PHYSICAL REVIEW E **74**, 026702 (2006)

Improving the global fitting method on nonlinear time series analysis

L. M. C. R. Barbosa
Universidade Federal Fluminense, Instituto de Física Campus da Praia Vermelha, Gragoatá, Niterói, RJ CEP: 24210-310, Brazil

L. G. S. Duarte, C. A. Linhares, and L. A. C. P. da Mota*
Universidade do Estado do Rio de Janeiro, Instituto de Física, Departamento de Física Teórica, 20559-900 Rio de Janeiro, Brazil

(Received 20 December 2005; published 4 August 2006)

We are concerned with improving the forecast capabilities of the global approach to time series. We assume that the normal techniques of global mapping are applied, the noise reduction is performed, etc. Then, using the mathematical foundations behind such approaches, we propose a method that, without a great computational cost, greatly increases the accuracy of the corresponding forecasting.

DOI: 10.1103/PhysRevE.74.026702 PACS number(s): 05.10.−a, 05.45.Tp

I. INTRODUCTION

For any observed system, physical or otherwise, one generally wishes to make predictions on its future evolution. Sometimes, very little is known about the system. Possibly, the dynamics behind the phenomenon being studied is unknown, and one is given just a time series of one (or a few) of its parameters. Therefore, performing a time-series analysis is the best one can do in order to learn the properties of the phenomenon. Its relevance may be gauged by the existence of extensive studies in a great diversity of branches of knowledge, in physics as well as in economics and the stock exchange, meteorology, oceanography, medicine, etc.

A time series is normally taken as a set of numbers that are the possible outcome of measurements of a given quantity, taken at regular intervals. In reality, however, the assumption that the time series reflects in some way the underlying dynamics of the systems is worsened by the fact that the measured data usually contain irregularities. These may be due to a random external influence on a linear system, a noise (induced possibly by the measuring apparatus or other sources of contamination) which gets mixed with the desired information, thereby hiding it. But it may well be that they appear as a manifestation of low-dimensional deterministic chaos resulting from an intrinsic nonlinear dynamics governing the quantity under study (over which a random noise may also be superimposed), with the characteristic sensitivity to initial conditions.

If the time series is the only source of information on the system, prediction of the future values of the series requires a modeling of the system's (perhaps nonlinear) dynamical law through a set of differential equations or through discrete maps. However, it is even possible that we do not know whether the measured quantity is the only relevant degree of freedom (frequently it is not) of the dynamical problem or how many of them there are.

Both noise-contaminated linear and nonlinear systems have nevertheless been studied with success, employing statistical tools and chaos-theory concepts, together with time-series analysis [1,2]. Given a time series, one should ask first whether it represents a causal process or whether it is stochastic. Tools have been developed to decide upon this fundamental question (the most common ones are spectral analysis, Lyapunov characteristic exponents, and correlation functions; see [3,4]). In the case of a series originating from a low-dimensionality chaotic dynamics, traditional linear methods of analysis are not adequate, but an analysis apparatus was devised for applications to such nonlinear systems [3,4] and we will not be concerned with stochastic processes in this paper.

Methods for dealing with nonlinear time series fall mainly into two categories: local or global methods. Local methods are based on the assumption that, while in the long run nearby trajectories on the phase space diverge considerably, they stay within the same neighborhood for a while. One may conjecture that to predict the next step in a time series, a good indication should come from the previous visits the system had made to the phase space neighborhood containing the "last point" of the series. An average of the behavior of the system for neighboring points, with a minimization of the distance in the phase space between them, gives good results for the next-step forecasting.

Global methods, on the other hand, postulate a functional form for the dynamics to be valid for any time. Usually one considers polynomials of a suitable degree, and one should devise a convenient way to estimate its coefficients. In this paper, we are going to concentrate on the global approach and, actually, we will start from the global mapping itself; i.e., we are not going to be concerned with how the global mapping was generated (there are many standard approaches to do it) and we will not deal with noise reduction either (such considerations are important when determining the mappings, etc.). We will focus on a method to, from any standard mapping one might have, improve the forecasting using it, without having to pay a very high computational price.

Nonlinear analysis of time series relies not on the original maps of the dynamic system, but on its *time-delay reconstruction*. All discussions on the nonlinear treatment of time series make use of this reconstruction scheme. There are already classical references dealing with the subject [1,3–8]. This method allows one to reconstruct the phase space of the system with reasonable accuracy, using the information contained in the series only.

*Electronic address: lduarte@dft.if.uerj.br, linhares@dft.if.uerj.br, damota@dft.if.uerj.br

Lorenz [9] showed that dynamic systems of low dimensionality could present strange attractors on their phase spaces. Takens [10] proposed a method to reconstruct such phase spaces from the knowledge of a time series obtained from the system. He demonstrated that the original attractor and the reconstructed one are characterized by the same asymptotic properties and topological characteristics [11]. So if we want to analyze the properties of the corresponding attractor of the system, we have to reconstruct it.

In [10], Takens used a method to reconstruct the phase space. Vectors $\vec{\xi}_i$ (with dimension m) are reconstructed from the time series x_i where $x_i=x(t_i)$, $i=1,\ldots,N$, as follows:

$$\vec{\xi}_i=\{x(t_i),x(t_i+p),\ldots,x(t_i+(m-1)p)\}, \tag{1}$$

where m is the embedding dimension and p is the time lag (for definitions, see [12]). Based on the trajectories of the reconstructed attractor, we can study various topological invariants of the system such as the Lyapunov exponents, the generalized entropies [11], etc. We can also extract the underlying dynamics via a global modeling of the system. For example, one can try to obtain a low-order Taylor series expansion for the system, thus obtaining a global mapping representing the system. We can use this mapping to perform a forecast of entries we ignore—i.e., in the future.[1]

II. AN ALGORITHM TO IMPROVE THE GLOBAL FORECASTING

A. Stating the problem

Suppose that the system can be modeled by a set of differential equations of low dimensionality. What we would like to obtain is some kind of global map that, given any point of the phase space, could calculate a subsequent point of the trajectory. If we know the set of differential equations (SDE) that models the system, we could find a solution (starting from an initial condition) by making a numerical integration through some map obtained from the SDE (probably a Runge-Kutta map, a Taylor series one, or an expansion in some function basis). For practical purposes (computers cannot work with infinity) a truncation must occur at some order of the series expansion. However, if the truncation order is low, we can run away from the real solution in a few time steps (even if each time step is very small). For chaotic systems a Runge-Kutta expansion of degree less than 4 is not used (in general). This implies that the map generated presents polynomials of high degree. Let us exemplify using one of the simplest chaotic system that exists, the Lorenz system:

$$\dot{x}_1=\sigma(x_2-x_1),$$

$$\dot{x}_2=-x_2-x_1x_3+Rx_1,$$

$$\dot{x}_3=x_1x_2-bx_3, \tag{2}$$

where σ, R, and b are parameters and the system presents chaotic behavior for $R>24,74$.

Why *one of the simplest*? Notice that this system possesses the minimum number of autonomous[2] differential equations permitting chaos: 3.[3] Besides that, chaos is a phenomenon that only takes place in nonlinear systems, and the smallest piece of nonlinearity that we can add to a linear system in order to turn it nonlinear is a quadratic term.

Observe that the Lorenz system presents only two nonlinear quadratic terms. Even in this simple case, as we will show, a Taylor series expansion of fourth order will lead to a map of fifth degree in three variables.

Consider the following initial condition: $x_1(0)=x_{10}$, $x_2(0)=x_{20}$, $x_3(0)=x_{30}$. We can expand the corresponding solution as

$$x_i=\phi_i(t)=\phi_i(0)+\frac{d\phi_i}{dt}(0)t+\frac{d^2\phi_i}{dt^2}(0)\frac{t^2}{2!}+\cdots. \tag{3}$$

Since the system is defined by the equation $\frac{dx_i}{dt}=f_i(\vec{x})$, we have that $\dot{x}_i=\dot{\phi}_i=f_i$,[4] implying that

$$\frac{d}{dt}\left(\frac{d\phi_i}{dt}\right)=\frac{df_i}{dt}=\sum_{j=1}^{3}\frac{\partial f_i}{\partial x_j}\frac{dx_j}{dt}=\sum_{j=1}^{3}\frac{\partial f_i}{\partial x_j}f_j. \tag{4}$$

We can notice that, for the case of the Lorenz system, this process will increase the degree of the polynomials forming the mapping by 1 for each order.[5] So the mapping corresponding to the fourth-order Taylor expansion is, at maximum, formed by fifth-degree terms. A polynomial mapping of fifth degree implies a total of 168 coefficients. Remember that, as mentioned, this is for one of the simplest chaotic dynamic system cases [i.e., three-dimensional and only two nonlinear (quadratic) terms].

It is important to notice that, in time-series analysis, we do not have the dynamic system to begin with. We, of course, will consider that there is such a system behind the series and we will look for determining it. With the explanations above, we hope to have made it clear that, even if the underlying system is as simple as the Lorenz's one, we will already have to face a great computational task (if one wants to use fourth order expansions—generally the minimum accuracy necessary for practical purposes) of determining the 168 coefficients. With more detail, using the Lorenz system as a model for the global fitting scheme, let us suppose that we have a time series produced from this system (for instance, take one of the coordinates of the system). After the usual phase-space reconstruction [4], say we want to have a fourth-order mapping (for the reconstructed system) with the

[1]One can also do forecasting in a local version via analyzing the behavior of close vectors (to the one just before the one to be predicted) in order to estimate the next (unknown) entry (see [1]).

[2]The time does not appear explicitly.

[3]In two dimensions we cannot have chaos because the trajectories cannot cross.

[4]Where $\dot{u}$ represents $\frac{du}{dt}$.

[5]Since the highest degree present in the functions f, g, and h is quadratic, the derivatives [present in Eq. (4)] are, at maximum, first-degree polynomials.

same accuracy that could be found in the fourth-order Taylor expansion for the Lorenz system. We would have to employ some minimization technique to determine 168 coefficients. In practice, this is a very high number, making the whole procedure computationally expensive.

So we are left with the hard choice of either paying the computational price mentioned above and be very patient or trying to decrease the degree of the mapping. Of course, there is no such thing as a "free lunch." The price for the latter choice would be that the accuracy would decrease (the corresponding Taylor expansion would be of lower order).

Therefore, despite the fact that the global approach has many attractive features, such as the fact that, once it is determined it is applicable to the whole series,[6] one sees that the effective use of it can be difficult to achieve in practice. So there is a clear demand for procedures that can, without increasing the degree of the global mapping, enhance the accuracy of such mappings. In the next subsection, before introducing one such attempt, we will talk about mappings.

B. Regarding mappings

In order to clarify the central idea of our proposed algorithm, let us make some comments and present some results concerning mappings representing the solutions for SDE's.

Consider the transformation group in n variables:

$$x_i^* = F_i(\vec{x},t), \tag{5}$$

where t is the group parameter. From Lie's theory [13–15], we know that this group is the solution to a SDE defined by

$$x_i = f_i(\vec{x}), \tag{6}$$

where $f_i(\vec{x}) \equiv \frac{\partial F_i}{\partial t}|_{t=0}$ and $\dot{x}_i \equiv \frac{dx_i}{dt}$. Therefore, the transformation group (5) [i.e., the solution to the dynamic system (6)] can be obtained from the group generator defined as the operator $X \equiv \Sigma_{i=1}^n f_i \frac{\partial}{\partial x_i}$, as follows:

$$x_i^* = F_i(\vec{x},t) = x_i + tX[x_i] + \frac{t^2}{2!}X^2[x_i] + \cdots = \sum_{k=0}^{\infty} \frac{t^k}{k!}X^k[x_i]. \tag{7}$$

In this way, starting from a generic point P_0, with corresponding coordinates $\vec{x}_{(P_0)}$, by choosing a time interval δt, the transformation group (5) generates a mapping M that takes a point on some given solution to the system and takes it to another such point that corresponds to a group parameter increased by δt:

$$x_{i(P+1)} = F_i(\vec{x}_{(P)}, \delta t) = \sum_{k=0}^{\infty} \frac{\delta t^k}{k!}X^k[x_{i(P)}]. \tag{8}$$

In practice, the process of numerically solving the SDE can be summarized by choosing a small time interval ($\delta t \ll 1$) and truncating the series (8) at some order N, thus obtaining a mapping $\bar{M}$ given by

$$\bar{x}_{i(P+1)} = \bar{F}_i(\vec{x}_{(P)}, \delta t) = \sum_{k=0}^{N} \frac{\delta t^k}{k!}X^k[x_{i(P)}], \tag{9}$$

where $\bar{x}_{i(P+1)}$ approaches $x_{i(P+1)}$ when $\delta t \to 0$. Defining the functions $\delta^k \varepsilon_i$ as

$$(\varepsilon_i(\vec{x}_{(P)}) = \delta^0 \varepsilon_i(\vec{x}_{(P)})) \equiv x_{i(P+1)} - \bar{x}_{i(P+1)} = \sum_{k=N+1}^{\infty} \frac{t^k}{k!}X^k[x_{i(P)}],$$

$$(\delta\varepsilon_i(\vec{x}_{(P)}) = \delta^1 \varepsilon_i(\vec{x}_{(P)})) \equiv \varepsilon_i(\vec{x}_{(P+1)}) - \varepsilon_i(\vec{x}_{(P)}),$$

$$\delta^k \varepsilon_i(\vec{x}_{(P)}) \equiv \delta^{k-1}\varepsilon_i(\vec{x}_{(P+1)}) - \delta^{k-1}\varepsilon_i(\vec{x}_{(P)}), \tag{10}$$

where ($k=2,\ldots$), one can notice that

$$\delta\varepsilon_i(\vec{x}_{(P)}) = \sum_{j=1}^{n} \frac{\partial \varepsilon_i(\vec{x}_{(P)})}{\partial x_j}\delta x_i + O(\delta x_i^2) \tag{11}$$

and, generally,

$$\delta^{k+1}\varepsilon_i(\vec{x}_{(P)}) = \delta\delta^k\varepsilon_i(\vec{x}_{(P)}) = \sum_{j=1}^{n} \frac{\partial \delta^k \varepsilon_i(\vec{x}_{(P)})}{\partial x_j}\delta x_i + O(\delta x_i^2). \tag{12}$$

Since $\delta t \to 0$ implies that $\delta x_i \to 0$, we can, using Eq. (12), enunciate the following result:

$$\lim_{\delta t \to 0} \frac{\delta^{k+1}\varepsilon}{\delta^k \varepsilon} = 0, \tag{13}$$

where k is a positive integer.

In the next subsection, based on this important result, we will present an algorithm that enhances the predictive power of global mappings for time series.

C. Mathematical basis for the algorithm

Based on the above result (13), we have produced an algorithm that allows for improving the forecasting for the global fitting of a time series.

As mentioned, we will suppose that the given time series originates from phenomena that can be described by a low-dimensional dynamic system (S_0). After phase-space reconstruction [10], we have a set of vectors defining a set of points along a single trajectory of the reconstructed systems (S_r).[7] As usual, what we would like to determine is a global mapping M that would (with infinite precision) represent the solutions of the system S_r. But, of course, in practice, what we can do is to produce a global mapping $\bar{M}$ through a procedure involving a minimization process.[8] If the mapping $\bar{M}$ produces good forecasting for the series, that means that the coefficients present on $\bar{M}$ are close to the analogous ones

[6]In the case of local mappings, we have to determine a mapping for each entry of the series.

[7]Takens [10] has demonstrated that the systems S_0 and S_r are topologically equivalent.

[8]In layman terms, what is done is to adjust the coefficients of the polynomial mapping (of a certain degree) to better reproduce the phase-space points.

present on the mapping M which can be represented by the infinite series (8) (and, ideally, it would describe S_r with absolute precision). In that situation, we would be in a similar position to the one presented in the last subsection (where we had just a truncated series because we knew the underlying SDE and could determine the Taylor expansion). Why similar? In the "real" case we are dealing with now, we only have the series and have to determine the mapping through a finite process and, therefore, the coefficients would not be exactly the same as in the truncated expansion of S_r. So defining functions ε_i and $\delta^k\varepsilon_i$ analogously to how we did in the last subsection, we would expect that Eq. (13) would be valid. Actually, in the real world, the inequality

$$\delta^{k+1}\varepsilon \ll \delta^k\varepsilon \tag{14}$$

is not valid for any positive integer k. The point is that, in actual calculations, δt would be a finite value (not infinitesimal) Δt. So at some integer value K, the inequality (14) would become

$$\Delta^{K+1}\varepsilon \approx \Delta^K\varepsilon. \tag{15}$$

The above reasoning allows us to build an easily applicable algorithm: Consider that we want to forecast the coordinate x_i (where i can take any value from 1 to the dimensionality of the reconstructed system) of a point $P+1$ that immediately follows a given point P. In order to produce the mapping $\bar{M}$, we use a certain number $a+1$ of points that precede the point $P+1$ (the points $P, P-1, P-2, \ldots, P-a$). Using this mapping, we can forecast the x_i coordinates for these $a+1$ points. Let us call these $a+1$ values $\bar{x}_i$. From these, we can define the functions $\Delta^k\varepsilon_i$ [analogously to the functions (10) in Sec. II B].

$$\begin{aligned}
\Delta^0\varepsilon_i(\vec{x}_{(J)}) &\equiv x_{i(J)} - \bar{x}_{i(J)},\\
\Delta^1\varepsilon_i(\vec{x}_{(J)}) &\equiv \Delta^0\varepsilon_i(\vec{x}_{(J)}) - \Delta^0\varepsilon_i(\vec{x}_{(J-1)}),\\
&\vdots\ \vdots\\
\Delta^k\varepsilon_i(\vec{x}_{(J)}) &\equiv \Delta^{k-1}\varepsilon_i(\vec{x}_{(J)}) - \Delta^{k-1}\varepsilon_i(\vec{x}_{(J-1)}),\\
&\vdots\ \vdots
\end{aligned} \tag{16}$$

where $(k=0,\ldots)$ and $(J=P-a+k,\ldots,P)$. Using these definitions, we can determine the values for k where we have $\Delta^{k+1}\varepsilon \approx \Delta^k\varepsilon$,[9] and, using this knowledge, we will see that we can improve the forecasting generated by the mapping $\bar{M}$. Let us clarify what we mean: if we want to forecast the value for the coordinate x_i of the point $P+1$, we may use the global mapping $\bar{M}$ that would produce the forecast $\bar{x}_{i(P+1)}$. We know that $x_{i(P+1)} - \bar{x}_{i(P+1)} = \Delta^0\varepsilon_i(\vec{x}_{(P+1)})$ and, therefore,

$$x_{i(P+1)} = \bar{x}_{i(P+1)} + \Delta^0\varepsilon_i(\vec{x}_{(P+1)}). \tag{17}$$

Notice that we do not know the value for $\Delta^0\varepsilon_i(\vec{x}_{(P+1)})$. But we know that $\Delta^1\varepsilon_i(\vec{x}_{(P+1)}) = \Delta^0\varepsilon_i(\vec{x}_{(P+1)}) - \Delta^0\varepsilon_i(\vec{x}_{(P)})$, implying that

$$\Delta^0\varepsilon_i(\vec{x}_{(P+1)}) = \Delta^0\varepsilon_i(\vec{x}_{(P)}) + \Delta^1\varepsilon_i(\vec{x}_{(P+1)}). \tag{18}$$

Let us examine this: we know the value for $\Delta^0\varepsilon_i(\vec{x}_{(P)})$ (i.e., $x_{i(P)} - \bar{x}_{i(P)}$) but we do not know $\Delta^1\varepsilon_i(\vec{x}_{(P+1)})$. However, if (P) and $(P+1)$ are sufficiently close (such that $\Delta^1\varepsilon_i \ll \Delta^0\varepsilon_i$), we can expect that we will gain information when substituting Eq. (18) into Eq. (17), obtaining

$$x_{i(P+1)} = \bar{x}_{i(P+1)} + \Delta^0\varepsilon_i(\vec{x}_{(P)}) + \Delta^1\varepsilon_i(\vec{x}_{(P+1)}). \tag{19}$$

Why do we gain information? If we compare Eq. (17) to Eq. (19), we can observe that the unknown term in Eq. (17) is $\Delta^0\varepsilon_i(\vec{x}_{(P+1)})$ which is (by hypothesis) much bigger than the unknown term in Eq. (19): $\Delta^1\varepsilon_i(\vec{x}_{(P+1)})$. So the term $\Delta^0\varepsilon_i(\vec{x}_{(P)})$ is a correction to $\bar{x}_{i(P+1)}$. Analogously, we have $\Delta^2\varepsilon_i(\vec{x}_{(P+1)}) = \Delta^1\varepsilon_i(\vec{x}_{(P+1)}) - \Delta^1\varepsilon_i(\vec{x}_{(P)})$, implying that

$$\Delta^1\varepsilon_i(\vec{x}_{(P+1)}) = \Delta^1\varepsilon_i(\vec{x}_{(P)}) + \Delta^2\varepsilon_i(\vec{x}_{(P+1)}). \tag{20}$$

Substituting this into Eq. (19), if (P) and $(P+1)$ are sufficiently close such that $\Delta^2\varepsilon_i \ll \Delta^1\varepsilon_i$, we would have a second-order correction to $\bar{x}_{i(P+1)}$. Actually, when the relation $\Delta^{k+1}\varepsilon_i \ll \Delta^k\varepsilon_i$ applies, we can further correct $\bar{x}_{i(P+1)}$—i.e.,

$$\begin{aligned}
x_{i(P+1)} = \bar{x}_{i(P+1)} &+ \Delta^0\varepsilon_i(\vec{x}_{(P)}) + \Delta^1\varepsilon_i(\vec{x}_{(P)}) + \cdots + \Delta^k\varepsilon_i(\vec{x}_{(P)})\\
&+ \Delta^{k+1}\varepsilon_i(\vec{x}_{(P+1)}).
\end{aligned} \tag{21}$$

Therefore, we can build a simple algorithm to improve the prediction $\bar{x}_{i(P+1)}$, obtained with mapping $\bar{M}$: we determine the integer k for which the approximation starts to fail—i.e., $\Delta^{k+1}\varepsilon_i \approx \Delta^k\varepsilon_i$—then we neglect the term $\Delta^{k+1}\varepsilon_i(\vec{x}_{(P+1)})$ and end up with

$$x_{i(P+1)} \cong \bar{x}_{i(P+1)} + \Delta^0\varepsilon_i(\vec{x}_{(P)}) + \Delta^1\varepsilon_i(\vec{x}_{(P)}) + \cdots + \Delta^k\varepsilon_i(\vec{x}_{(P)}). \tag{22}$$

The remaining question is how to define $\Delta^{k+1}\varepsilon_i \approx \Delta^k\varepsilon_i$. Let us elaborate on the analysis just made above. We are interested in using an approximation, a kind of Taylor series expansion, when trying to forecast the time series. What one might expect from such a situation? In a perfect world, the terms in the series would, gradually, become smaller in an infinite fashion. Of course, as already mentioned above, we are dealing with a real series, where each entry is not infinitesimally apart the previous one and is, actually, finitely separated. How "finitely separated" depends on the particular series under study and, being more rigorous, on the particular section of the series we are considering. This translates to the fact that, if one considers the absolute values of the differences $\Delta^k\varepsilon_i$, they will decrease with increasing values for k until this value reaches the magnitude defined by the "non-infinitesimal" character of the time series we have just emphasized, where this character will then make the values for the differences oscillate (for a while) around this magnitude

[9]There is a finite range for the values for k in which that happens. After a certain value, the $\Delta^{k-1}\varepsilon_i$ start to diverge.

(since this magnitude would dominate over the initial tendency of the differences to decrease). With the increasing values for k, this initial tendency of the differences to decrease will cease as our approximation (Taylor like) stars to diverge from the actual value for the series. We will then see the absolute values for the following differences start to increase and rapidly diverge. That clearly, if one thinks in plotting the (absolute) values for the differences, defines a plateau where $\Delta^{k+1}\varepsilon_i \approx \Delta^k \varepsilon_i$ and our above-introduced method will work at its best.

D. Steps of the algorithm

Consider that we have already reconstructed the phase space from the time series under study and that we want to forecast the $P+1$ entry (P is the last known value of the series). This entry corresponds to a coordinate of a reconstructed vector on the phase space (as usual). Using a global mapping $\bar{M}$, obtained via standard k-fold validation procedures [16], we do the following.

(i) Set $n=10$.

(ii) We calculate the absolute value for the functions $\Delta^k \varepsilon_i$ [see Eq. (16)] up to $k=n$ for the point P.

(iii) We check to see if we have already found the plateau; i.e., we look for the value of k for which $|\Delta^k \varepsilon_i| < |\Delta^{k+1}\varepsilon_i|$. Note that this checking can be very easily automatized.

(iv) If the checking returns false, we set $n=n+10$ and return to step (ii). Otherwise we would have found the corrected value for $x_{i(P+1)}$ as

$$x_{i(P+1)} \cong \bar{x}_{i(P+1)} + \Delta^0 \varepsilon_i(\vec{x}_{(P)}) + \Delta^1 \varepsilon_i(\vec{x}_{(P)}) + \cdots + \Delta^k \varepsilon_i(\vec{x}_{(P)}). \tag{23}$$

III. APPLICATIONS

In this section, we are going to present two applications of the above-introduced improved forecast method. We will start by introducing the time series in question and present the reconstruction parameters and the associated global mapping. We then will proceed to the algorithm, following the steps just introduced, and compare the average performance for the "usual" and the improved approaches.

A. Application 1: Lorenz system

1. Time series

This is an "academic" application in the sense that it, certainly, originates from a dynamic system and we actually even know which one. But it is important in order for us to see the ideas of the improved method working in an arena that suits it very nicely.

The time series was generated taking the consecutive values for the x_1 coordinate of the Lorenz system [see Eq. (2)], starting from the initial condition $x_{1_0}=-0.333\,666\,666\,7$, $x_{2_0}=-0.333\,666\,666\,7$, $x_{3_0}=21.999\,666\,666\,7$, using an eighth-order Runge-Kutta numerical integration [17]. The series presents 600 entries (see Fig. 1 for a plotting of this time series).

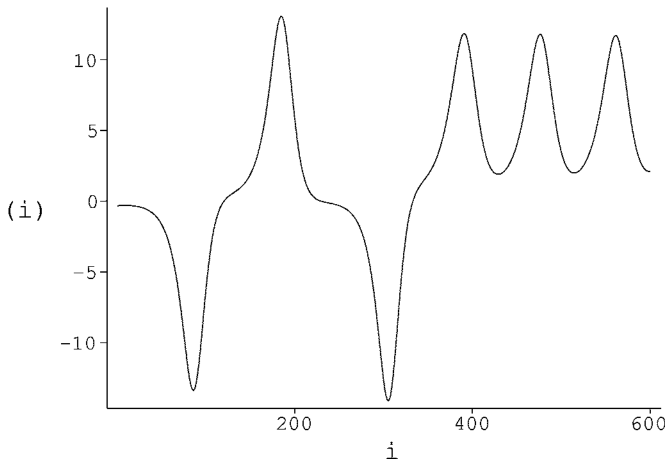

FIG. 1. Lorenz time series. The horizontal axis marks the position of the entry (i) and the vertical on the value for the entry [$X(i)$].

Now, in order to apply the global analysis [1,4] to this time series, we have to reconstruct the phase space. To do that, we need to determine the relevant parameters: namely, the time lag and the embedding dimension (see [12]). For this present case, the reconstruction parameters are (time lag)=6 and (embedding dimension)=3. So in the remainder of this subsection, we will call these three dimensions of the reconstructed phase space for the Lorenz system (x,y,z). In real life, we use the whole time series we know and measure to produce the global mapping and use it to predict future (unknown) entries. Here, in order to evaluate the accuracy of the predictions we obtain using a regular global fitting and our improved one, we are going to use an initial portion of the series to generate the mapping and the other (remaining) portion of the series as our testing ground; i.e., we will apply our mappings to entries in that region and compare them to the actual values to see how the mappings fared. In the present case, the first 140 entries constitute our portion of the series used to build the mapping up. Basically, we use all the vectors reconstructed from these entries and produce a quadratic fitting minimizing the distances from this fitting (when applied to each vector) to the actual values via, for instance, a least-mean-squares procedure. Actually, we also have used an improvement (a very standard one) called a k-fold validation [16]. In layman's language, basically what this k-fold validation does is to average up several mappings. Doing all this, the global mapping we have derived (and to be used on this application henceforth) is

$$\begin{aligned}\bar{M} = {} & 1.317833301x - 0.005266089766x^2 + 0.07580676400xy \\ & - 0.1245478927xz - 0.01839588238y^2 \\ & + 0.06578287850yz - 0.01025562766z^2 \\ & - 0.4700554502y + 0.1415056465z. \end{aligned} \tag{24}$$

2. Inner works of the improved forecast algorithm

Let us now, using two generic points from the series, exemplify the workings of our improved method.

TABLE I. We plot the $|\Delta^k\varepsilon|$, for entry 316, for the Lorenz system time series. IGF is our improved global fitting result corresponding to the particular value of k.

k	$\lvert\Delta^k\varepsilon(\vec{x}_{(P)})\rvert$	IGF	Error
1	0.015127123	−1.042168004	0.06845950907
2	0.000902659	−1.041265345	0.01821336445
3	0.000202785	−1.041468130	0.001257951581
4	0.000032905	−1.041435225	0.001901570346
5	0.000014807	−1.041450032	0.0004798094839
6	0.000018502	−1.041468534	0.001296743462
7	0.000011662	−1.041456872	0.0001769639541
8	0.000009405	−1.041447467	0.0007260995232
9	0.000006933	−1.041454400	0.00006039627084
10	0.000006450	−1.041460850	0.0005589295589
11	0.000005072	−1.041455778	0.00007191861186
12	0.000011998	−1.041443780	0.001080123451
13	0.000020272	−1.041423508	0.003026630927
14	0.000055212	−1.041368296	0.008328060030
15	0.000149033	−1.041219263	0.02263813544
16	0.000346170	−1.040873093	0.05587720869
17	0.000723317	−1.040149776	0.1253297515
18	0.001398781	−1.038750995	0.2596400156
19	0.002499815	−1.036251180	0.4996710232
20	0.004042167	−1.032209013	0.8877979118

TABLE II. We plot the $|\Delta^k\varepsilon|$, for entry 533, for the Lorenz system time series. IGF is our improved global fitting result corresponding to the particular value of k.

k	$\Delta^k\varepsilon(\vec{x}_{(P)})$	IGF	Error
1	0.005833355	7.225283437	0.005138551644
2	0.000348756	7.225632193	0.0003119163708
3	0.000008184	7.225640377	0.0001986532783
4	0.000008912	7.225649289	0.00007531497425
5	0.000003874	7.225653163	0.00002170045565
6	0.000001257	7.225654420	0.000004304108231
7	0.000000331	7.225654751	0.0000002767915261
8	0.000000150	7.225654901	0.000002352727972
9	0.000000251	7.225655152	0.000005826461624
10	0.000000532	7.225655684	0.00001318911622
11	0.000001007	7.225656691	0.00002712556956
12	0.000001712	7.225658403	0.00005081892419
13	0.000002692	7.225661095	0.00008807506360
14	0.000004115	7.225665210	0.0001450249201
15	0.000006683	7.225671893	0.0002375148085
16	0.000012706	7.225684599	0.0004133604651
17	0.000028487	7.225713086	0.0008076084753
18	0.000068998	7.225782084	0.001762511561
19	0.000166009	7.225948093	0.004060005784
20	0.000380304	7.226328397	0.009323252011

Consider the entries $P=316$ and $P=533$, with respective values of $-1.370\,578\,116$ and $6.860\,383\,245$. The values for the entries $P=317$ and $P=534$, the "next" entry for each case considered here, are $-1.041\,455\,029$ and $7.225\,654\,731$. Let us see how the "usual" global fitting fares in these entries. Using the mapping presented in Eq. (24), we get the following forecasting for the entries $P=317$ and $P=534$: $-0.782\,644\,049$ and $7.062\,3742\,64$. These present a "percentage error" (given by $|100\times(\text{value}-\text{forecast})/\text{value}|$) of $24.850\,903\,09$ and $2.259\,732\,482$, respectively. How about the improved method?

In order to apply our method we have to find the plateau by finding the value for k to which $|\Delta^k\varepsilon_i|<|\Delta^{k+1}\varepsilon_i|$. Let us do that for the couple of points chosen above.

a. $P=316$. As "prescribed" above, what we have to do is, by looking at Table I, second column, determine at which value of $k\Delta^k\varepsilon(\vec{x}_{(P)})$ stops decreasing for the first time (and begin the oscillations we have mentioned in Sec. II C). From Table I, we see that happens for $k=5$. Using this in Eq. (23), we find (see Table I) that the "percentage error" for our method is $0.000\,479\,809\,5$.

b. $P=533$. Again, what we have to do is, by looking at Table II, second column, determine to which value of $k\Delta^k\varepsilon(\vec{x}_{(P)})$ stops decreasing for the first time (and begin the oscillations we have mentioned in Sec. II C). From Table II, we see that happens for $k=3$. Using this in Eq. (23), we find (see Table II) that the "percentage error" for our method is $0.000\,198\,653\,3$.

As we mentioned in Sec. II C, we expect the absolute values of $\Delta^k\varepsilon(\vec{x}_{(P)})$ to oscillate when $|\Delta^k\varepsilon_i|\approx|\Delta^{k+1}\varepsilon_i|$. That fact is illustrated, for the entries $P=316$ and $P=533$, respectively, in Figs. 2 and 3.

3. Performance comparison

The reader may ask, why these two entries above? Fair enough, they are not special at all. So in order to confirm the fact that our new approach may be an advantage, let us make a general survey of the entries in the time series. We take 21 entries, equally distributed, in the last part (not used when

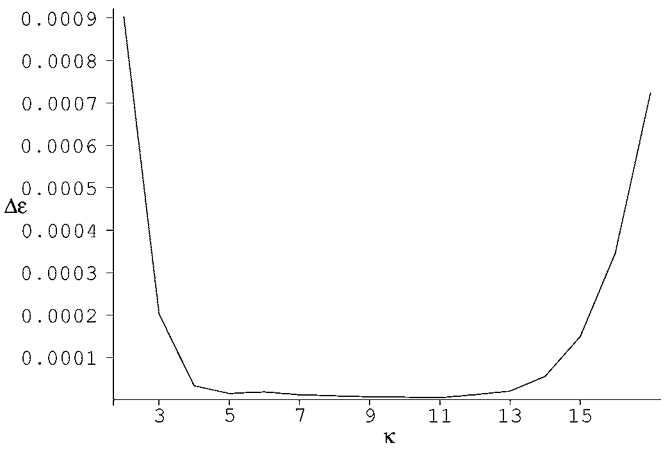

FIG. 2. The plot shows the values of the $|\Delta^k\varepsilon|$ against the number k, for entry 316, for the Lorenz system time series. The value of k that our procedure defines as the beginning of the plateau for this case is $k=5$.

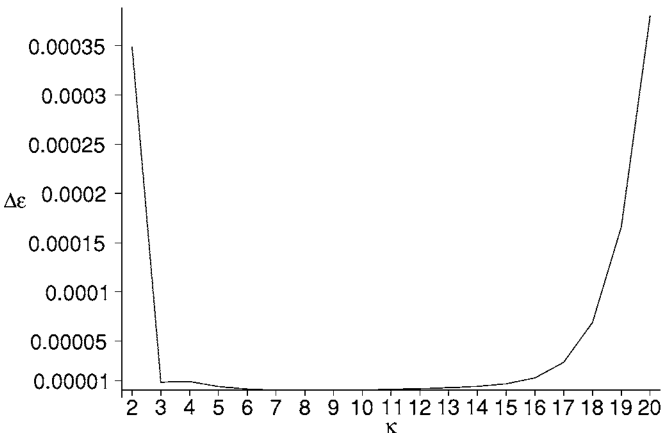

FIG. 3. The plot shows the values of the $|\Delta^k \varepsilon|$ against the value of k, for entry 533, for the Lorenz system time series. The value of k that our procedure defines as the beginning of the plateau for this case is $k=3$.

producing the global mapping) of the time series. The results are presented in Table III.

The idea behind presenting the results for points equally spaced in the entire time series (meaning the entire testing ground defined above) was to provide the information on all the time series; i.e., it is very important (for many series) for the section in which the analysis is carried out. So we have decided to present the results for many points, evenly distributed along every section of the time series. But for completeness, we will present the average percentage error (for the

TABLE III. Comparison between the global fitting and the improved global fitting for the Lorenz time series.

N	GF error	IGF error
300	2.949988321	0.0007879476836
310	9.384955078	0.003895726200
320	602.9300615	0.6329500170
330	4.588145520	0.00003360765471
340	0.6804060144	0.000006172635711
350	1.799833141	0.000002478938512
360	1.908426262	0.00003660198120
370	1.992750955	0.00003095127924
380	5.351353323	0.005676508421
390	15.14897504	0.00006055593702
400	17.40670926	0.03531034693
410	11.79516640	0.00003410572689
420	6.417051601	0.000009213921336
430	3.561881518	0.0000003239205212
440	2.472699706	0.0000008353740914
450	2.070219097	0.000002536966462
460	2.807205469	0.00004274128369
470	9.466251097	0.03219442293
480	17.78540068	0.00003404766212
490	15.54182977	0.00001293826149
500	9.424552928	0.0000008546217942

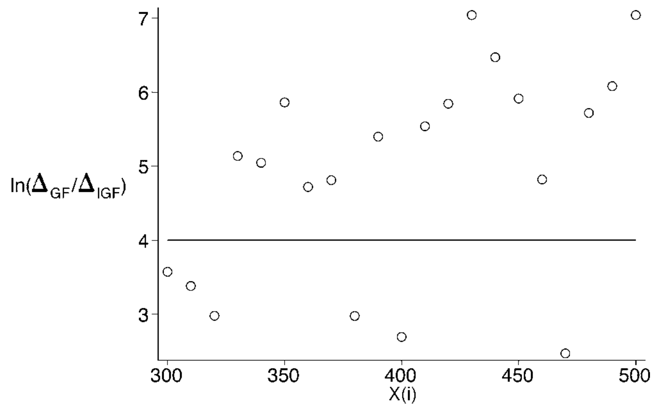

FIG. 4. The plot is for $\ln(\Delta_{GF}/\Delta_{IGF})$ against the position in the time series (i). The line marks the threshold where, above it, Δ_{IGF} starts to be smaller than e^{-4} times Δ_{GF}.

improved global fitting) for the whole testing ground for the time series and for the 21 entries used in Table III. The percentage error for the whole series is 0.160 196 168 3$e-1$ and for the 21 entries on the table is 0.338 584 919 9$e-1$. Both are compatible, showing that the chosen 21 are representative of the totality of the possibilities. The percentage error for the "regular" global fitting is (for the whole series) 10.589 399 30.

As can be seen, our method is a great improvement of accuracy when compared with the "plain" global fitting. To help in this analysis we present Fig. 4 where we plot $\ln(\Delta_{GF}/\Delta_{IGF})$, where Δ_{GF} and Δ_{IGF} are, respectively, the percentage errors in the global fitting and the improved global fitting. As can be seen from the figure, most of the IGF errors are smaller than e^{-4} times the GF errors.

B. Application 2: Heart beat

1. Time series

Let us now deal with a more "real" example, where we deal with data extracted from nature, we do not know the system behind the phenomenon, etc. The following time series was obtained[10] from measurements of the heart beat rate in a person performing many different activities. The series presents 1744 entries (see Fig. 5 for a plotting of this time series).

In order to produce the global mapping for this case, we have proceeded in the same fashion as we did in the Lorenz-system time-series application. So we will not repeat the whole explanation of the procedures involved here. Refer to Sec. IV A above. For this application, the reconstruction parameters are (time lag)=10 and (embedding dimension)=3. So in the remainder of this subsection, we will call these three dimensions of the reconstructed phase space for the heart beat data (x,y,z). The global mapping we have derived

[10]See Ref. [18]. http://ecg.mit.edu/time-series/

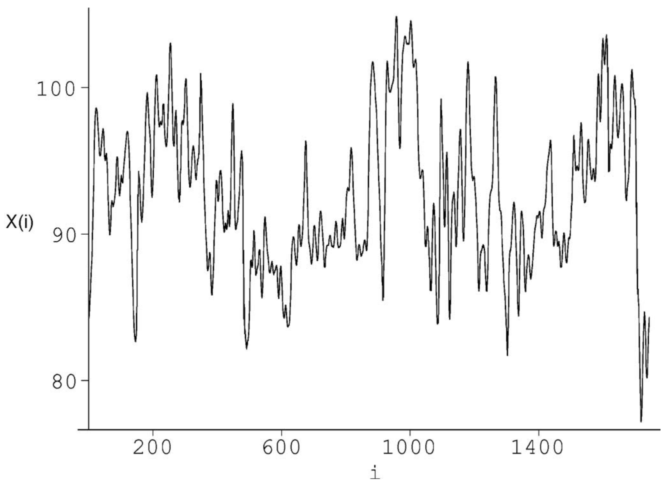

FIG. 5. Heartbeat data. The horizontal axis marks the position of the entry (i) and the vertical on the value for the entry $[X(i)]$.

(and to be used on this application henceforth) is

$$\bar{M} = -1.172534275x - 0.2617292220z^2 + 0.4661889468yz - 0.3426537822y^2 - 0.1107944588xz + 0.3574412776xy - 0.1136908621x^2 + 15.65933419z - 12.98866231y. \qquad (25)$$

2. Inner works of the improved forecast algorithm

Let us now, using two generic points from the series, exemplify the workings of our improved method.

As in the previous application, consider the entries $P=737$ and $P=1016$, with respective values of 89.188 756 24e and 94.250 981 25. The values for the entries $P=738$ and $P=1017$, the "next" entry for each case considered here, are 89.167 431 26 and 94.289 815 63. Let us see how the "usual" global fitting fares in these entries. Using the mapping presented in Eq. (25), we get the following forecasting for the entries $P=738$ and $P=1017$: 90.996 977 and 82.611 004. These present a "percentage error" (defined above) of 2.051 809 404 and 12.386 079 61, respectively. How about the improved method?

TABLE IV. We plot the $|\Delta^k\varepsilon|$, for entry 737, for the time series with the heart beat data.

k	$\Delta^k\varepsilon(\vec{x}_{(P)})$	IGF	Error
0	1.40709476	89.58988224	0.4737727375
1	0.06841402	89.52146822	0.3970473916
2	0.11034922	89.63181744	0.5208024650
3	0.44228546	90.07410290	1.016819288
4	0.56575633	90.63985923	1.651306928
5	0.43932335	91.07918258	2.144001787
6	0.00174856	91.07743402	2.142040802
7	0.99434282	90.08309120	1.026899538
8	3.10231730	86.98077390	2.452304983
9	7.64654504	79.33422886	11.02779598
10	17.67633668	61.65789218	30.85155498

TABLE V. We plot the $|\Delta^k\varepsilon|$, for entry 1016, for the time series with the heart beat data.

k	$\Delta^k\varepsilon(\vec{x}_{(P)})$	IGF	Error
0	9.89683125	92.50783525	1.889896982
1	2.18086125	94.68869650	0.4230370664
2	1.05010575	95.73880225	1.536737144
3	0.01110500	95.72769725	1.524959626
4	0.49314562	95.23455163	1.001949143
5	0.37340972	94.86114191	0.6059257579
6	0.43065248	95.29179439	1.062658521
7	2.97396166	98.26575605	4.216723082
8	10.77153154	109.0372876	15.64057780
9	32.44747527	141.4847629	50.05306984
10	86.66665506	228.1514179	141.9682512

In order to apply our method we have to find the plateau by finding the value for k to which $|\Delta^k\varepsilon_i|<|\Delta^{k+1}\varepsilon_i|$. Let us do that for the couple of points chosen above.

a. $P=737$. As "prescribed" above, what we have to do is, by looking at Table IV, second column, determine to which value of $k\Delta^k\varepsilon(\vec{x}_{(P)})$ stops decreasing for the first time (and begin the oscillations we have mentioned in Sec. II C). From Table IV, we see that happens for $k=1$. Using this in Eq. (23), we find (see Table IV) that the "percentage error" for our method is 0.397 047 391 6.

b. $P=1016$. Again, what we have to do is, by looking at Table V, second column, determine to which value of $k\Delta^k\varepsilon(\vec{x}_{(P)})$ stops decreasing for the first time (and begin the oscillations we have mentioned in Sec. II C). From Table V, we see that happens for $k=3$. Using this in Eq. (23), we find (see Table V) that the "percentage error" for our method is 1.524 959 626.

As in the previous application, we expect the absolute values of $\Delta^k\varepsilon(\vec{x}_{(P)})$ to oscillate when $|\Delta^k\varepsilon_i|\approx|\Delta^{k+1}\varepsilon_i|$. That fact is illustrated, for the entries $P=737$ and $P=1016$, respectively, in Figs. 6 and 7.

3. Performance comparison

Let us make a general survey of the entries of this time series. We take 26 entries, equally distributed, in the last part (not used when producing the global mapping) of the time series. The results are presented in Table VI.

The idea behind presenting the results for points equally spaced in the entire time series (meaning the entire testing ground defined above) is the same one explained in the section regarding the Lorenz system. The percentage error for the whole series is 1.877 994 467 and for the 26 entries in the table is 1.429 515 613. Both are compatible, showing that the chosen 26 are representative of the totality of the possibilities. The percentage error for the "regular" global fitting (for the whole series) is 5.971 546 764.

As can be seen, in the majority of cases, our method is, for this more "realistic" case, also a great improvement of

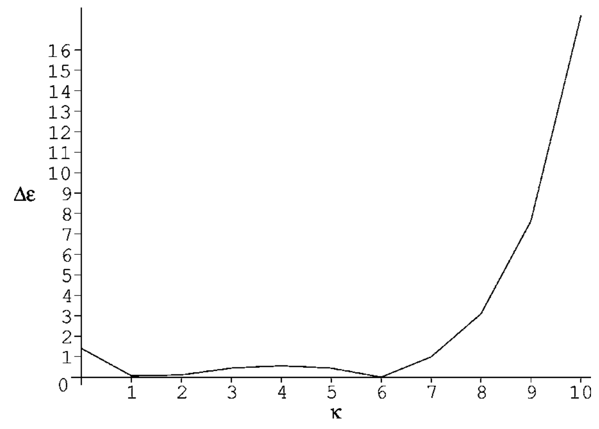

FIG. 6. The plot shows the values of the $|\Delta^k \varepsilon|$ against the value of k, for entry 737, for the heartbeat time series. The value of k that our procedure defines as the beginning of the plateau for this case is $k=1$.

accuracy when compared with the "plain" global fitting. To help in this analysis we present Fig. 8 where we plot $\ln(\Delta_{GF}/\Delta_{IGF})$, where Δ_{GF} and Δ_{IGF} are, respectively the percentage errors in the global fitting and the improved global fitting. As can be seen from the figure, most of the IGF errors are more than five times smaller than the GF errors.

IV. CONCLUSION

There is a huge demand for improving methods that do not cost too high a computational price to achieve desired levels of accuracy in time series analysis.

Here, we have presented one such method. The basic rationale behind it is that we can make use, as explained in Sec. II, of the underlying (assumed) low-dimensionality dynamics to correct our forecast. It is important to mention that, in order to apply the method, one does not have to quantify the hyperbolicity (or the low dimensionality, for that matter) of the time series. The steps of the procedure will take (automatically) care of stopping when this hyperbolicity "spoils" the correcting power of the method. So the algorithm is secure. It is also useful to remember that our efforts here are aimed at avoiding the computational cost of the fitting and minimizing procedures. So our method is not equivalent to fittings, with the same computational cost, in any shape or form.

We have presented two applications of our method: The first one is a (we are going to call it) pure low-dimensional known system, from which we generated a time series. The

TABLE VI. Comparison between the global fitting and the improved global fitting for the time series with heart beat data.

N	GF error	IGF error
500	3.397125034	2.168865202
540	9.089213645	1.889289807
580	5.874155830	0.1858455792
660	1.844997944	1.332964363
700	0.3583492837	0.01355789862
740	2.400959523	0.1672692403
780	0.4937771060	0.1540773590
820	1.249041343	0.1600760146
860	10.26039663	0.8787867282
900	6.962087707	0.8589041386
980	4.565823761	0.2342961164
1020	14.67475919	3.116867623
1060	1.249489572	0.5393005133
1100	1.075608307	2.987879519
1140	0.2169661915	0.08177074130
1180	13.66135448	0.4848516873
1220	2.732512685	0.8664583752
1260	6.861097043	0.6660946812
1340	9.400776847	0.9535935739
1380	1.392042607	0.2352261816
1420	0.8176069275	0.5917808721
1460	2.813583072	0.2695624030
1500	6.384669758	6.651398095

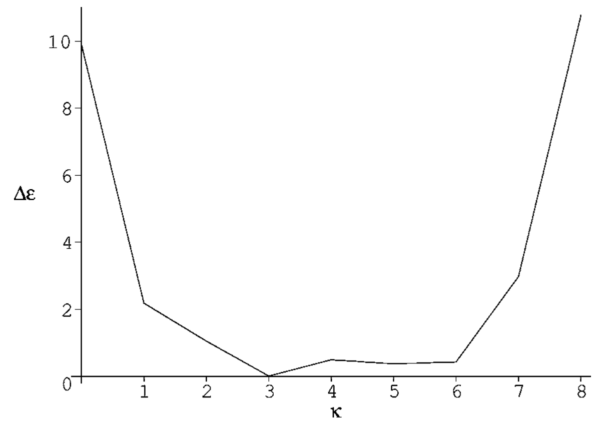

FIG. 7. The plot shows the values of the $|\Delta^k \varepsilon|$ against the vale for k, for entry 1016, for the heartbeat time series. The value of k that our procedure defines as the beginning of the plateau for this case is $k=3$.

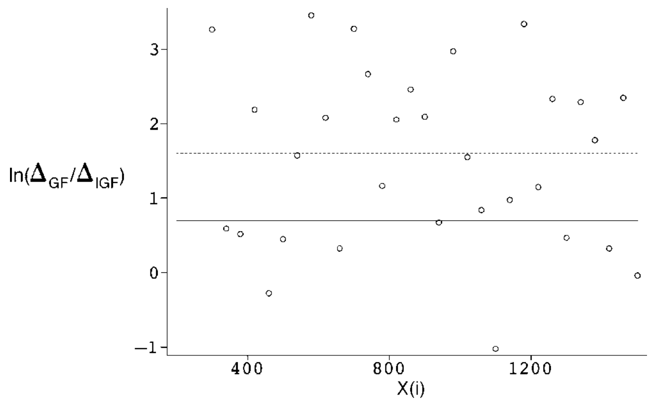

FIG. 8. The plot is for $\ln(\Delta_{GF}/\Delta_{IGF})$ against the position in the time series (i). The solid line marks the threshold where, above it, Δ_{IGF} starts to be the half of Δ_{GF} and the dotted line the threshold where, above it, Δ_{IGF} starts to be the fifth of Δ_{GF}.

reason for this application is to use the method on a controlled arena; i.e., we can see the method working at its best. What do we mean by its best? Could we not have gotten better results than the ones presented in Sec. III A? Of course we could have, for instance, if we have made the time series more "dense;" i.e., if we had used smaller values for Δt, of course, the results would be better. Indeed, we can do the same indefinitely up to infinite precision. What we mean by "its best" is the fact that there is not, for sure, any high-dimensional behavior. We have then demonstrated that the ideas behind our method work quite nicely.

The second application corresponds to a time series obtained from measurements; i.e., we do not have any prior knowledge about the (possible) dynamic system underlying it. We have found that, after the usual techniques have been used to produce the global mapping, we could improve the forecast capabilities of the fitting quite a bit (see Sec. III B), thus demonstrating the practicality of our approach in an uncontrolled situation.

Our method has, of course, its limitations. Perhaps the most obvious one is the fact that it will not help much in the case where the time series is "sparse;" i.e., as we have mentioned just above, as Δt becomes large, the method will not work. The limitation so far is that we do not have a criterion, as yet, to, just by quickly inspecting the time series, determine if our method applies well or not. One has to have a go and, in a testing arena, verify if the method is improving things.

That leads to future work: the production of a fast algorithm to test the time series for applicability (or not) of the method. One other possible line of research to be pursued is to improve our algorithm in the sense of using more information contained in the plateau than we are using now. So far, we are taking the first piece of data on the plateau, but as we have explained in Sec. II C, the values for the corrections will oscillate from that point on. It is reasonable to look for an algorithm to extract information from this oscillation.

[1] H. Abarbanel, R. Brown, J. Sidorowich, and L. Tsimring, Rev. Mod. Phys. **65**, 1331 (1993); Physica D **35**, 335 (1989); J. Farmer, and J. Sidorowich, Phys. Rev. Lett. **59**, 845 (1987).
[2] K. Alligood, T. Sauer, J. Yorke, *Chaos—An Introduction to Dynamic Systems* (Springer, Berlin, 1996); S. Strogatz, *Nonlinear Dynamics and Chaos* (Addison-Wesley, Reading, MA, 1994).
[3] H. Kantz, and T. Schreiber, *Nonlinear Time Series Analysis* (Cambridge University Press, Cambridge, England, 1997).
[4] H. Abarbanel, *Analysis of Observed Chaotic Data* (Springer-Verlag, New York, 1996).
[5] H. Abarbanel, R. Brown, J. Sidorowich, and L. Sh. Tsimring, Rev. Mod. Phys. **65**, 1331 (1993).
[6] R. Hegger, H. Kantz, and T. Schreiber, Chaos **9**, 413 (1999).
[7] T. Schreiber, Phys. Rep. **308**, 1 (1999).
[8] D. Kugiumtzis, B. Lillekjendlie, and N. Christophersen, Model. Identif. Control **15**, 205 (1994); **15**, 225 (1994).
[9] E. Lorenz, J. Atmos. Sci. **20**, 130 (1963).
[10] F. Takens, in *Dynamical Systems and Turbulence*, edited by D.A. Rand and L.S. Young, Springer Lectures Notes in Mathematics Vol. 898 (Springer-Verlag, New York, 1981), p. 366.
[11] P. Grassberger and I. Procaccia, Phys. Rev. A **28**, 2591 (1983); A. Cohen and I. Procaccia, Phys. Rev. A **31**, 1872 (1985).
[12] M. Kennel, R. Brown, and H. Abarbanel, Phys. Rev. A **45**, 3403 (1992); T. Sauer, J. Yorke, and M. Castagli, J. Stat. Phys. **65**, 579 (1991).
[13] H. Stephani, in *Differential Equations: Their solution using symmetries*, edited by M. A. H. MacCallum, (Cambridge University Press, New York, 1989).
[14] G. W. Bluman, and S. Kumei, *Symmetries and Differential Equations, Applied Mathematical Sciences*, Vol. 81 (Springer-Verlag, Berlin, 1989).
[15] P. J. Olver, *Applications of Lie Groups to Differential Equations* (Springer-Verlag, New York, 1986).
[16] W. Sarle (unpublished).
[17] L. G. S. Duarte, L. A. C. P. da Mota, H. P. de Oliveira, R. O. Ramos, and J. E. F. Skea, Comput. Phys. Commun. **119**, 256 (1999).