Software to build dynamical systems models from time series with chaotic behavior

J L Cruz1, R M Gutiérrez1,2, and C G Pastrán1,3
1 Centro de Investigaciones en Ciencias Básicas y Aplicadas, Universidad Antonio Nariño, Bogotá, Colombia
2 Science Division, New York University, Abu Dhabi, United Arab Emirates
3 Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia
E-mail: jcruz40@uan.edu.co

Abstract. In physics and other sciences there are dynamic systems that exhibit chaotic behavior. Understanding and predicting the behavior of these systems depends on the ability to build models of them. That is why software written in the python language is being developed that allows us to experiment numerically in an agile and precise way with different models of the dynamics of a time series that presents chaotic behavior. This software has a graphical interface divided into three sections. The first section has tools that allow you to interact with the built models. The second section has tools that allow you to establish whether a time series can exhibit chaotic behavior. The third section has tools that allow models to be built from time series. Finally, using the Lorenz system, an example of the use and utility of this software for the construction of models from time series and its possible usefulness in real data is presented.

1. Introduction
Building models from time series has always been a major challenge for physics and other sciences and is more complicated when trying to modeling dynamic systems with chaotic behavior. Chaos is a behavior that some deterministic nonlinear dynamic systems exhibit, characterized by being unpredictable in the long-term and presenting a strange attractor in the phase space. In the nonlinear time series analysis literature, there are different methods to identify when a time series has a possible chaotic behavior and how to build models from them. All these methods are implemented in computer programs written in various programming languages. This diversity can represent difficulties when working on chaos modelling from time series, among which we can highlight: Not finding all the necessary methods programmed in the same language, routines programmed many years ago that no longer conform to the evolution and current standards of programming languages and may not be able to be executed or are run with errors, very complicated routines to understand and use, lack of a structure in the routines or software that guide you from the identification of chaos in the time series to the construction of models.

The most important attempts to implement several of the time series analysis methods in one language are: the Chaos Data Analyser software [1] and the time series analysis (TISEAN) package [2]. The first is software written in power basic by Sprott J C in 1998 [1] and although its interface is intuitive due to a neat menu of options, it lacks ease and agility in its use. The second is a set of routines written in C++ that allows you to identify chaos in time series and create models using
autoregressive–moving-average (ARMA) and least squares methods. However, this set of routines lacks a step-by-step structure and is becoming obsolete in terms of the new standards and has not been updated or maintained since 2007.

To solve these problems, chaotic model reconstruction from time series (CMRTS) was created, a software written in Python that is a modern programming language. It does not cover all the methods of nonlinear time series analysis, but rather those necessary and sufficient to go from the chaos identification in time series to the model’s construction. The models created are 3-dimensional second order polynomials. Three dimensions because it is the minimum dimension in which any chaotic dynamics [3] can be represented and second order to capture the nonlinearities of the system. Furthermore, CMRTS has a graphical interface divided into three parts that serves as a step-by-step guide and allows an easy, fast and efficient interaction to create and test models, saving time and avoiding unnecessary calculations.

Section 2 presents a brief description of the methods, routines and libraries that have been used and are necessary for the CMRTS operation; section 3 presents the graphical interface and its use. Section 4 presents some results obtained using CMRTS; in section 5 the conclusions of the use of CMRTS and future perspectives.

2. Methods and libraries

The methods that were used to go from the identification of a possible chaotic behavior in the time series to the modeling of the same are presented below. There is also a brief description of the libraries that were created and their functionality in the construction of the models.

2.1. Methods

The following methods are used to identify chaos in time series: The power spectrum to identify if the time series does not show periodic or stochastic behavior, the false nearest neighbors to calculate the embedding dimension in which it is possible to reconstruct the dynamics of the system and as an estimator of the noise level in the time series, largest Lyapunov exponent that if positive indicates sensitivity to initial conditions that is a characteristic of chaos, fractal dimension indicating if the reconstructed attractor from time series is a strange attractor another feature of chaos [4,5].

The modeling is made from the procedure shown in [6] and is designed to build models of the form shown in Equation (1), where $\vec{x}$ is the vector of variables $(x,y,z)$, taking into account that these variables are not spatial but represent variables of the phase space of the system under study. $\vec{c}_x$ is the vector of coefficients and $\vec{\Pi}(\vec{x})$ is a orthonormalized polynomial base; to compute $\vec{c}_x$ and $\vec{\Pi}(\vec{x})$ from a time series we use the following methods: The Takens-Mañe [7,8] reconstruction method to “convert” the time series into a 3-dimensional vector of the form $(x,y,z) = (x_n, x_{n+\tau}, x_{n+2\tau})$, where $x_n$ is the time series and $\tau$ is the reconstruction delay time, which is calculated using mutual information theory, the Gramm-Schmidt orthonormalization method that guarantees linear independence of polynomials $\vec{\Pi}(\vec{x})$, the Adams-Moulton method and the least squares method to adjust the values of the $\vec{c}_x$ coefficients of the model. Finally, with these calculated coefficients the model can now be constructed as shown in Equation (1) [6].

$$\hat{x} = F_\vec{x}(\vec{x}) = \vec{c}_x \cdot \vec{\Pi}(\vec{x})$$  \hspace{1cm} (1)

2.2. Libraries

For the proper functioning of CMRTS it is recommended to have the following python libraries installed: Numpy [9] and Scipy [10] which are used for various numerical calculations, Matplotlib [11] is the most complete python library to create graphs, WxPython which is the fork for python of WxWidgets used to create the user interfaces and nolitsa which is an independent library created by Mannattil M [12] and used in its article for the analysis of non-linear time series.
On the other hand, the following libraries were created independently: Mirk4 that implements the fourth order Runge-Kutta method and serves to numerically integrate the 3-dimensional and second order models created with the use of CMRTS. The CMRTS library has also been created, which is the main library of CMRTS and allows to create models from time series. This library is based on the work done in [6] and finally CMRTS_GUI a library that contains the constructor of the CMRTS’s graphical interface.

3. Chaotic model reconstruction from time series functionalities
Figure 1 shows the appearance of the CMRTS graphical interface, here you can see that it has 3 sections and a notepad to keep track of the experiments and take notes.

3.1. The model sections
The first section called Model is made up of a table that contains all the coefficients that the model can have, in total 30. After this have a series of buttons that allow us to interact quickly with the model represented in the coefficient table (CT). The first button Lorenz places the coefficients of the well-known Lorenz model on CT, as shown in Figure 2. Then there is the Gen ST button that allows the integration of the model that is in the CT, here we use the Mirk4 library. Then there are the buttons that allow you to make different plots (Plot x, Plot y, and Plot z) with which the variables x, y, z are plotted respectively, 3D Plot that represents the 3 variables in the phase space (Plot y vs x, Plot z vs x, and Plot z vs y) that allow visualizing the attractor from the different xy, xz and yz planes. Then there is the Clear button that zeroes all the CT coefficients.

After these buttons are a series of text boxes that allow you to have control over the aspects of the model interaction. These aspects are: N which is the number of iterations made in the model integration; Dt is the step between iterations of the Runge-Kutta method; Tau is the delay time used
when working with reconstruction; Load has the name of the file that contains the time series that will be analyzed; Save has the name with which the file that contains the model integration data is saved.

3.2. The pre-process sections
This section has five buttons that allow you to calculate and graph the methods used to identify if there is possible chaos in the time series. Spectrum is used to see the power spectrum of the time series. Find Tau is used to calculate the best $\tau$ that can be used in the reconstruction, for this the method of mutual information is used, it also shows in a graph how the reconstructed attractor would look using the calculated $\tau$. False nearest neighbors (FNN), it allows us to see graphically the behavior of the method of false nearest neighbors to deduce the minimum dimension in which the dynamic system model could be represented, as in CMRTS the models have 3 dimensions this method is used to estimate the amount of noise that the time series has. Lyapunov is used to estimate the largest Lyapunov exponent and thus verify if the time series has sensitivity to the initial conditions. D2 is used to estimate the fractal dimension of the time series, it is used in two ways graphically and numerically.

3.3. The process sections
In this section there are 3 buttons that guide us in the process of building models from 3 time series. This is the shortest section in the graphical interface, but the most important in fact this section is the one that gives the name to CMRTS. The Model button implements the CMRTS library that is responsible for building the model. In this case we can use two options, model from a time series or model from a 3-dimensional data vector, data to be used must be saved in a plain text file and the name of this file should appear in the Load box of the model section. If you are going to model from a 3-dimensional data vector, the Tau box must be equal to zero and if you want to model from a time series, the Tau box must be different from zero, preferably use the value that is I calculate using Find Tau. The Set Model button fills the CT cells with the coefficients that were calculated using Model. The Add Noise button gives us the option of contaminating the time series with white Gaussian noise to test the effects of noise on the construction of the models or on their dynamics. Finally, there is an area called notes and here you can take notes in the form of plain text.

4. Results obtained using chaotic model reconstruction from time series
To show how CMRTS works, the Lorenz model will be used, which is a well-known and studied model in chaos theory and by default, appears in CMRTS as can see in Figure 2. Just click on the Lorenz button and this appears on the CT. We integrate the model using the Gen ST button and we already have the file called TestST.dat to start working. The first thing we can do is graph the integrated model. In Figure 3(a) you can see the graphic representation of the variable $x$; in Figure 3(b) next you see the Lorenz attractor which represents the 3 integrated dimensions and in Figure 3(c) the attractor appears seen from the xz plane. This is just a sample of the graphics that you can do.

![Figure 3](image_url)

**Figure 3.** Graphs obtained from the Lorenz model: (a) time series of the $x$ variable; (b) the Lorenz attractor, (c) the Lorenz attractor seen in the xz plane.
Figure 4 shows the graphs obtained using the tools in section 2 of CMRTS. Figure 4(a) corresponds to the power spectrum that, as can be seen, does not correspond to a periodic or stochastic system, which indicates that the system could be chaotic; of course, in this case we know that it is chaotic. Figure 4(b) corresponds to the mutual information, from here the recommended delay time for the reconstruction is obtained, in this case $\tau = 16$. Figure 4(c) corresponds to the false nearest neighbors' method and is indicating that the minimum dimension in which the system can be reconstructed is 3, if this time series were not ideal, higher dimensions would be obtained. Figure 4(d) corresponds to the largest Lyapunov exponent calculation which, as can be seen, appears written in the title of the graph. In this section, the correlation dimension is also calculated, which in this case gives us a value of 2.11, which has a 2% relative error with respect to the accepted value in the literature, which are 2.068.

![Figure 4](image)

**Figure 4.** Obtained results by using the section 2 tools in the Lorenz system: (a) the power spectrum; (b) average mutual information to find the reconstruction time delay $\tau$; (c) embedding dimension using the false nearest neighbors, and (d) largest Lyapunov exponent estimation.

Figure 5 shows 3 different graphs that correspond to the model’s construction by varying some parameters. In Figure 5(a) appears the attractor recovered using the 3 integrated variables of the Lorenz model, that is, when the CMRTS Tau box is equal to zero, in this case no reconstruction is performed. In Figure 5(b) appears the attractor obtained by using only the integrated variable $x$ of the Lorenz system, in this case the reconstruction is done using $\tau = 16$, which was obtained previously. Also, at first glance, this attractor does not correspond to the Lorenz attractor. In Figure 5(c) appears one of the many advantages of CMRTS, to be able to vary the "Tao" of the reconstruction, build the model and test it very quickly, thus we have obtained an attractor that at least visually has a more similar appearance to Lorenz attractor, using a $\tau = 69$.

It should be clarified that the exact coefficients of the Lorenz model were only obtained when the integrated data of the 3 variables were used. When using a single variable applying the reconstruction theorem, in no case seen so far have the exact Lorenz coefficients been found. With these results we
can see that CMRTS is very useful in the model’s construction of dynamic systems, from a time series. In this way we have created a useful tool in the study of physical systems with proven chaotic behavior. In addition to having other areas of application such as economics, engineering, and population behaviors. With the advantage of being able to create and test models quickly, allowing to have several tests of the same system in a short time.

5. Conclusions
With CMRTS it has been possible to have software programmed in Python, a modern and easy to understand language, which also allows us the possibility of working with a graphical interface and this helps to save much time both in the analysis of time series and in the construction of the models from them. CMRTS can be used with real data and build models from laboratory observations. It can be used to model a variety of physical systems that exhibit chaotic behavior, from the creation of stars and planetary movements, to the weather or the human brain.

CMRTS also continues to evolve according to specific needs and later it will also integrate a routine that is already in development and allows you to do the entire modelling procedure automatically, it will be enough to have the data and press a button. For now, we have managed to build models one after another of the same time series but varying the reconstruction time, constructing around 100 different models in less than 10 minutes, taking into account that we do calculations using a 3 MHz processor. This allows us to test many models in a short time, which is how we discover the attractor shown in the third graph of Figure 5. To test the usefulness of CMRTS to real physical systems, it is necessary to test how much noisy time series affects the construction of the models, the minimum amount of data necessary to obtain a model, and the incidence of the time series sampling frequency in the model construction.

Acknowledgments
The authors acknowledge to Colciencias, Colombia, for financial support with equipment and scholarships through calls # 727 and #757.

References
[1] Sprott J C 2003 Chaos and Time-Series Analysis (Oxford: Oxford University Press)
[2] Hegger R, Kantz H, Schreiber T 1999 Practical implementation of nonlinear time series methods: The TISEAN package Chaos: An Interdisciplinary Journal of Nonlinear Science 9(2) 413
[3] Robinson J C 1998 All possible chaotic dynamics can be approximated in three dimensions Nonlinearity 11(3) 529
[4] Abarbanel H D I 1996 Analysis of Observed Chaotic Data (New York: Springer-Verlag)
[5] Schuster H G, Just W 2005 Deterministic Chaos (Weinheim: WILEY-VCH)
[6] Gutierrez R M 2004 Optimal nonlinear models from empirical time series: An application to climate, International Journal of Bifurcation and Chaos 14(6) 2041
[7] Takens F 1980 Dynamical systems and turbulence *Lecture Notes in Mathematics* vol 898, ed Dold A, Eckmann B (Berlin: Springer-Verlag) p 366
[8] Mañe R 1980 Dynamical Systems and turbulence *Lecture Notes in Mathematics* vol 898, ed Dold A, Eckmann B (Berlin: Springer-Verlag) p 230
[9] Oliphant T E 2015 *A guide to NumPy, 2nd Edition* (USA: CreateSpace Independent Publishing Platform)
[10] Virtanen P, Gommers R, Oliphant T E, Haberland M, Reddy T, Cournapeau D, Van der Walt S J 2020 SciPy 1.0: fundamental algorithms for scientific computing in Python *Nature Methods* 17(3) 261
[11] Hunter J D 2007 Matplotlib: A 2D graphics environment *Computing in Science & Engineering* 9(3) 90
[12] Kim J H, *et al.* 2016 The AGORA high-resolution galaxy simulations comparison project. II. Isolated disk test *The Astrophysical Journal* 833(2) 202:1