R$. On the other hand, the elements of $\{\vec{\xi}_j\}_L$ and $\{\vec{\xi}_j\}_R$ will point the box B_s through the index $\pi(\vec{\xi}_j)$: $\pi(\vec{\xi}_j)=B_s$ with $\vec{\xi}_j \in \{\vec{\xi}_j\}_L$ and $\pi(\vec{\xi}_j)=B_{s+1}$, where $\vec{\xi}_j \in \{\vec{\xi}_j\}_R$. Finally, if $\{\vec{\xi}_j\}_L$ and $\{\vec{\xi}_j\}_R$ are not terminal nodes, the input variables of axes_divide for each node or branch will be defined as follows at (ii) and (iii), and we will split each set $\{\vec{\xi}_j\}_L$ and $\{\vec{\xi}_j\}_R$ again calling axes_divide.

(ii) Define the left branch. This branch is associated with the left box. The obvious choice for the center of the left box p_{iL} along the axis i is the first quarter of the main box, i.e., $p_{iL}=p_i-\ell/4$, since a box centered here always covers the left side of the attractor. However, sometimes there is free space in the phase space. The free space is verified taking the following distances:

$$d^i=p_i-v_{iL}, \quad d^{ii}=v_{iL}-a_i-\ell/2, \tag{A2}$$

where v_{iL} is the maximum value of the left side of the attractor along axis i. When d^i is smaller than d^{ii}, we can move p_{iL} to $v_{iL}-\ell/4$ dispersing the left box. Now the center of the left box is

$$p_{iL}=\begin{cases} v_{iL}-\ell/4, & d^i \leqslant d^{ii}, \\ p_i-\ell/4, & d^i > d^{ii}. \end{cases} \tag{A3}$$

The main advantage of this movement is the eventual elimination of extra boxes in posterior applications of the procedure axes_divide, since the left side of the left box might be empty. When the left branch is created, the phase space is also divided in two, so the new limit of the left side of the phase space is

$$\vec{a}_L=\vec{a}, \quad \vec{b}_L=\begin{cases} b_k, & k \neq i \\ p_k, & k=i. \end{cases} \tag{A4}$$

After that, we call axes_divide using $\{\vec{\xi}_j\}_L$, $\vec{p}_L$, $\vec{a}_L$, $\vec{b}_L$, ℓ, and i as input variables.

(iii) The procedure for the right branch is the same as the left. So,

$$d^{iii}=u_{iR}-p_i-1, \quad d^{iv}=b_i-u_{iR}-\ell/2, \tag{A5}$$

$$p_{iR}=\begin{cases} u_{iR}+\ell/4-1, & d^{iii} \leqslant d^{iv} \\ p_i+\ell/4, & d^{iii} > d^{iv}, \end{cases} \tag{A6}$$

$$\vec{b}_R=\vec{b}, \quad \vec{a}_R=\begin{cases} a_k, & k \neq i \\ p_k+1, & k=i, \end{cases} \tag{A7}$$

where u_{iR} is the minimum value of the right side of the attractor along axis i. We call axes_divide again using $\{\vec{\xi}_j\}_R$, $\vec{p}_R$, $\vec{a}_R$, $\vec{b}_R$, ℓ, and i as input variables. The term -1 at p_{iR}

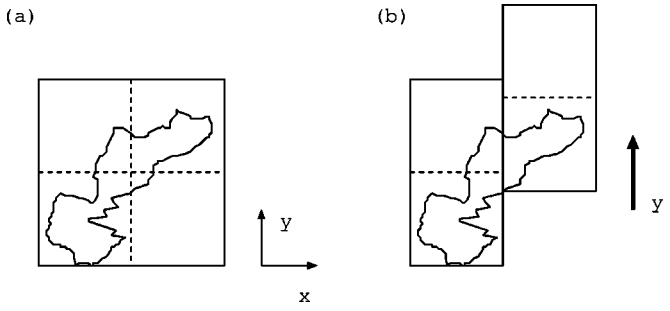

FIG. 7. (a) A two-dimensional attractor covered by a single box. Dashed lines indicate the square division in the box with size $\ell/2$. (b) The movement of the box along the y axis. (c) The elimination of the unnecessary box.

in Eqs. (A5) and (A6), as well as $+1$ at $\vec{a}_R$ in Eq. (A7) is related with the representation of variables as integer numbers.

axes_divide also performs a second movement in the perpendicular semiplane of the Cartesian axis, see Fig. 7. When the attractor is divided at p_i along the axis i, sometimes the best choice for the division of the next axis is not at p_{i+1}. So, we might reduce a bit more the number of boxes N using the following procedure.

(a) If the lengths of the subsets $\{\vec{\xi}_j\}_L$ and $\{\vec{\xi}_j\}_R$ are larger than $\ell/2$ in the axis k, $i<k\leqslant d$, i.e., if the length of the subsets is too large for the second movement, nothing will happen and the procedure will be finished.

(b) In case $b_k-v_k\leqslant\ell/2$, we will define $p_k=v_k-\ell/2$. In another case $p_k=u_k+\ell/2-1$. We check through the conditional expression $b_k-v_k\leqslant\ell/2$ if there is free space for the box movement at the top of the attractor. In the affirmative case, the box is moved to this direction. On the other hand, the box moves to the bottom of the attractor. Using this movement an empty side opens into the box. This empty side will be eliminated with the future application of axes_divide.

There is one last problem with the sides ℓ of the boxes, which are always given by 2^k, $k\leqslant n_b$. axes_divide always splits boxes into two parts with the same length, and ℓ must have the same computer representation of the phase space. So, ℓ is an integer 2^k because the division of ℓ must result in another integer. On the other hand, Lyapunov exponents are very sensitive to the definition of neighborhood [40,42], limiting the proposed algorithm, since we need boxes with sizes different from 2^k. We avoid the problem by changing proportionally the data instead of the box size. So, we renormalize the time series by writing $\{x_j\}_{\text{new}}=\{x_j/2^{\log_2 \ell_{\min}[\text{mod}1]}\}$. Now, the new $\ell_{\min}$ values $2^{\text{integer}(\log_2 \ell_{\min})}$ for axes_divide.

[1] H.D.I. Abarbanel, *Analysis of Observed Chaotic Data* (Springer-Verlag, New York, 1996).
[2] H.D.I. Abarbanel, R. Brown, J.J. Sidorowich, and L.Sh. Tsimring, Rev. Mod. Phys. **65**, 1331 (1993).
[3] J.M. Greene and J.-S. Kim, Physica D **24**, 213 (1987).
[4] I. Goldhirsch, P.-L. Sulem, and S.A. Orszag, Physica D **27**, 311 (1987).
[5] I. Shimada and T. Nagashima, Prog. Theor. Phys. **61**, 1605 (1979).
[6] G. Benettin, L. Galgani, A. Giorgilli, and J.-M. Strelcyn, Meccanica **15**, 9 (1980).
[7] A. Wolf, J.B. Swift, H.L. Swinney and J.A. Vastano, Physica D **16**, 285 (1985).
[8] M. Sano and Y. Sawada, Phys. Rev. Lett. **55**, 1082 (1985).
[9] J.-P. Eckmann, S.O. Kamphorst, D. Ruelle, and S. Ciliberto, Phys. Rev. A **34**, 4971 (1986).
[10] R. Brown, P. Bryant, and H.D.I. Abarbanel, Phys. Rev. A **43**, 2787 (1991).
[11] K. Geist, U. Parlitz, and W Lauterborn, Prog. Theor. Phys. **83**, 875 (1990).
[12] A. Karantonis and M. Pagitsas, Phys. Rev. E **53**, 5428 (1996).
[13] V.I. Oseledec, Tr. Mosk. Mat. Obsc. Moscow Math. Soc. **19**, 179 (1968).
[14] M.C. Kim and C.S. Hsu, J. Stat. Phys. **45**, 49 (1986).
[15] A Boyarsky, J. Stat. Phys. **50**, 213 (1988).
[16] G. Froyland, K. Judd, and A.I. Mees, Phys. Rev. E **51**, 2844 (1995).
[17] A.I. Mees, Int. J. Bifurcation Chaos Appl. Sci. Eng. **1**, 777 (1991).
[18] J.P. Crutchfield and B.S. McNamara, Complex Syst. **1**, 417 (1987).
[19] F. Takens, in *Dynamical Systems and Turbulence, Warwick, 1980*, edited by D. Rand and L.S. Yang, Lecture Notes in Mathematics Vol. 898 (Springer, Berlin, 1981), p. 366.
[20] D.E. Knuth, *Fundamental Algorithms—The Art of Computer Programming* (Addison-Wesley, Reading, MA, 1968), Vol. 1.
[21] N.N. Oiwa and N. Fiedler-Ferrara, Physica D **124**, 210 (1998).
[22] J.-P. Eckmann and D. Ruelle, Rev. Mod. Phys. **57**, 617 (1985).
[23] J.H. Wilkinson, *The Algebraic Eigenvalue Problem* (Clarendon Press, London, 1965).
[24] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes—The Art of Scientific Computing* (Cambridge University Press, Cambridge, 1989).
[25] B. Carnahan, H.A. Luther, and J.O. Wilkes, *Applied Numerical Methods* (Wiley & Sons, New York, 1969).
[26] J.H. Friedman, J.L. Bentley, and R.A. Finkel, ACM Trans. Math. Softw. **3**, 209 (1977).
[27] T. Schreiber, Int. J. Bifurcation Chaos Appl. Sci. Eng. **5**, 349 (1995).
[28] Y. Fisher, *Fractal Image Compression—Theory and Application* (Springer-Verlag, New York, 1995).
[29] A.M. Albano, A. Passamante, and M.E. Farrell, Physica D **54**, 85 (1991).

[30] A.M. Fraser and H.L. Swinney, Phys. Rev. A **33**, 1134 (1986).
[31] Th. Buzug, T. Reimers, and G. Pfister, Europhys. Lett. **13**, 605 (1990).
[32] W. Liebert, K. Pawelzik, and H.G. Schuster, Europhys. Lett. **14**, 521 (1991).
[33] M.B. Kennel, R. Brown, and H.D.I. Abarbanel, Phys. Rev. A **45**, 3403 (1992).
[34] J.-P. Eckmann and D. Ruelle, Physica D **56**, 185 (1992).
[35] E.N. Lorenz, J. Atmos. Sci. **20**, 130 (1963).
[36] A.J. Lichenberg and M.A. Liberman, *Regular and Stochastic Motion* (Springer-Verlag, New York, 1983).
[37] M. Hénon, Commun. Math. Phys. **50**, 69 (1976).
[38] D.E. Rossler, Phys. Lett. **57A**, 397 (1976).
[39] M.C. Mackey and L. Glass, Science **197**, 287 (1977).
[40] N.N. Oiwa, Master dissertation, Universidade de São Paulo, 1994.
[41] W.P. de Sá, Master's dissertation, Universidade de São Paulo, 1987; D.F. da Cruz, Master's dissertation, Universidade de São Paulo, 1987.
[42] N.N. Oiwa and N. Fiedler-Ferrara (unpublished).
[43] C.P.C. Prado and N. Fiedler-Ferrara, Plasma Phys. Controlled Fusion **33**, 493 (1990).