Computer and Information Science; Vol. 5, No. 4; 2012
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Simulation and Visualization of Chaotic Systems

Athanasios I. Margaris[1]

[1] TEI of Larissa, Department of Computer Science and Telecommunications, Larissa, Thessaly, Greece

Correspondence: Athanasios I. Margaris, TEI of Larissa, Department of Computer Science and Telecommunications, Larissa, Thessaly, Greece. E-mail: amarg@uom.gr

Received: April 16, 2012 Accepted: May 7, 2012 Online Published: June 1, 2012

doi:10.5539/cis.v5n4p25 URL: http://dx.doi.org/10.5539/cis.v5n4p25

Abstract

The objective of this paper is to present a suite of applications that allow the simulation and study of chaotic systems, as well as the estimation of the most important properties associated with them. These applications implement fundamental algorithms from the field of chaotic system dynamics, such as the reconstruction of the system trajectory in the appropriate embedding space, and the estimation of the Lyapunov exponents and the fractal dimension. Furthermore, they provide additional features such as the study of bifurcation diagrams and the detection of chaotic regions in the parameter space. The current version of the applications has been developed in the programming framework of Visual C++ 6.0 and they can be used under the operating system of Microsoft Windows.

Keywords: chaotic systems, reconstruction, fractal dimensions, Lyapunov exponents, unstable periodic orbits

1. Introduction

Chaotic dynamics is one of the most interesting branches of the physical sciences and the development of algorithms and techniques allowing the study of dynamical systems, has been paid with great attention the last decades. This rapid development of nonlinear system study it can be definitely attributed to the enormous advances in the field of computing systems, since this study is performed via simulation techniques that require high computational cost as well as a large amount of processing time.

There are many applications developed during the last years for studying chaotic systems. Typical examples include the AUTO continuation and bifurcation software for ordinary differential equations (Doedel et al., 1997), the CANDYS/QA package for numerical bifurcation analysis of dynamical systems (Feudel & Jansen, 1992), the CONTENT multi-platform environment for dynamical system study (Kuznetskov & Levtitin, 1997), the DDE BIFTOOL MATLAB package for numerical bifurcation analysis of delay differential equations (Engelborghs, 2000), the DSTool dynamical system toolkit for exploring dynamical systems (Back et al., 1992), the Dynamics / Numerical Explorations package for the exploration of diffeomorphic and non invertible two dimensional maps (Nusse & Yorke, 1997), the GAIO experimental software for the numerical investigation of dynamical systems (Dellnitz & Hohmann, 1997), and the TISEAN time series analysis tool based on the usage of surrogate data (Hegger, Kantz, & Schreiber, 1999). Thses applications must support a lot of different functions associated with chaotic data, the most important of them are the following:

- The generation of chaotic time series of known chaotic systems (such as the Henon, the Rossler and the Lorenz attractors) or the import of unknown time series produced by experiments of any type (a typical example is the EEG time series emerged by electroencephalograms).
- The plotting of the time series in the computer screen.
- The ability to detect determinism in a time series and the rejection of time series associated with random mechanisms.
- The reconstruction of the system trajectory in the appropriate embedding space with the proper values of the embedding dimension d and the time delay τ to be selected automatically by the application.
- The quantitative characterization of the reconstructed trajectory via the prediction of parameters such as the Lyapunov exponents, the fractal dimensions (i.e. the capacity, information and correlation dimension) as well as the topological entropy.
- The identification and plotting of the basin of attraction for each attracting point of the system.

- The generation and study of bifurcation diagrams for any one of the system parameters.
- The experimental detection of chaotic regions in the parameter space of the dynamical system.
- The extraction of unstable periodic orbits and the description of the system in the phase space.

There are many applications of this nature that work in text mode and under Unix-like operating systems, and each one of them provides only a subset of the above functions. From this point of view, the construction of a Windows application supporting the complete set of chaotic system characterization algorithms it is considered an important task. In the following description, the interface of such an application is presented, together with the computations applied to the chaotic data as well as the visualization of them in the computer screen. Some of the algorithms can be applied to any time series of known or unknown nature, while some others are 'hardwired' to specific system examples in order to point out the design issues and the user interface techniques that can be used to provide a friendly and easy to use application. A detailed description of those applications is presented in the following sections.

2. Chaotic Time Series Generation

Time series generation is a fundamental utility of an application used for the study of the chaotic systems, since it provides to the user data series for further processing. The application supports the generation of time series for the most important chaotic systems, which in general are the following:

(1) The one dimensional logistic map defined by the equation

$$x_{n+1} = \lambda x_n (1 - x_n) \tag{1}$$

A typical value for the λ parameter that leads to a chaotic behavior is the value λ=3.95.

(2) The two dimensional Henon map defined by the equations

$$\begin{aligned} x_{n+1} &= 1 - \alpha x_n^{\,2} + \beta y_n \\ y_{n+1} &= x_n \end{aligned} \tag{2}$$

In this system, the most commonly used parameter values, are the values α=1.28 and b=-0.3.

(3) The three dimensional Lorenz attractor defined by the system of differential equations

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= \rho x - xz - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \tag{3}$$

or in discrete form

$$\begin{aligned} x_{n+1} &= x_n + \Delta t(\sigma x_n - \sigma y_n) \\ y_{n+1} &= y_n + \Delta t(\rho x_n - x_n z_n - y_n) \\ z_{n+1} &= z_n + \Delta t(x_n y_n - \beta z_n) \end{aligned} \tag{4}$$

The most commonly used parameter values are the values σ=10, β=8/3, ρ=28 and Δt=0.06.

(4) The two dimensional Ikeda map defined by the equations

$$\begin{aligned} x_{n+1} &= R + C_2(x_n \cos\tau_n - y_n \sin\tau_n) \\ y_{n+1} &= C_2(x_n \sin\tau_n + y_n \cos\tau_n) \end{aligned} \tag{5}$$

where $\tau_n = C_1 - C_3/(1 + x_n^2 + y_n^2)$ and typical parameter values R=1, C_1=0.4, C_2=0.9 and C_3=6.

(5) The three dimensional Rossler attractor defined by the system of differential equations

$$\begin{aligned} \frac{dx}{dt} &= -y - z \\ \frac{dy}{dt} &= x + \alpha y \\ \frac{dz}{dt} &= \beta + (x - c)z \end{aligned} \tag{6}$$

or in discrete form

$$\begin{aligned} x_{n+1} &= x_n - \Delta t(y_n + z_n) \\ y_{n+1} &= y_n + \Delta t(x_n - \alpha y_n) \\ z_{n+1} &= z_n + \Delta t[\beta + (x_n - c)z_n] \end{aligned} \tag{7}$$

with typical parameter values α=0.1, β=0.1, c=18 και Δt=0.06.

Besides the above chaotic time series, the application also supports the generation of a random time series with the seed value of the random number generating function to be defined by the user. In the case of d-dimensional chaotic systems (d>1) the user can store the data values of the individual components in separate data files for further processing. In each case, the user has to specify the number of samples of the time series and the name of the data file to be created. Figure 1 shows the Dialog Box used for the generation of the time series associated with the previously described systems.

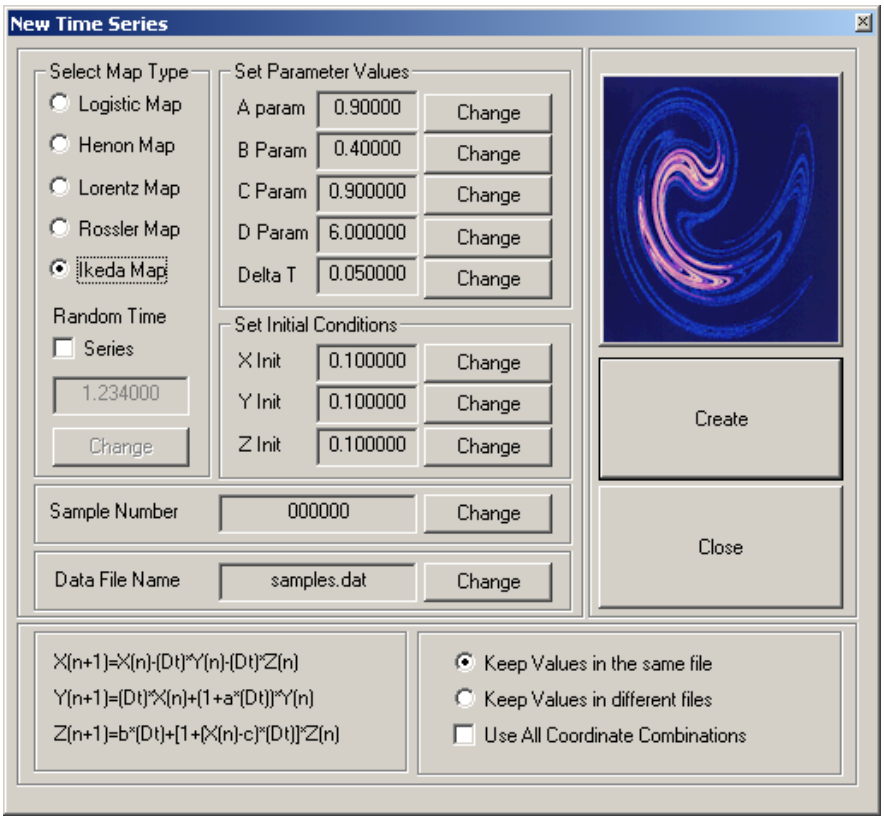

Figure 1. Generation of time series associated with the most commonly used chaotic systems

The import of a time series emerging from a measurement process is also supported by the application. This is the most interesting case, since the complete characterization of an unknown time series is a challenging aspect; this characterization includes the estimation of parameters such as the Lyapunov exponents and the fractal dimensions, the identification of the chaos transition mechanisms and the chaotic regions in the system phase space, as well as the periodic orbits of the system and the type of the stability describing them. These algorithms

can be found in the literature and those implemented here are briefly discussed.

The *Data Plotting* dialog box displays the time series under consideration in the computer screen and provides to the user a lot of different functions such as the ability to zoom in and out, to plot a grid for better preview and to display the coordinates (sample, position) associated with the current mouse cursor position. Since this is a trivial task, this dialog box is not presented here.

3. Phase Space Reconstruction

Phase space reconstruction is one of the fundamental stages in the processing of chaotic time series. This technique is based to the fact that the dynamics of a system defined in an abstract geometrical space, R^n can be completely determined by a recorded times series of one of its coordinates. The main operation of this technique is the construction of a new multidimensional space known as embedding space, used for the definition of the reconstructed trajectories of the system. By using the available time series described above, it is possible to reconstruct all the features of the system such as its fractal dimensions, its Lyapunov exponents, as well as its topological entropy.

The main method used for the reconstruction of the embedding space is the method of delays, that allows the construction of a d-dimensional point as a vector $\boldsymbol{r}(t)=\{x(t),x(t-\tau),x(t-2\tau),.....,x(t-(d-1)\tau)\}$ where d is the dimension of the reconstructed space, known as embedding dimension, and τ is the time delay of the system. According to Taken's theorem (Takens, 1981), if the embedding dimension d is related to the actual dimension of the phase space n, via the equation $d=2n+1$, then, the original and the reconstructed space are topologically equivalent and the corresponding trajectories are correlated via a diffeomorphic transformation.

The dialog box that allows the reconstruction of the system trajectory in the appropriate embedding space supports all the required functions associated with the reconstructed phase space such as the determination of the values of the embedding dimension and the time delay, the specification of the trajectory length, as well as the exclusion of the first N points of the time series (for any user-defined value of N), if they are interpreted as transient chaos. When the reconstruction terminates, the values of the reconstructed points are displayed in the main list box of the window; the user has the ability to delete the trajectory to reconstruct it again with other parameter values of to save it to a hard disk file.

The choice of the correct values for the embedding dimension d, and the time delay τ, is the subject of a lot of research efforts. The predefined values for these parameters are the values $d=3$ and $\tau=3$ but the user has the ability to use his/her own values. The program helps the user to choose the correct values of those parameters, by displaying an auxiliary dialog box named *Reconstruction Parameters Calculation*; this dialog box is shown in Figure 2.

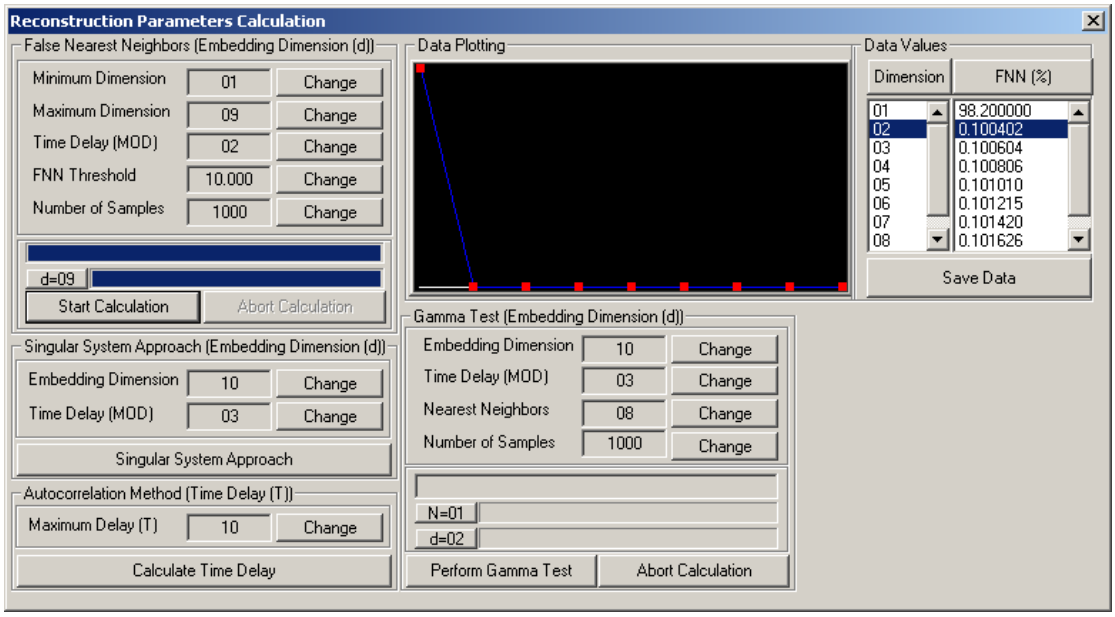

Figure 2. Estimation of the phase space reconstruction parameters

The *Reconstruction Parameters Calculation* dialog box, implements a lot of known methods used for the estimation of the optimum values for the reconstruction parameters, such as the False Nearest Neighbors (FNN method) (Kennel, Brown, & Abarbanel, 1992), the Singular System Approach (SSA method) (Broomhead & King, 1986) and the Γ-test method (Stefansson, Concar, & Jones, 1997) for the estimation of the embedding dimension d, as well as the autocorrelation method (Albano, Muench, Schwartz, Mees, & Rapp, 1988) for the estimation of the time delay τ. In Figure 2, the FNN method is used to calculate the optimum dimension of the embedding space for a time series emerged from the Henon attractor. The user can set the range of values for the tested dimensions - usually a maximum dimension d=10 is more than enough - and the time delay τ. After the termination of the calculation - all these procedures are multi-threaded - the program fills the list boxes at the right side of the dialog with the estimated values and plots the result in the computer screen. It is not difficult to notice that the optimum value of the embedding dimension for the Henon map, is d=2. Regarding the methods implemented by this dialog, they work, in general, in the following way:

(1) False Nearest Neighbors: The FNN method is based on the geometrical properties of the attractor as it is reconstructed in embedding spaces with continuously increasing dimensions. If the embedding dimension is not appropriate, the reconstructed attractor will be characterized by the presence of self-intersections; therefore, trajectory points that are far away from each other will be mapped to very close locations. On the other hand, if the attractor is reconstructed in the appropriate embedding space, the previously described self-intersections will be removed and the false nearest neighbors will be eliminated.

To describe the method, let us consider two points $\boldsymbol{r}=(r_1,r_2,....,r_d)$ and $\boldsymbol{s}=(s_1,s_2,....,s_d)$ in a d-dimensional embedding space with an Euclidean distance

$$R_d(r,s)=\sqrt{(s_1-r_1)^2+(s_2-r_2)^2+.....+(s_d-r_d)^2} \tag{8}$$

If the attractor is reconstructed in a d+1-dimensional space, it is not difficult to prove that the distances $R_d(\boldsymbol{r},\boldsymbol{s})$ and $R_{d+1}(\boldsymbol{r},\boldsymbol{s})$ are related through the equation

$$K=\frac{R_{d+1}^2(r,s)-R_d^2(r,s)}{R_d^2(r,s)}=\frac{(s_{d+1}-r_{d+1})^2}{R_d^2(r,s)} \tag{9}$$

Defining the quantity $R(\boldsymbol{r},\boldsymbol{s})=\sqrt{K}$ and a threshold value R_{thr}, the point **s** is characterized as a false nearest neighbor of the point **r** if the condition $R(\mathbf{r},\mathbf{s})> R_{thr}$ holds (a typical value of R_{thr} is $R_{thr}=2$). The identification of the optimum value for the embedding dimension requires the estimation of the percentage of the trajectory points associated with false nearest neighbors. This estimation is performed for each dimension in the typical range $(d_{min}, d_{max}) = (1, 9)$. In this case, the optimum dimension is the one associated with zero or negligible percentage of those points.

(2) Singular System Approach: In the SSA method the chaotic attractor is reconstructed using a large value for the embedding dimension m, much greater than the value predicted by the Taken's theorem. In the next step, we use the reconstructed vectors $x_k=\{x_k^{\,1},x_k^{\,2},.....x_k^{\,m}\}$, to construct the embedding matrix of dimension mxN where N is the number of reconstructed points.

$$X=\begin{pmatrix} x_1^1 & x_1^2 & x_1^3 & ..... & x_1^m \\ x_2^1 & x_2^2 & x_2^3 & ..... & x_2^m \\ x_3^1 & x_3^2 & x_3^3 & ..... & x_3^m \\ ..... & ..... & ..... & ..... & ..... \\ x_N^1 & x_N^2 & x_N^3 & ..... & x_N^m \end{pmatrix} \tag{10}$$

After the generation of the embedding matrix, the covariance matrix $\mathbf{C}=\mathbf{X}^\mathbf{T}\mathbf{X}$ can be created; this is an orthogonal matrix with dimensions mxm.

The identification of the appropriate value for the embedding dimension is based to the estimation of the eigenvalues of the covariance matrix by using the SVD (Singular Value Decomposition) method. In this method, the **C** matrix is written as the matrix product $\mathbf{C}=\mathbf{U}\Sigma\mathbf{V}^\mathbf{T}$ where **U** and **V** are orthogonal matrices, while, the diagonal matrix **Σ**, keeps in its main diagonal the m eigenvalues of the matrix **C**. The sorting of these eigenvalues, leads to the arrangement

$$\sigma_1 \le \sigma_2 \le \sigma_3 \le \cdots \le \sigma_d >> \sigma_{1d+1} \le \sigma_{d+2} \le \cdots \le \sigma_m \tag{11}$$

In this case, the optimum value for the embedding dimension d, is equal to the number of eigenvalues that are very large compared to the remaining values associated with the system noise.

(3) Γ-test method: this method is based on the assumption that the time series under consideration is associated with a dynamical system described by a smooth and continuous function φ: $R^m \rightarrow R$ and produces an output in the form $y=\varphi(\mathbf{x},r)=f(x_1,x_2,\ldots,x_n)+r$ where **x** is the input vector supplied to the system and r is a zero mean stochastic variable describing the noise associated with the estimation of the system output.

To describe the method, let us consider an arbitrary sample (**x**,y) and a second sample (**x**', y') such that the point **x**' to be the closest neighbor of the point **x** and let us define the statistical quantity

$$\gamma = \frac{1}{2M}\sum_{i=1}^{M}[(y'(i)-y(i))^2] = \frac{1}{2} < (y'-y)^2 > \tag{12}$$

with the parameter M to represent the number of the reconstructed points. If we denote as **x**(N(i,p)) the p_{th} nearest neighbor of the sample (**x**(i),y(i)) we can also define the quantities

$$\delta(h) = \frac{1}{M}\sum_{i=1}^{M}[(x(N(i,p))-x(i))^2] \tag{13}$$

$$\gamma(h) = \frac{1}{2M}\sum_{i=1}^{M}[(y(N(i,p))-y(i))^2] \tag{14}$$

for the h_{th} nearest neighbors of that sample. Summing these quantities for all the p nearest neighbors and dividing the estimated sum by their number, we get the new quantities

$$\Delta(p) = \frac{1}{p}\sum_{h=1}^{P}\delta(h) \tag{15}$$

$$\Gamma(p) = \frac{1}{p}\sum_{h=1}^{P}\gamma(h) \tag{16}$$

From these quantities, the quantity Δ(p) describes the mean square distance of the h≤p nearest neighbors, while the quantity Γ(p) is nothing more than a more accurate value of the γ statistical property defined above, calculated over the h≤p nearest neighbors.

The calculation of the pair (Δ(p),Γ(p)) for each sample (x(i),y(i)) and the estimation of the least squares line, leads to the equation $y = Ax + \Gamma$, where Γ is a measure of the variation of the statistical quantity r. In the ideal case of a noise free system, this quantity should be equal to zero, but in general it has a non zero but very small value.

By applying the Γ-test procedure for a set of dimension values in the typical range (d_{min}, d_{max}) = (1, 9), the optimum embedding dimension is estimated as the one associated with the smallest value of the parameter Γ.

(4) The autocorrelation method: the autocorrelation method allows the calculation of the optimum value for the time delay τ. Recalling from the basic theory that the autocorrelation between a time series x(t) and its delayed version x(t+τ) is given by the equation

$$r(\tau) = \frac{\sum_{k=1}^{N} x(k)x(k+\tau)}{\sum_{\kappa=1}^{N}|x(k)|^2} \tag{17}$$

it can be proven that for a stochastic or a chaotic time series the above equation can be reduced in the form $r(\tau)=\alpha\exp(-\tau/T)$ where T is the autocorrelation time. Based on this description we can estimate the optimum value for the time delay τ by setting τ=T; in this case, the autocorrelation function it reduces to the (1/e) of its initial value. Another frequently used approach, considers as the optimum time delay value, the value of τ for which the autocorrelation function is zeroed for the first time; in such a case, the time series x(t) and x(t+τ) are completely uncorrelated each other.

After the termination of the phase space reconstruction procedure the user can plot the reconstructed trajectory in

the computer screen. The current version of the program allows the plotting only in 2-D space, while for higher dimensions the trajectory points are saved to a plain text file compatible with the GNU Plot plotting utility, allowing thus the user to plot the trajectory by using this application.

4. Quantitative Characterization

The quantitative characterization of a chaotic system, is associated with the calculation of measurable indices of chaotic behavior from the reconstructed trajectory such as the Lyapunov exponents and the fractal dimensions. In this section, custom made tools for performing such a characterization are presented; these tools provide also the ability to construct and study bifurcation diagrams of any type to draw conclusions regarding the behavior of the system under consideration.

In order to point out design guidelines for the construction of such an application, the characteristics of a specific chaotic system will be extracted. This system is the simple recurrent neuron model studied by Pasemann (Pasemann, 1993; Pasemann, 1997) that is ideal for this presentation since it has a lot of free parameters. According to the theory of the time space neural models, the resistance capacitance model (RC model) describes the post synaptic neuron potential as an RC circuit characterized by an electric current flow (Haykin, 1994). In this model, the synaptic weights, w, are modeled as conductivities, while, the resistance R, simulates the non linearity that produces the neuron output. By applying the Kirchhoff's law to this simple electric circuit, the time evolution of the neuron potential is given by the differential equation

$$C_j \frac{d\upsilon_j(t)}{dt} + \frac{\upsilon_j(t)}{R_j} = \sum_{i=1}^{N} w_{ij} x_i(t) + \vartheta_j \tag{18}$$

where $\upsilon_j(t)$ is the potential of the j_{th} neuron of the network, R_j is the resistance of the RC neuron model, C_j is the leakage capacitance and θ_j is the neuron bias unit. The solution of this equation is the function $\upsilon_j(t)$ that gives the variation of the neuron potential υ_t as a function of time, t. In the above equation it is implied that the j_{th} neuron of the network is connected to N input neurons.

Even though this approach can be applied to describe any neuron type from the neurodynamics perspective, in this presentation is used for the sake of simplicity, a more restricted model. In this limited model, the neural network has only one neuron with a self interaction component (i.e. an output synapse that sends the neuron output back to its input). If we denote the synaptic weight as w and the bias unit as θ, the above equation can be written in the simpler form

$$\frac{d\upsilon(t)}{dt} = -\frac{\upsilon(t)}{R} + wx(t) + \vartheta \tag{19}$$

In this form, the leakage capacitance has been set to unity and the index j is not used anymore, since this simple model is characterized by the presence of only one neuron. Furthermore, the neuron output x(t) is modeled by the sigmoidal function

$$x(t) = \frac{1}{1 + exp[-\upsilon(t)] / c} \tag{20}$$

with a sigmoidal slope equal to 1/c and therefore, the state equation of the single recurrent neuron dynamical system gets the form

$$\frac{d\upsilon(t)}{dt} = f[\upsilon(t)] = -\frac{\upsilon(t)}{R} + \frac{w}{1 + exp[-\upsilon(t)] / c} + \vartheta \tag{21}$$

The chaos exploration and detection techniques that presented in the next sections are based to the discrete model emerged by this equation. However, the extension of those techniques to the case of any chaotic dynamical system is straightforward.

4.1 The Discrete Recurrent Neuron Model

The study of the recurrent model for the continuous case can be performed by using the Runge-Kutta method, since the presence of the exponential term in the sigmoidal function does not allow the analytic study of the model. The application of those method shows that the time series of the post synaptic potential always converges to a single value whose sign depends on the sign of the various system parameters.

On the other hand, the appearance of periodic and chaotic effects in the time evolution of the neuron activation

potential υ(t) requires the transformation of the continuous model described above to a discrete form. The conversion of the differential equation (21) to a difference equation, leads to the result

$$\upsilon_{n+1} = \left(1 - \frac{\Delta t}{R}\right)\upsilon_n + \frac{w\Delta t}{1 + exp[-\upsilon(t)] / c} + \vartheta\Delta t \tag{22}$$

The above equation is a recursive one, and therefore, one can choose an initial value υ_0 and generate a time series by iterating it. In this procedure, the time interval Δt determines the time scale of the system, while, the quantities w, R and θ can be treated as free parameters whose values affect the shape and the nature of the system time series. More specifically, it can be proven through experimentation, that this time series can be converging, periodic or chaotic for different combinations of the above parameters.

4.2 Creating and Exploring Bifurcation Diagrams

The chaotic features of the dynamical system under consideration can be presented by appropriate bifurcation diagrams that show the stable fixed points of the system for a range of values of the control parameters. In these diagrams the horizontal axis represents the parameter of interest, while, the vertical axis is associated with the values of the neuron synaptic potential. There are five such diagrams, namely the weight bifurcation diagram, the bias bifurcation diagram, the time interval bifurcation diagram, the internal resistance bifurcation diagram and the sigmoidal slope bifurcation diagram. The main screen of the program shows the bifurcation diagram of the selected parameter, with a typical example to be shown in Figure 3.

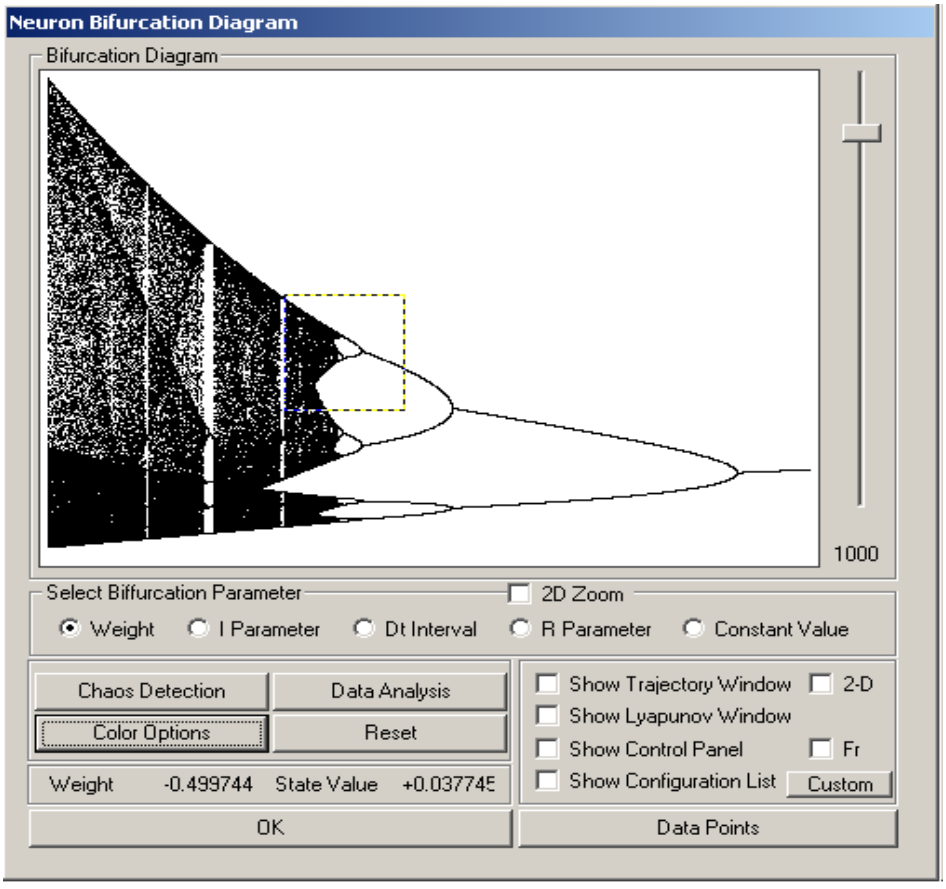

Figure 3. A typical weight bifurcation diagram

The generation of bifurcation diagrams can be easily performed by applying the following algorithm (Alligood, Sauer, & Yorke, 1996):

1) Set the parameter of interest α to the minimum value α_{min} of the desired value range for which the bifurcation diagram is going to be drawn.

2) Choose a random value υ for the neuron synaptic potential in the interval [0,1].

3) Calculate the associated system orbit by iterating the recursive equation (22).

4) Ignore the first 100 iterates and plot the orbit beginning with iterate 101.

5) Increment the parameter value using the predefined step and apply the whole procedure again until the parameter of interest to reach its maximum value, α_{max}.

The plotting of points produced in this way, approximate either fixed or periodic sinks or their attracting sets.

Figure 3 shows all the available user options such as the selection of the appropriate diagram type, the specification of the density of diagram points (the vertical slider control), as well as the display of the coordinates of those points. This dialog provides also the ability to apply a lot of useful techniques to the chaotic data by using a set of controls the most important of them are the following:

- A push button entitled *Chaos Detection* that opens the *Chaos Detection* dialog box, allowing thus the experimental detection of chaotic states for the various combinations of the values of the system parameters.
- A push button entitled *Data Analysis* that allows the characterization of the system trajectory for the continuous neuron model by applying the Runge-Kutta numerical method.
- A push button entitled *Data Points* that displays the coordinate values of each point of the current bifurcation diagram. Furthermore, if the *Trajectory* and the *Lyapunov Exponent* child windows are active, the values of the data points and the Lyapunov exponent of the trajectory associated with the current bifurcation diagram point are also shown.
- A check box entitled *Show Trajectory Window* that displays or hides the *Trajectory* child window. The system trajectory is shown in an 1-D fashion if the *2-D* check box is unchecked or in a 2-D fashion, otherwise.
- A check box entitled *Show Lyapunov Window* that displays or hides the *Lyapunov* child window.
- A check box entitled *Show Configuration List* that displays or hides the *Configuration List* child window.

The determination of the values of the system parameters is performed by a child window entitled *Control Panel* via a very simple and user friendly interface. The program also provides a drag-and drop based zoom capability in the currently selected bifurcation diagram allowing thus the exploration of the chaotic nature of the system.

4.3 Identification and Plotting of the System Trajectory

The plotting of the system trajectory in the computer screen is supported by the *Trajectory* window in conjunction with the *Bifurcation* window: as the user moves the mouse cursor in the bifurcation diagram, the contents of the trajectory window are updated accordingly in real time, to draw the new trajectory. Recalling that each point in the bifurcation diagram corresponds to a different trajectory whose stable fixed points are plotted against the selected parameter value, it is clear that the converging, periodic, and chaotic trajectories shown in the trajectory window are mapped to appropriate regions in bifurcation diagram for which the system behaves in a convergent, periodic, or chaotic way.

The real time generation of the system trajectory is based to the retrieval of the current value of the bifurcation parameter as well as the fixed values of the remaining parameters and to the extraction of the time series by using the recursive equation (22). For each one of the trajectories shown in the *Trajectory* window, the program displays the associated parameter value as well as the value of the calculated Lyapunov exponent. Closely related to the *Trajectory* window, is the *Trajectory Properties* child dialog box, whose main function is the calculation of the Fourier transform of the current trajectory and the plotting of the Fourier coefficients and the power spectrum. The application also supports the display of a graph in the form *{log[P(f)]-log(f)}* that can be used to distinguish between a white noise and a chaotic time series: the white noise is characterized by a zero slope in such a diagram, while a chaotic time series is associated with a negative slope (Syriopoulos & Leontitsis, 2000). This diagram is shown in Figure 4.

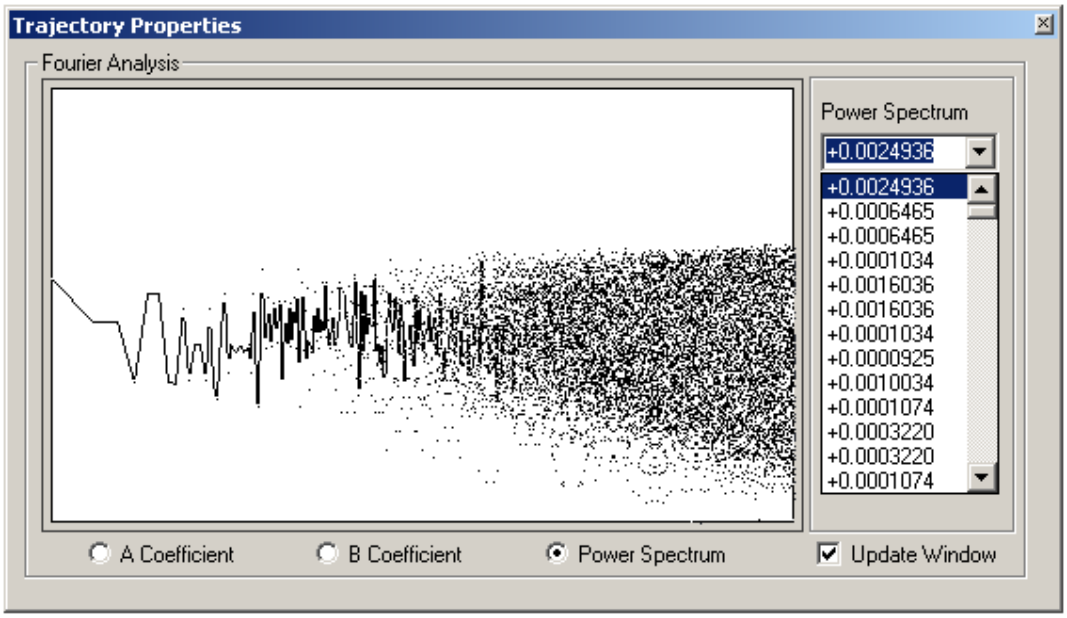

Figure 4. Trajectory Properties - the [logP(f)-log(f)] diagram

The graph type the user wants to display - namely, a (f,α_f), (f,β_f) or (f,P(f)) diagram - is selected by using the appropriate radio button. Regarding the values of the Fourier coefficients, they are calculated via the equations

$$a_f = \frac{2}{N}\sum_{i=1}^{N} x_n \cos\left[\frac{2kf(n-1)}{N}\right] \tag{23}$$

$$\beta_f = \frac{2}{N}\sum_{i=1}^{N} x_n \sin\left[\frac{2kf(n-1)}{N}\right] \tag{24}$$

Finally, the corresponding value of the power spectrum for each frequency f, is calculated as

$$P(f) = \sqrt{a_f^2 + \beta_f^2} \tag{25}$$

For each diagram type, the data values that are plotted against the sample number are shown in the combo box located to the right side of the dialog. The check box entitled *Update Window* allows the update of the window contents in real time, as the user moves the mouse cursor in the *Bifurcation Diagram* dialog box.

The program features described in the previous paragraphs are associated with the 1-D trajectory plotting. However, the system trajectory can be plotted in a 2-D fashion in a diagram of the form $(\upsilon_{n+1},\upsilon_n)$ if the user checks the *2-D* check box in the *Bifurcation Diagram* window. It is not difficult to understand, that the curve drawn in this way is in fact the chaotic attractor of the system whose shape is completely determined by the values of the system parameters. The contents of this window are updated in real time as the user moves the mouse cursor in the *Bifurcation Diagram* window. In this case, the controls in the right panel of the windows are all disabled. If the user does not want to update the window contents, these controls are enabled allowing the user to explore the time evolution of the system by drawing the so-called cobweb plot. This plotting is performed by a child application thread in an animated fashion with the time evolution step to be determined by the user. At any moment, the user can press the *Stop* button to temporarily stop the animation, or the *Finish* button to terminate the procedure - in the last case, the time delay of the drawing operation is set to zero and the procedure is completed in a few seconds. A typical shape of the chaotic attractor of the 1-D recurrent neuron system as well as the progress of the cobweb plotting is shown in Figure 5.

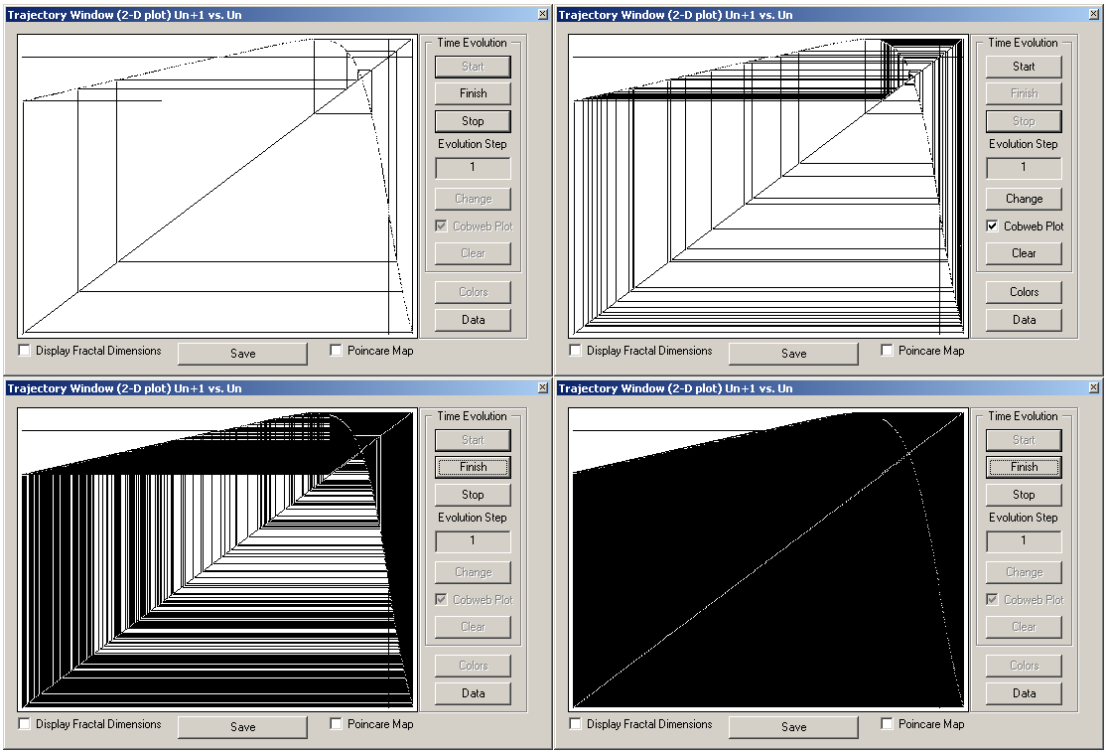

Figure 5. The cobweb plot

4.4 Estimation and Plotting of Fractal Dimensions

One of the most important algorithms of the application is the estimation and plotting of the fractal dimension of the chaotic system under consideration. This dimension is calculated directly from the experimental data, by applying the algorithm of Liebovich and Toth (Liebovich & Toth, 1989) that is composed by the following steps:

1) The trajectory of the system is reconstructed in the proper embedding space with the appropriate values of the embedding dimension d and the time delay τ.

2) For each point of the reconstructed trajectory, its coordinates are normalized in the interval $(0,2^{k-1})$ - where k is a parameter of the algorithm - and their integer part is converted to binary form with k bits length.

3) The space occupied by the attractor is covered with hyper-cubes of length m. Then, a binary number M of k bits length with k-m 1's followed by m 0's is defined.

4) For each coordinate x_i of each trajectory point, the binary number $y_i = x_i$ AND M (i=1, 2, 3,...., d) of k bits length is defined where AND is the logical conjunction operator. In the next step, a new number z is defined from the concatenation of the bits of the y_i values. In this way, the trajectory points that belong to the same hypercube will have the same value for the z parameter.

5) The above procedure is repeated for each trajectory point, the sequence of z values produced in this way is sorted, and the number of different z values is counted. The number of those values represents the number of the hyper-cubes used to cover the space of the attractor and therefore it is a measure of the dimension of the attractor in the phase space.

The algorithm of Liebovich and Toth has been implemented by Sarraille and Difalco (Sarraille & Difalco, 1992); the developed software is named FD3 and it is used extensively for experimental calculation of fractal dimensions for various dynamical systems. The FD3 application is an improved version of the Liebovich and Toth algorithm since it has the ability to calculate simultaneously the values of all the three different dimension types, namely, the capacity, the information and the correlation dimension. These dimensions are treated as special cases of a generalized dimension D(q)=I(q,ε)/log(ε) where

$$I(q,\varepsilon) = \frac{1}{1-q} log\left(\sum_{m=1}^{N(\varepsilon)} p(m,\varepsilon)^q \right) \tag{26}$$

In the above equation, the quantity p(i,ε) describes the percentage of attractor points located inside the i_{th} hypercube of side size ε, while, N(ε) is the minimum number of hyper-cubes required to cover completely the surface of the attractor. Based to the definition of the generalized dimension D(q), one can proves that the capacity, information and correlation dimension can be derived from the generalized dimension for the values q=0, q=1 and q=2 respectively.

In FD3 application, the k parameter of Liebovich and Toth algorithm has the value k=32 and therefore the coordinates of the reconstructed point are normalized in the interval $(0,2^{32}-1)=(0,4294967295)$. In the next step the counting of the hyper-cubes of side size $\varepsilon=2^m$ required for the complete coverage of the attractor surfaces, is performed for 32 different values of m ($0\leq m\leq 31$). For each such value, the following parameters are calculated:

1) The current value of $m=\log_2(\varepsilon)$, the number N(ε) of the hyper-cubes required for the complete coverage of the attractor (the estimation of this number is based to the algorithm of Liebovich and Toth), and the logarithm log(N(ε)) of that number.

2) The sum

$$I(\varepsilon)=\sum_{i=1}^{N(\varepsilon)} z_i \log(z_i) \tag{27}$$

where z_i is the number of attractor points inside the i_{th} hypercube of the phase space. This sum is estimated, since it is required by the information dimension calculation algorithm.

3) The sum

$$F(\varepsilon)=\sum_{i=1}^{N(\varepsilon)} z_i^2 \tag{28}$$

where the z_i parameter has been described above. This sum is estimated since its negative logarithm is required in the correlation dimension calculation algorithm.

After the calculation of the above quantities for each value of m in the interval [0,31], the approximate estimation of the three dimension values is performed for each space scale; in a more detailed description, the values of the capacity, information and correlation dimensions are approximated by the quantities $D_{cap} \approx \log(N(\varepsilon))-\backslash\log(N(2\varepsilon))$, $D_{inf} \approx I(\varepsilon)-I(2\varepsilon)$, $D_{cor} \approx F(2\varepsilon)-F(\varepsilon)$.

These approximate values are estimated for each value of the m parameter, and then, the final dimension values (over all the time scales) are estimated as the slope of the associated least squares line calculated by the previous values. It is important to notice, that the least squares procedure does not use all the m values but only those in the interval [22,29]. The selection of only those values is based on the fact that the values $0\leq m<22$ lead to saturation effects, while the values $m>29$ are characterized by poor resolution leading thus to inaccurate values for the capacity, information and correlation dimensions.

The estimation of these three dimension types is based to the use of the previously described algorithm (the FD3 freely available code has been embedded to the application). These dimensions are estimated not only for a specific trajectory, but also for the trajectories associated with each point of the bifurcation diagram currently in use. This calculation - as well as the drawing of the estimated values - is the main function of the *Fractal Dimension Plotting* child window that is enabled when the user checks the *Fr* check box in the *Bifurcation Diagram* window. This dialog box is shown in Figure 6.

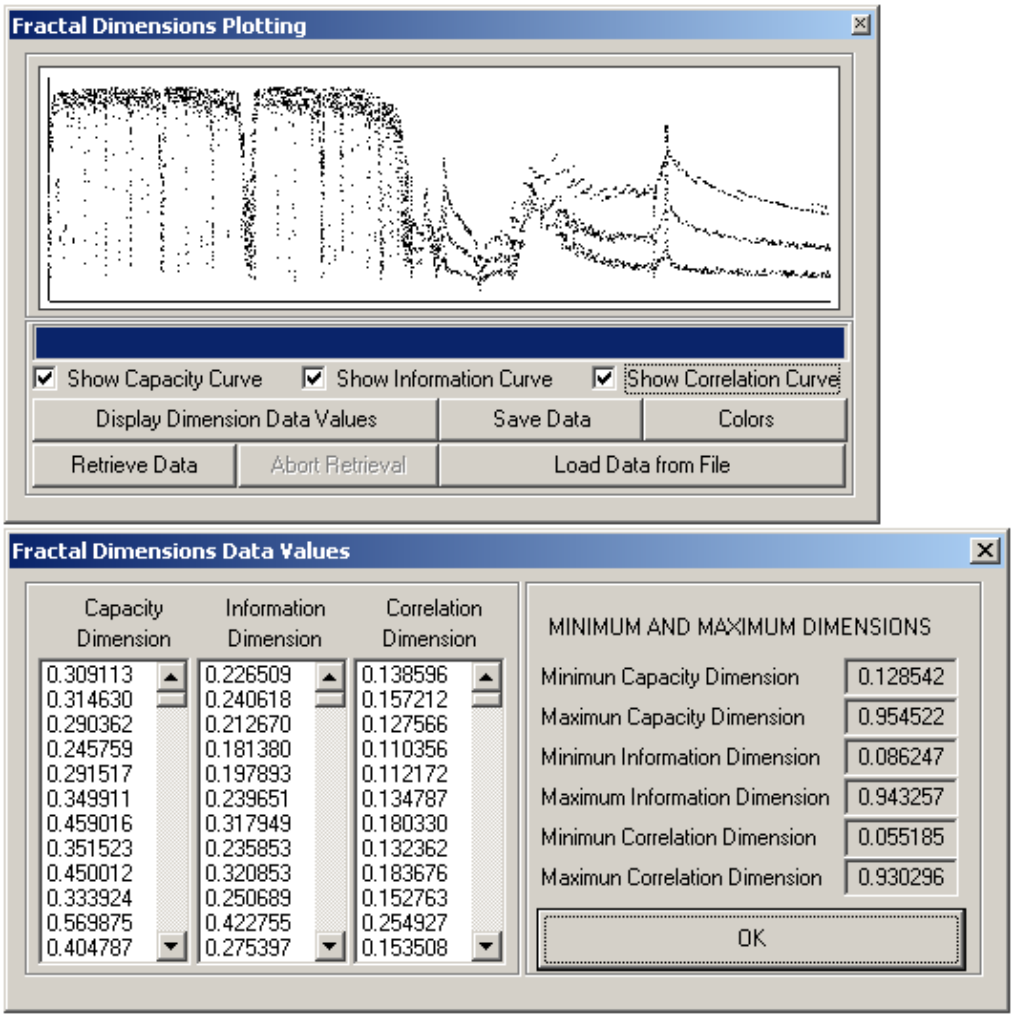

Figure 6. Estimation and plotting of the fractal dimensions

The calculation of the three dimension values for each point of the bifurcation diagram is performed if the user presses the *Retrieve Data* push button. This button initializes a child thread to perform the operation, whose duration is large enough for dense bifurcation diagrams. If the user wants to abort the procedure the *Abort Retrieval* push button can be used to kill the thread and terminate its operation. The user can save the estimated dimension values to a data file and loads them later for further processing. These data values can also be previewed by the user by clicking the button *Display Dimension Data Values*. In this case, these values are shown in the three list boxes of the second dialog box of Figure 6, together with its minimum and maximum values for each one of them.

4.5 Estimation and Plotting of the Lyapunov Exponent

The *Lyapunov Exponent* child window displays the Lyapunov exponent of each trajectory associated with the current bifurcation diagram. This exponent plays a very important role to the study and the characterization of a dynamical system, since a positive Lyapunov exponent indicates a chaotic behavior. In this simple one-dimensional case, the Lyapunov exponent is estimated very easily by the algorithm described by Sprott (Sprott, 2010), but in other cases when the system study requires a phase space reconstruction and a higher dimension value the Lyapunov exponent may be calculated by other methods such as the Wolf's algorithm (Wolf, Swift, Swinney, & Vastano, 1985).

The estimation of the Lyapunov exponent by using the Sprott's algorithm is based to the following procedure:

1) Start with any initial condition in the basin of attraction, or even better, with a point known to be on the attractor.

2) Iterate a few hundred times until the orbit is in the attractor.

3) Select a nearby point separated by a distance d_0 from the current trajectory point.

4) Advance both orbits by one iteration and calculate the new separation d_1 between the new points.

5) Evaluate the quantity $\log(|d_1/d_0|)$ in any convenient basis. Readjust one orbit so its separation is d_0 in the same direction of d_1.

6) Repeat steps 4 to 6 as many times as it is necessary and calculate the average of step 5.

Since the Lyapunov exponent is a single scalar value for each point of the bifurcation diagram, the real time update of the Lyapunov window as the mouse cursor is moved inside the diagram, does not make sense. However the user can enable the appearance of a vertical indicator which is moved accordingly in the *Lyapunov* window indicating the exponent corresponding to the current bifurcation diagram point. The value of the associated Lyapunov exponent is also printed in the bottom area of the window. The user can also save the values of the Lyapunov exponent to a data file for further processing by using the *Save Data* push button.

In the case of unknown time series embedded in geometrical spaces with arbitrary values for the embedding dimension d and the time delay τ, this simplified procedure cannot be applied; in this case, the estimation of the Lyapunov exponent is performed by using a lot of different algorithms (Bremen, Udwadia, & Proskurowski, 1997; Brown, Bryant, & Abarbanel, 1991; Bryant, Brown, & Abarbanel, 1990; Chistianssen & Rugh, 1997; Diakonos, Pingel, & Schmelcher, 2000; Hegger, 1999; Oiwa & Fielder, 1998; Oiwa & Fielder, 2002; Pyragas, 1997; Rosenstein, Collins, & de Luca, 1993; Sano & Sawada, 1985; Wright, 1984) each one of them has its own advantages and disadvantages and its own speed and accuracy. In this project this task is performed by the Wolf's algorithm (Wolf, Swift, Swinney, & Vastano, 1985), that allows the estimation of the largest positive Lyapunov exponent of an unknown time series; the main idea behind this algorithm, is the selection of a trajectory point, the identification of its nearest spatial neighbor and the measurement of their distance as they evolve in time by t_e (this parameter is known as evolution time). If this distance exceeds a predefined maximum value, one of the points is substituted by some other point; otherwise the points continue to evolve without changes. If this fiducial trajectory has traversed the entire data file, the maximum positive Lyapunov exponent is estimated by the equation

$$\lambda_1 = \frac{1}{t_M - t_0} \sum_{k=1}^{M} log_2 \frac{L'(t_k)}{L(t_{k-1})} \tag{29}$$

where M is the total number of replacement steps, while L and L' are the distances between the two points before and after the time evolution.

The estimation of the maximum Lyapunov exponent by the Wolf's algorithm is based to the construction of a database; this database keeps for each hypercube of a grid defined in the d-dimensional embedding space with L hypercubes per dimension (the so-called grid analysis), the number of time series points located on that cube (the situation is identical with the one described in the Liebovich-Toth algorithm section). In this way, the fixed time evolution algorithm that estimates the Lyapunov exponent does not use for each point the whole set of the remaining trajectory points, but only those points located on the same hypercube, leading thus, to a very high execution speed.

Since the percentage of hypercubes with one or more trajectory points is in general very small compared with the total number of cubes inside the grid, a virtual array is defined and the cubes are inserted to it, in the order they appear. However, during calculation, these hypercubes have to be traversed in ascending order with respect to their grid coordinates. For this reason a logical hypercube organization is defined via a second table in which the hypercubes are ordered in an ascending manner, while, their physical organization defined by the order they appear, may be anyone. The location of the last time series data entered to each cube is stored into a third buffer which, in conjunction with a forth one, allows the retrieval of the set of points associated with a hypercube. The structure of the database used by the Wolf's algorithm is shown in Figure 7 (Margaris, Kofidis, & Roumeliotis, 2009).

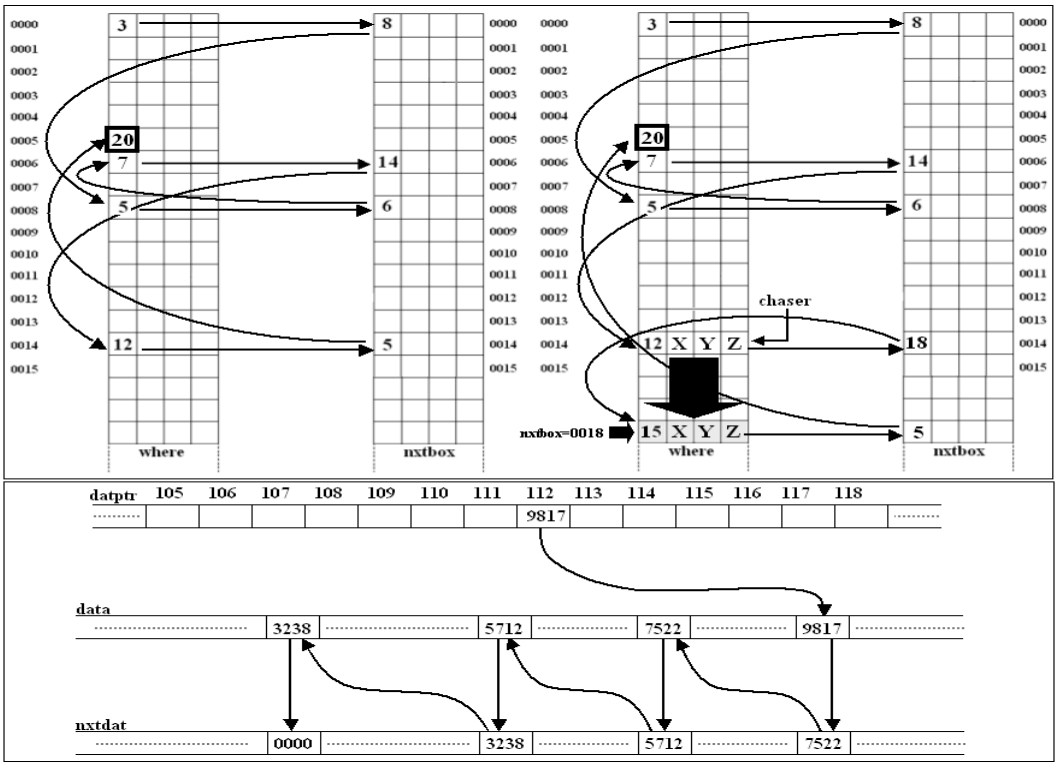

Figure 7. The database structure of the Wolf's algorithm: the *nxtbox* table identifies the logical successor of the corresponding hypercube of the *where* table, while the buffers *nxtdat* and *datptr* allow the retrieval of the *data* time series points associated with a specific cube

The estimation of the Lyapunov exponent includes the following steps:

(Step A) A pair of points z[m] and z[n] is selected and the reconstructed vectors

$$\mathbf{X}_m = \{z[m], z[m+\tau], z[m+2\tau], ..., z[m+(d-1)\tau]\} \text{ and } \mathbf{X}_n = \{z[n], z[n+\tau], z[n+2\tau], ..., z[n+(d-1)\tau]\}$$

are formed. In the beginning of the algorithm, this pair is composed by the first trajectory points and its closest spatial neighbor (from the view point of the reconstructed vectors in the phase space) but during algorithm execution this pair is moved forward to traverse the whole trajectory.

(Step B) The database is searched to find a new better point that is going to substitute one of the points of the initial pair. A trajectory point z[k] is characterized as acceptable and candidate to substitute another point if it satisfies the following conditions:

- The separation of the two points in the time series is greater than or equal to the evolution time t_e, namely, $|k-m| \geq t_e$.
- The separation of points k and h satisfies the inequality $|k-h| \geq 2t_e$ where $h=N-(d-1)\tau-t_e$ is the last time series point processed by the algorithm.
- The Euclidean distance $|\mathbf{X}_m-\mathbf{X}_k|$, where $\mathbf{X}_k=\{z[k], z[k+\tau], z[k+2\tau], ..., z[k+(d-1)\tau]\}$ is greater than a predefined minimum distance D_{min} and less than its previous value D'_{min}.
- The angle θ between the vectors $\mathbf{X}_m-\mathbf{X}_n$ and $\mathbf{X}_m-\mathbf{X}_k$ in the d-dimensional embedding space satisfies the inequality $\theta<\theta_{max}$ where θ_{max} is a predefined maximum orientation error.

The search procedure and the examination of the validity of the above conditions, is performed for a set of points that belong to a certain region of the embedding space. This region is defined by the hypercube

$$C_m = \{c_m^1, c_m^2, ..., c_m^d\} = \left\langle L\frac{z[m]-z_{min}}{z_{max}-z_{min}}, L\frac{z[m+\tau]-z_{min}}{z_{max}-z_{min}}, ...L\frac{z[m+(d-1)\tau]-z_{min}}{z_{max}-z_{min}} \right\rangle \quad (30)$$

(where z_{max} and z_{min} is the maximum and the minimum time series values and L the grid resolution) that contains the vector $\mathbf{X}_m$ and all the adjacent hypercubes in all dimensions that are R hypercubes away from it. The value of R is a function of the hypercube side size $\varepsilon=(z_{max}-z_{min})/L$ and a predefined maximum distance D_{max}. There are $(2R+1)^d$ hypercubes in this region to be examined, and for each one of them, the following procedures are applied:

1) The hypercube coordinates $C_l=\{c_l^1,c_l^2,\cdots,c_l^d\}$ are estimated. If there are coordinates c_i^j such that $c_i^j>1$ or $c_i^j>L$ (j=1,2,....,d) the hypercube is rejected.

2) The entry T of the *where* table that keeps the coordinates of the current hypercube is identified. The hypercube is ignored if it does not exist in the application database.

3) The value *datptr[T]* (see Figure 7) is retrieved, to allow the extraction of all the time series points associated with the current hypercube. For each one of those points, the following operations are performed:

- If the relation $|k-m|<t_e$ holds, the point is rejected since it violates the first condition.
- If the relation $|k-h|<2t_e$ holds, the point is rejected since it violates the second condition.
- The reconstructed vector $\mathbf{X}_k=\{z[k], z[k+\tau],...,z[k+(d-1)\tau]\}$ is formed.
- The Euclidean distance

$$D_m^k=\sqrt{\sum_{i=1}^{d}(X_m^i-X_k^i)^2} \tag{31}$$

is estimated. If this distance satisfies the inequality $D_m^k<D_{min}$ or $D_m^k\geq D'_{min}$ where D'_{min} is the previous value of the D_{min} the point is rejected since it violates the third condition.

- The value of the quantities $\Gamma=\sum_{i=1}^{d}(X_m^i-X_k^i)(X_m^i-X_n^i)$ and $\lambda=\cos(\theta)$ where

$$\lambda=\left|\frac{\sum_{i=1}^{d}(X_m^i-X_k^i)(X_m^i-X_n^i)}{\sqrt{\sum_{i=1}^{d}(X_m^i-X_k^i)^2}\sqrt{\sum_{i=1}^{d}(X_m^i-X_n^i)^2}}\right| \tag{32}$$

are estimated. If $\lambda>1$, the assignment $\lambda=\cos(\theta)=1$ is performed.

- The orientation error θ is estimated via the equation

$$\vartheta=\frac{180}{\pi}\arccos\lambda\approx 57.295827\cos^{-1}\lambda \tag{33}$$

(in radians). If the equation $\theta\geq\theta'$ holds - where θ' the previous value of θ - the point is rejected since it violates the fourth condition.

(Step C) The search procedure is performed for all points of all hyper-cubes that belong to the search area. After the termination of the search procedure, the position of the identified point is returned back (if such a point is available), otherwise the procedure returns a value of zero.

(Step D) If a new point cannot be found, the search procedure is called again and again with a value two times greater than the previous one, for the allowed maximum separation (in other words we set $D_{max}=2D_{max}$ and try again). When such a point k is finally identified, it is marked as the new point by setting n=k.

(Step E) The two points are evolved in time by setting $m=m+t_e$ and $n=n+t_e$. The reconstructed vectors

$$\mathbf{X}_m=\{z[m],z[m+\tau],z[m+2\tau],....,z[m+(d-1)\tau]\} \text{ and } \mathbf{X}_n=\{z[n],z[n+\tau],z[n+2\tau],....,z[n+(d-1)\tau]\}$$

are formed and their Euclidean distance

$$D_m^n=\|X_m-X_n\|=\sqrt{\sum_{i=1}^{d}(X_m^i-X_n^i)^2} \tag{34}$$

is estimated. If the condition $D_m^n \le 0.1D_{min}$ holds, the assignment $D_m^n = 0.1D_{min}$ is performed.

(Step F) The number M of the replacements performed so far, is increased by one (M=M+1) and the current value of the Lyapunov exponent is calculated from the equation

$$zLyap = \frac{log(D_m^n / D_{m'}^{n'})}{tEval \cdot M \cdot t_s \cdot log(2.0)} \tag{35}$$

where $D_{m'}^{n'}$ is the old distance between the pair of points, and t_s is the value of the time step parameter.

(Step G) If the condition $D_m^n < D_{max}$ holds, the two points are considered close enough in the embedding space and no replacement is performed. Otherwise, the search operation is performed again to locate the trajectory point used in the next replacement.

(Step H) If the current point position satisfies the condition m>h, the fiducial trajectory has traversed the entire data file, and the maximum positive Lyapunov exponent has been estimated. On the other hand, if the condition n>h holds, the time series is characterized as problematic; in this case we set m=m-t_e and we repeat the procedure from the beginning.

The application dialog box that allows the estimation of the maximum positive Lyapunov exponent of an unknown time series for arbitrary values of the embedding dimension d and the time delay τ is shown in Figure 8. The application allows the time series data plotting, the preview of its values in the right list box, the generation of the database and the preview of its contains in the two left list boxes as well as the determination of the algorithm parameters (namely the embedding dimension d, the time delay t, the grid resolution L, the minimum separation D_{min}, the maximum separation D_{max}, the maximum orientation error θ_{max}, the time step t_s and the evolution time t_e) by using the corresponding controls in the top of the window. Regarding the main frame of the window, it plots during execution, the projection of the trajectories associated with the current pair of data points (z[m], z[n]) in the 2-D space. In this way, the user can assess in a graphical way, if the evolution of the system is the desired one. The algorithm is executed in a multi-threaded fashion and the user can interrupt the process at any time, to repeat it with different parameter values. Another interesting feature of the application is the stability analysis module that allows the automatic calculation of the Lyapunov exponent for various parameter combinations in order to study its variation with respect to them.

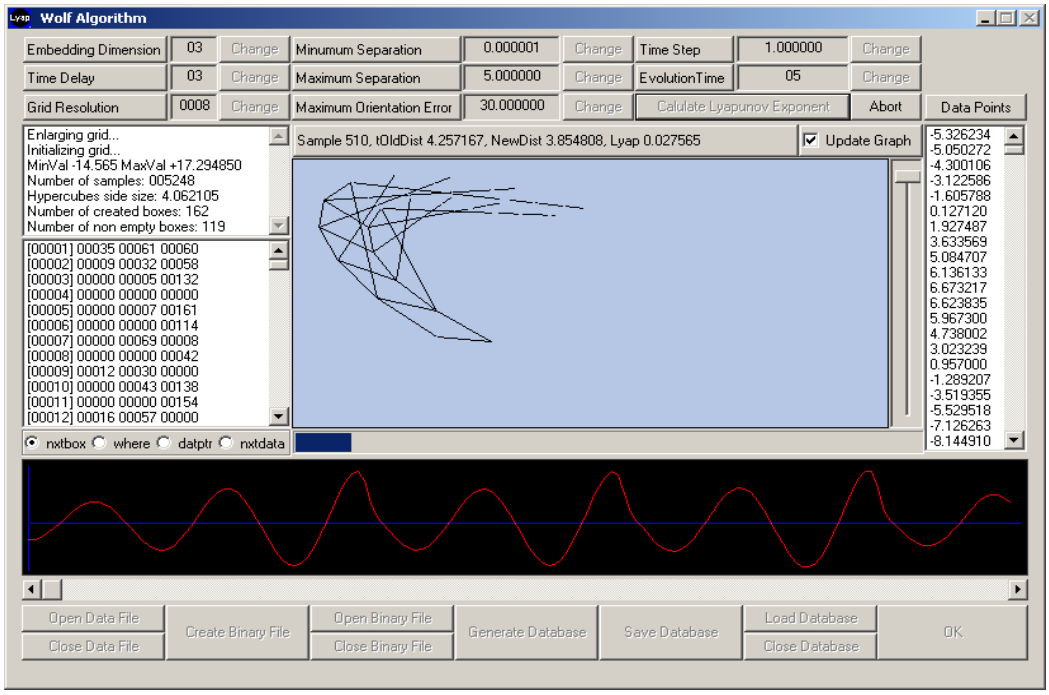

Figure 8. Estimation of the Lyapunov exponent using the Wolf's algorithm

4.6 Creation and Preview of Parameter Configurations

One of the most fundamentals stages of the chaotic systems study, is the exploration of the various regions of the state space to detect chaotic features and to retrieve information about the system behavior under various circumstances. In the case of the recurrent chaotic neuron, there are five system parameters with any possible value, and therefore, the system can be studied in a 5-D state space. However, since this study is very difficult, a simplified procedure has been adopted: the application displays in the screen the bifurcation diagram with respect to the parameter of interest, and the values of the remaining parameters are tuned to detect chaotic states or bifurcation patterns of interest.

Since the complete description of each bifurcation diagram instance requires the recording of a lot of information (such as the parameter type, the minimum and maximum value of this parameter as well as the values of the remaining four parameters), the *Configuration List* dialog box has been implemented to help this procedure. This dialog box, shown in Figure 9, it is used in conjunction with the bifurcation dialog, and its main function is to store a list of bifurcation diagram configurations. When the user wants to keep the configuration of a bifurcation diagram of interest, the *Add* push button is used to store this configuration to a single linked list of nodes of the appropriate type. The user has the ability to add as many as configuration he/she wants, to delete existing configurations, and to navigate between them. During this navigation, the corresponding bifurcation diagram is drawn in the screen, and the associated values are displayed in the configuration window. The user can also save the configuration list to a data file, for further processing.

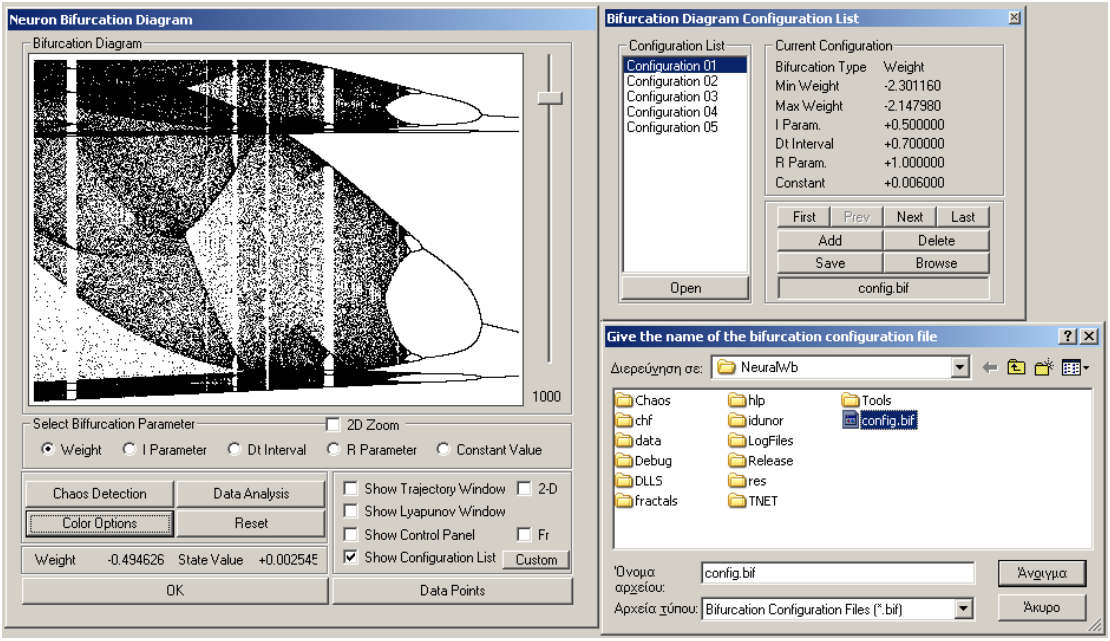

Figure 9. The Configuration List child window

When the user loads a bifurcation diagram from a configuration file, the contents of the *Trajectory*, the *Lyapunov* and the *Fractal Dimension* windows are updated automatically, to reflect the contents of the new loaded bifurcation diagram.

5. Automatic Detection of Chaotic States

One of the most interesting stages of a chaotic system study is the identification of chaotic regions in the parameter space of the system. There are many ways to perform such a task and the most commonly used method is the estimation of the Lyapunov exponent of the system trajectory. At it is well known from the literature, the Lyapunov exponent is a measure of convergence of neighboring trajectories. A positive Lyapunov exponent indicates that these trajectories diverge with time leading thus to a chaotic system behavior.

Since each trajectory of the recurrent neuron system is generated by the recursive equation (22), it is clear that the Lyapunov exponent of the system is a function of the five system parameters. Therefore, the identification of the chaotic regions in the parameter space, can be performed in an experimental way by estimating the Lyapunov

exponent for all possible combinations of the values of those parameters and by characterizing as chaotic states, the values of the parameter vector **p**={w,θ,Δt,R,c} associated with a positive Lyapunov exponent.

The experimental detection of the chaotic states for the recurrent neuron dynamical system is performed using the dialog shown in Figure 10.

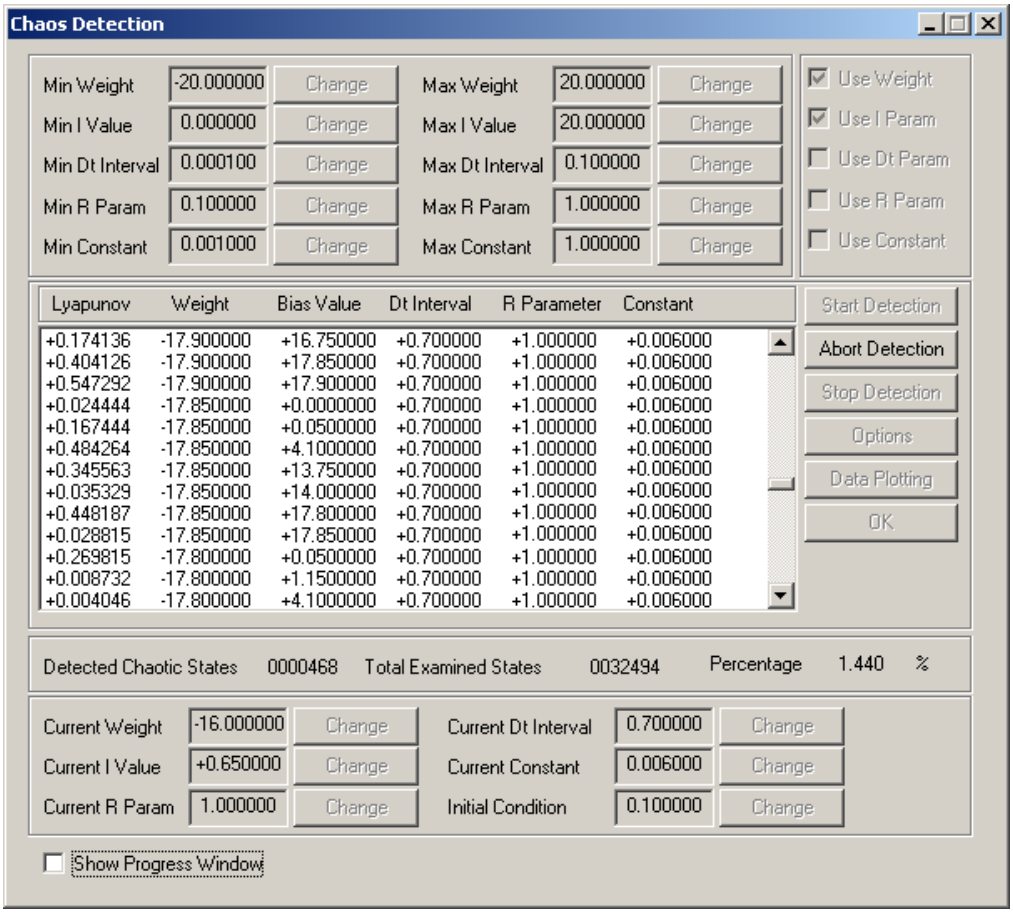

Figure 10. Detection of chaotic states in the recurrent neuron model

In the most general case where all the five parameters are used, the chaos detection procedure is performed by means of a five-level nested loop with each parameter to be varied by its own variation step. However, in most cases the user does not want to perform a complete chaos detection in the 5D state space and the search procedure is performed for certain combinations of the system parameters. A typical example of this case is a chaos detection procedure in which only the weight and the bias parameters are varied between predefined minimum and maximum limits, while the other three parameters are kept fixed. In this case, the chaos detection procedure is restricted to the [weight, bias] parameter space that defines a plane. This situation is displayed in Figure 10, too, where the user has checked only the 'Use Weight' and 'Use Bias' check boxes, leaving the other boxes unchecked.

From the above description it is obvious that in the full chaos detection procedure the total number of examined states is equal to $N_{total}=N_w*N_b*N_t*N_R*N_c$ where

$$N_w=(maxW-minW)/stepW$$

$$N_b=(maxB-minB)/stepB$$

$$N_t=(maxDT-minDT)/stepDT$$

$$N_R=(maxR-minR)/stepR$$

$$N_c=(maxC-minC)/stepC$$

respectively. According to the values of the system parameters used in each case, some of these states are characterized as chaotic (associated with a positive Lyapunov exponent), while the remaining states are marked as non chaotic (i.e. they are converging or periodic). If we denote with $N_{chaotic}$ the number of chaotic states, then the percentage of these states is equal to $P_{chaotic}=(N_{chaotic}/N_{total})*100\%$ while the number of states that are non chaotic is equal to $N_{nonChaotic}=N_{total}-N_{chaotic}$.

For each one of the examined states, the EstimateLyapunovExponent() function is called to calculate the value of the Lyapunov exponent for the current parameter combination by implementing the Sprott's algorithm described above. According to the value of the Lyapunov exponent, the appropriate function is called to add the parameters and the Lyapunov exponent values to the main list box of the *Chaos Detection* window and to increment the associated counter to estimate the new percentage of the chaotic states. The detection procedure and the update of the window contents, is performed by a child thread whose operation can be aborted by the user by pressing the *Abort Detection* push button. A progress window can also be optionally displayed to indicate the progress of the chaos detection operation.

The *Chaos Detection* dialog box presented in Figure 10 shows a detection procedure in progress and for this reason the great majority of the window controls are disabled to prevent the user for changing the parameter values at run time. When the procedure terminates, the user can press the *Data Plotting* push button to view the simulation results in a graphical way. In the current version of the program, the data plotting can handle only 1-D and 2-D data files – with respect to the number of the varied parameters - while for more complicated cases, it can save the data to a text file that uses the GNU Plot data file format. In this way, the user has the ability to generate data plots via this application. A typical screen shot of the *Data Plotting* window is shown in Figure 11.

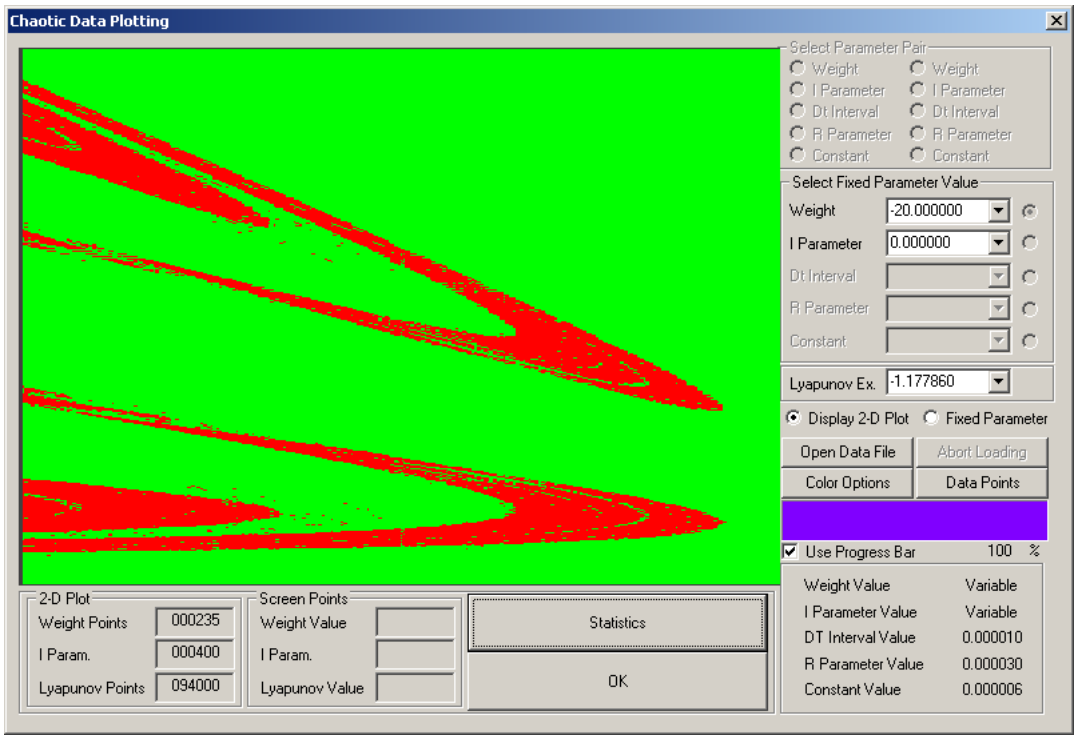

Figure 11. Plotting the results of the chaos detection procedure

In Figure 11, the results of a chaos detection procedure performed in the plane (weight,bias) are shown. The horizontal axis represents the weight parameter and the vertical axis represents the bias parameter. Each one of the points of this diagram corresponds to specific values of the weight and the bias parameter, while the values of the remaining parameters are displayed in the lower right side of the window. The pixel corresponding to a point with coordinates (w_m, i_m) is plotted with red color if the associated Lyapunov exponent is positive (and therefore specifies a chaotic state) and with green color if the associated Lyapunov exponent is negative (and therefore specifies a converging or periodic state). The pattern shown in Figure 11 is identical with the pattern associated with the application of the theoretical model studied by Pasemann (Pasemann, 1993; Pasemann, 1997). In the

lower left corner of the window the number of the weight, bias and Lyapunov points are shown, while, the *Screen Points* control group will display (in a future version) the coordinates of the point associated with the current mouse cursor position.

The data files whose contents are shown in the *Data Plotting* window are loaded by a child thread that is executed if the user presses the *Open Data File* push button and specifies a valid data file via the associated *Browse* Dialog Box. When the file loading operation terminates, the application performs the following tasks: (a) fills the appropriate combo boxes with the values of the selected parameters and the associated Lyapunov exponents (b) initializes the appropriate controls by displaying into them the associated values and (c) plots the graph by using red color for the chaotic states and green color for the non chaotic states. The last operation requires the identification of the maximum and the minimum value of both parameters - in our case of the weight and the bias parameter - and it can take a long time for very large data files.

In the case of 2-D data plotting, the user has the ability to display the 1-D curve associated with a specific value of the parameter of interest. To understand this feature let us view Figure 12. In this figure the user wants to view in a graphical way the values of the weight parameter associated with the bias parameter value θ=2.25. In order to do this, the appropriate combo box has to be expanded and the desired bias value has to be selected. In this case the application identifies the weight values associated with the selected parameter and draws them in the screen in a 1-D fashion. By applying this procedure and by selecting the bias values in the expanded combo box one after the other, the user can see how the weight parameter is varied with the variation of the bias parameter. The same procedure can be applied to preview the variation of the bias parameter with respect to the variation of the weight parameter.

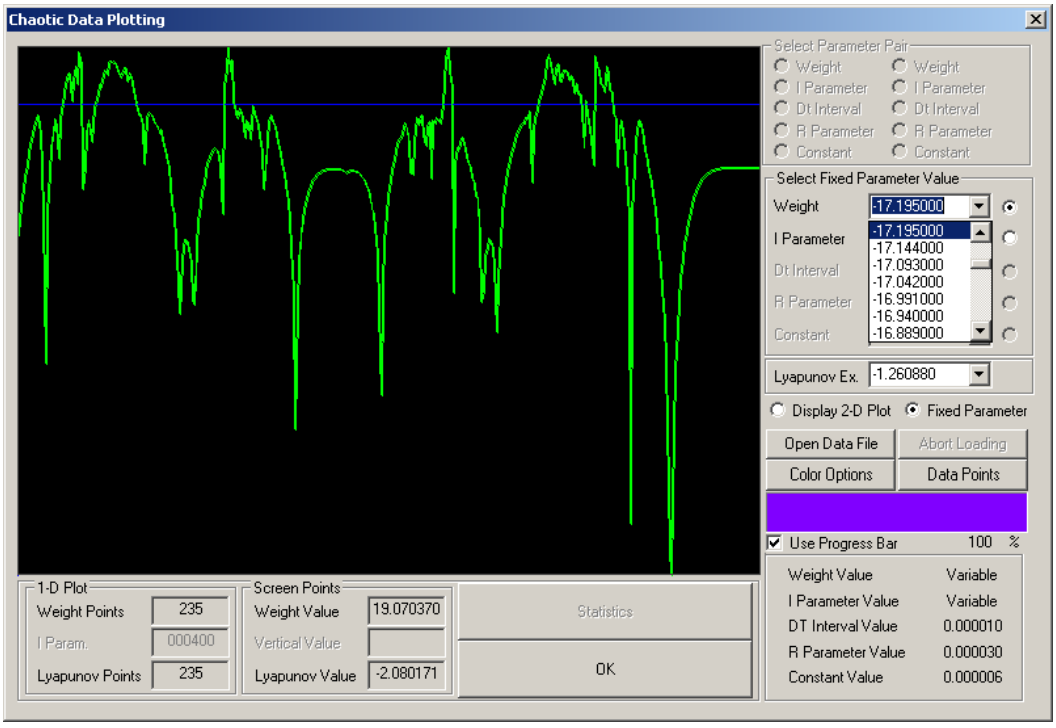

Figure 12. 1-D plot for the two-dimensional chaos detection results

6. Identification of the Basin of Attraction

The basin of attraction of a dynamical system is defined as the set of points in the space of system variables, such that any set of initial conditions that belong to it, dynamically evolve to a particular attractor (Alligood, Sauer, & Yorke, 1996). For the case of a two dimensional system, the set of those points can be identified by applying the two dimensional Newton's root finding numerical method described by the recursive matrix equation (Scheid, 1968)

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} - \begin{bmatrix} j_{11}(x_n, y_n) & j_{12}(x_n, y_n) \\ j_{21}(x_n, y_n) & j_{22}(x_n, y_n) \end{bmatrix}^{-1} \begin{bmatrix} f(x_n, y_n) \\ g(x_n, y_n) \end{bmatrix} \quad (36)$$

where $J=\{j_{ij}(x_n,y_n)\}$ (i,j=1,2) is the Jacobian matrix of the system, calculated on the point (x_n,y_n). In order to apply this method, one has to start from an initial point (x_0,y_0) and iterate the above equation to get a sequence of phase space points (x_i,y_i). The limit of this sequence would be either one of the system fixed points (if fixed points are available) or the infinity (in this special case the initial condition belongs to the basin of infinity).

In order to demonstrate the application of the Newton's method let us identify the basin of attraction of a simple system defined by the equations $f(x,y)=x^3-3xy^2-1$ and $g(x,y)=3x^2y-y^3$. This system has three different roots with values $(x_0,y_0)=(-1/2,\sqrt{3}/2)$, $(x_1,y_1)=(-1/2,-\sqrt{3}/2)$ and $(x_2,y_2)=(1,0)$.

Let us consider now the dialog box of Figure 13 that can be used to identify the basin of attraction in the state space. In the first step the user has to specify the limits of the state space region to which the search procedure is going to be applied, the variation steps in both directions, and the number of iterations of the iterative Newton method for each point (x,y) of the state space. In the next step, the coordinates of the system roots are entered (it is obvious that these roots must be known in advance) and added to the root list of the dialog box (the user can add and delete as many as root he/she wants). As the user enters the roots of the system one after the other, they are drawn in the screen as it is shown in Figure 13 for the case of the above system.

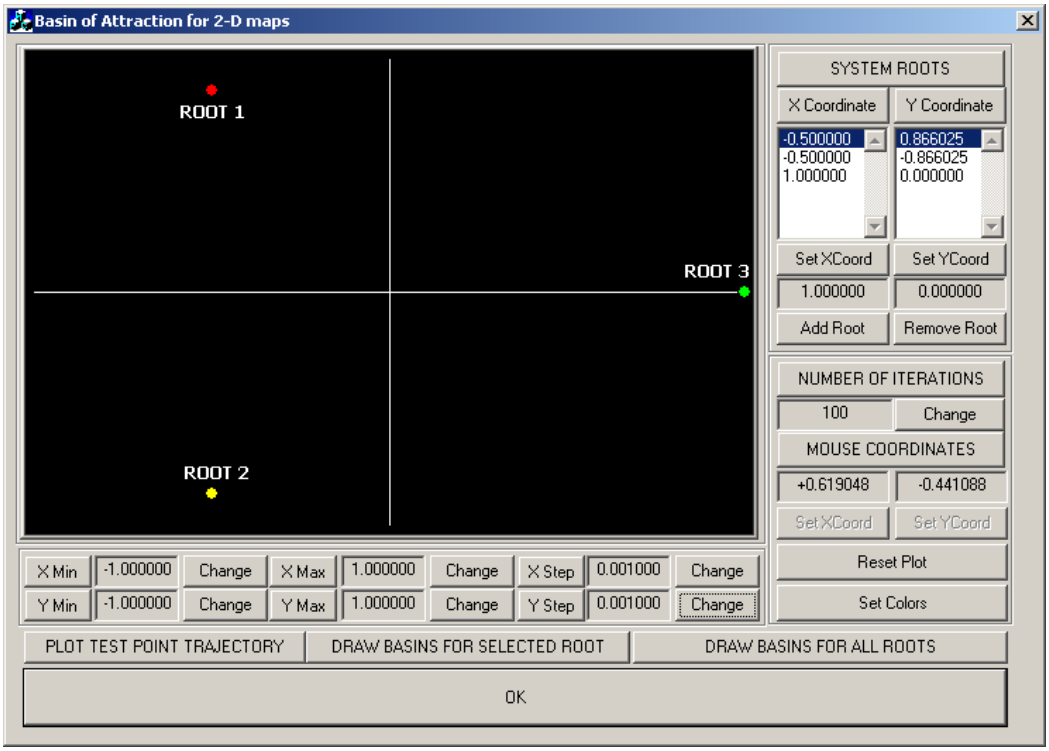

Figure 13. Identifying the roots of the examined system

The program offers to the user three different options: (a) to specify a test point and let the program identify and plot the generated sequence (the limit of this sequence is one of the three system roots or the infinity) (b) to select a root from the root list and let the program to identify and plot the basin of attraction of the selected root and (c) to run the program to identify and plot the basin of attraction for all the system roots. These actions can be performed by using the push buttons shown in the figure. The basin of attraction for all the three roots of the predefined system, are shown in Figure 14 and as it can easily seen, it is a fractal structure.

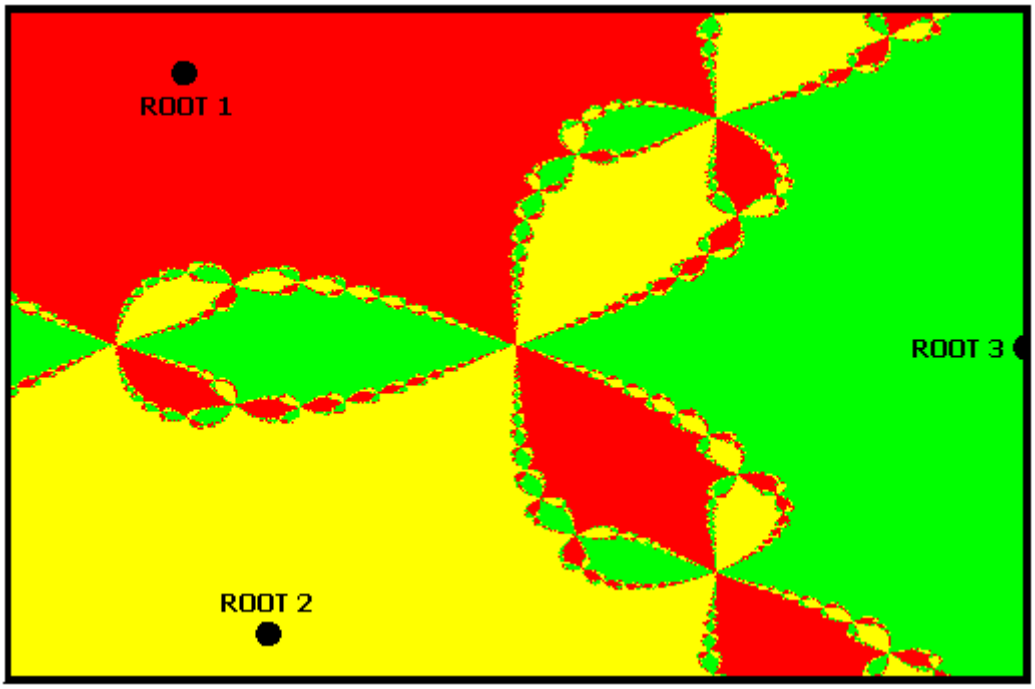

Figure 14. The Basin of Attraction of all the three roots

7. Unstable Periodic Orbits Extraction

Unstable periodic orbits play a very important role to the study and the characterization of chaotic dynamical systems, since they contribute to the determination of a large number of important ergodic properties such as fractal dimensions, Lyapunov exponents and topological entropy; furthermore, these orbits are used for the control and synchronization of chaotic systems. Therefore the detection of those orbits from an experimental time series data has become a central issue in the chaotic systems study.

There are many different techniques for identifying such orbits, and the one implemented here is the algorithm of So et al. (1996; 1997). This technique is based to a transformation of the experimental time series data using information of the local linear dynamics along a trajectory that maps the transformed data in the delay coordinate space in the close neighbor of the periodic orbits. Periodic orbits can then be extracted by looking for peaks in the finite grid approximation to the distribution function of the transformed data.

In order to present the application that allows the extraction of the periodic orbits, let us present the structure of the implemented algorithm. Before applying this algorithm, a phase space reconstruction procedure has to be performed by using the appropriate values for the embedding dimension d and the time delay τ. In this way, each point of the phase space is represented by a vector in the form

$$z(n)=[z_1(n), z_2(n),\ldots..,z_d(n)]=[x(n),x(n-1),\ldots..,x(n-d+1)] \tag{37}$$

After this step, the UPO extraction algorithm can be applied; the main steps of the algorithm are the following:

In order to extract the set of periodic orbits with a period of p, a cluster $Z=(z_1, z_2, z_3,\ldots, z_p)$ of p points has to be formed - each point can be considered as a d-dimensional vector and therefore the number of cluster components is equal to pd - and a transform J has to be applied to it, in order to get the transformed cluster

$$Z'=(z'_1, z'_2, z'_3,\ldots\ldots, z'_p) \tag{38}$$

This transformed cluster has the form $Z'=J(Z,R)=A^{-1}B$ where the matrices A and B are defined as

$$A=\begin{pmatrix} -S(z_1,z_2,R_1) & I & \ldots\ldots & 0 \\ 0 & -S(z_2,z_3,R_2) & \ldots\ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I & 0 & \ldots\ldots & -S(z_p,z_1,R_p) \end{pmatrix} \tag{39}$$

$$B = \begin{pmatrix} F(z_1) - S(z_1, z_2, R_1)z_1 \\ F(z_2) - S(z_2, z_3, R_2)z_2 \\ \vdots \\ F(z_p) - S(z_p, z_1, R_p)z_p \end{pmatrix} \tag{40}$$

In the above equations, the transform **S**(**z**, **z**',R) is a square matrix of order d given by the equation

$$S(z, z', R) = \nabla F(z) + R\left[F(z) - z'\right] \tag{41}$$

where R is a tensor quantity of dimensions dxdxd. It is not difficult to note, that the points $\mathbf{z}_i$ and $\mathbf{z}_j$ appeared in the transform $S(z_i, z_j, R)$ are consecutive points of the reconstructed trajectory and therefore they are related via the equation $z_j=F(z_i)$ where F is the underlying d-dimensional nonlinear map.

The quantity $\nabla F(z)$ for each trajectory point **z** can be estimated directly from the experimental data. In order to do this we have to write it in the form

$$\nabla F(z) = \begin{pmatrix} \frac{\partial f(z)}{\partial z_1} & \frac{\partial f(z)}{\partial z_2} & ..... & \frac{\partial f(z)}{\partial z_{d-1}} & \frac{\partial f(z)}{\partial z_d} \\ 1 & 0 & ..... & 0 & 0 \\ 0 & 1 & ..... & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & ..... & 1 & 0 \end{pmatrix} \tag{42}$$

where f(**z**) is a scalar function defined by the equation

$$z(n+1) = \begin{pmatrix} z_1(n+1) \\ z_2(n+1) \\ z_3(n+1) \\ \vdots \\ z_d(n+1) \end{pmatrix} = F[z(n)] = \begin{pmatrix} F_1[z(n)] \\ F_2[z(n)] \\ F_3[z(n)] \\ \vdots \\ F_d[z(n)] \end{pmatrix} = \begin{pmatrix} f[z(n)] \\ z_1(n) \\ z_2(n) \\ \vdots \\ z_{d-1}(n) \end{pmatrix} \tag{43}$$

The numerical estimation of the partial derivatives $\partial f(\mathbf{z})/\partial z_i$ (i=1,2,….,d) can be performed by the least squares method if a set of M neighboring points is selected (in most case these points are temporal neighbors rather than spatial neighbors) and their coordinates are used in the previous equations. In this case, the system of equations

$$\begin{aligned} w_1^1(n+1) - z_1(n+1) &= \nabla f(z(n))[w^1(n) - z(n)] \\ w_1^2(n+1) - z_1(n+1) &= \nabla f(z(n))[w^2(n) - z(n)] \\ w_1^3(n+1) - z_1(n+1) &= \nabla f(z(n))[w^3(n) - z(n)] \\ &..... \\ w_1^d(n+1) - z_1(n+1) &= \nabla f(z(n))[w^d(n) - z(n)] \end{aligned} \tag{44}$$

is constructed that allows the easy estimation of the gradient components.

It can be proven that the transform J leaves unchanged the true periodic trajectories $\mathbf{Z=(Z_1, Z_2, ….., Z_p)}$ and maps any other combination of p points to the neighborhood of these true trajectories. Therefore these trajectories can be easily identified by looking for peaks in the finite grid approximation to the distribution function. According to the theoretical model, it is possible the appearance of spurious peaks due to stationary points of the transform J that are not true periodic points. However the position of those peaks depends on the values of the vector of tensors, **R**, in contrast with the position of the true periodic points which are independent of **R**. Therefore, these spurious peaks can be eliminated by repeating the calculation of the transformed cluster a lot of times and averaging the values of the coordinates of the transformed points. On the other hand, the positions of the true periodic points will remain unchanged.

Generally speaking, the transformation described above has to be applied for each group of p reconstructed

points. However, since this is a very time consuming operation, a simplified procedure is applied. In this procedure, a backbone of p temporally consecutive points is formed and then, for each one of those points its K closest spatial neighbors are identified. In this way, a cluster of p(K+1) trajectory points is formed and the periodic point transform J is applied for all the possible permutations of points $\mathbf{z}_k$ (k=1, 2, …, p) that belong to the cluster - we recall from the theory that the ordering of points does make sense and therefore two groups of the same points but in different ordering are not identical. By sliding the backbone of p consecutive points along the whole data set and performing this operation, we examine in practice the main majority of the combinations of p points, saving thus a large amount of execution time.

The application of the periodic point transform to a set of reconstructed points is shown in Figure 15. This figure shows all the available dialog boxes used by the algorithm - these dialogs, in general perform the following functions:

1) The *Unstable Periodic Orbits Extraction* dialog allows between others, the specification of the period of the UPOs to be extracted as well as the other algorithm parameters such as the number of spatial and temporal neighbors used by the algorithm.

2) The *Transformed Points & Cluster Vector* dialog creates the backbone cluster for each group of p points according to the algorithm described above.

3) The *Permutation Dialog* computes all the possible permutations associated with the current cluster and shows their contents in the main list box of that dialog

4) The *Transformation Dialog* either transforms only the current backbone, or slides the backbone along the trajectory to calculate the periodic transform for all the groups of p points.

5) The *Options* dialog allows the determination of the log file names and the specification of the parameter values associated with the elimination of the spurious peaks. There are three different files updated by the application: the permutation file that keeps the permutations successfully transformed, the transformation file that keep the coordinates of the transformed points estimated by the algorithm and the error file that keeps the permutations that could not be transformed for some reason.

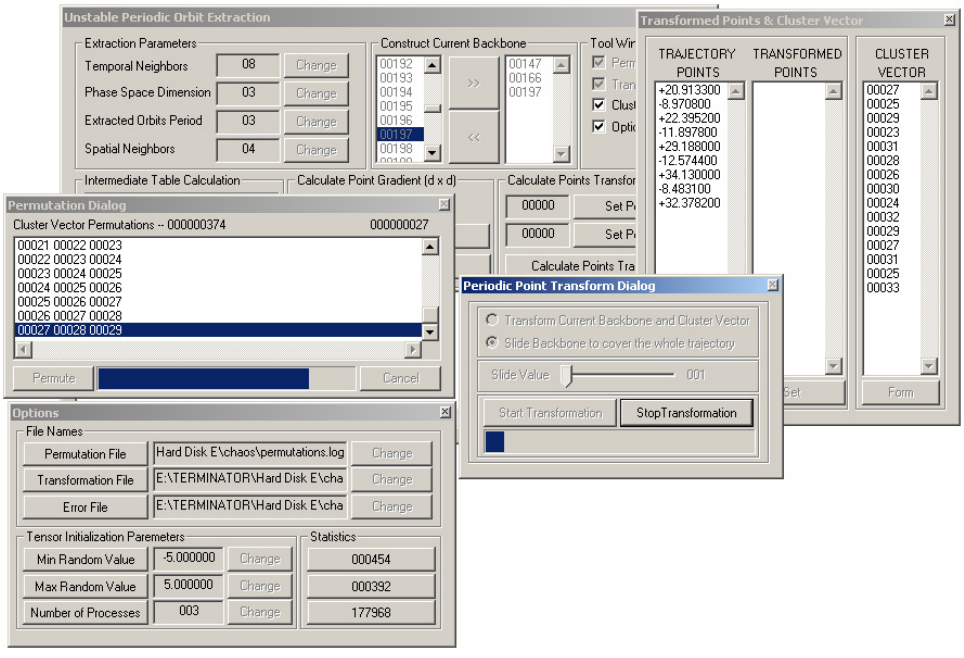

Figure 15. The application of a periodic point transform to a set of reconstructed points

After the application of the periodic point transform the user can preview the coordinates of the extracted points by using the *UPO Extraction Data Points* dialog shown in Figure 16. This dialog box is composed of two list boxes; the first list box contains the successfully transformed permutations and the second list box contains the corresponding coordinates of the transformed points. Since the number of data is generally very large, the data values are shown in pages each one of them contains 1000 lines. The user can move between consecutive pages

by using the navigation arrows; in this case the data are loaded in the memory when the user presses the *Load Current Group* push button.

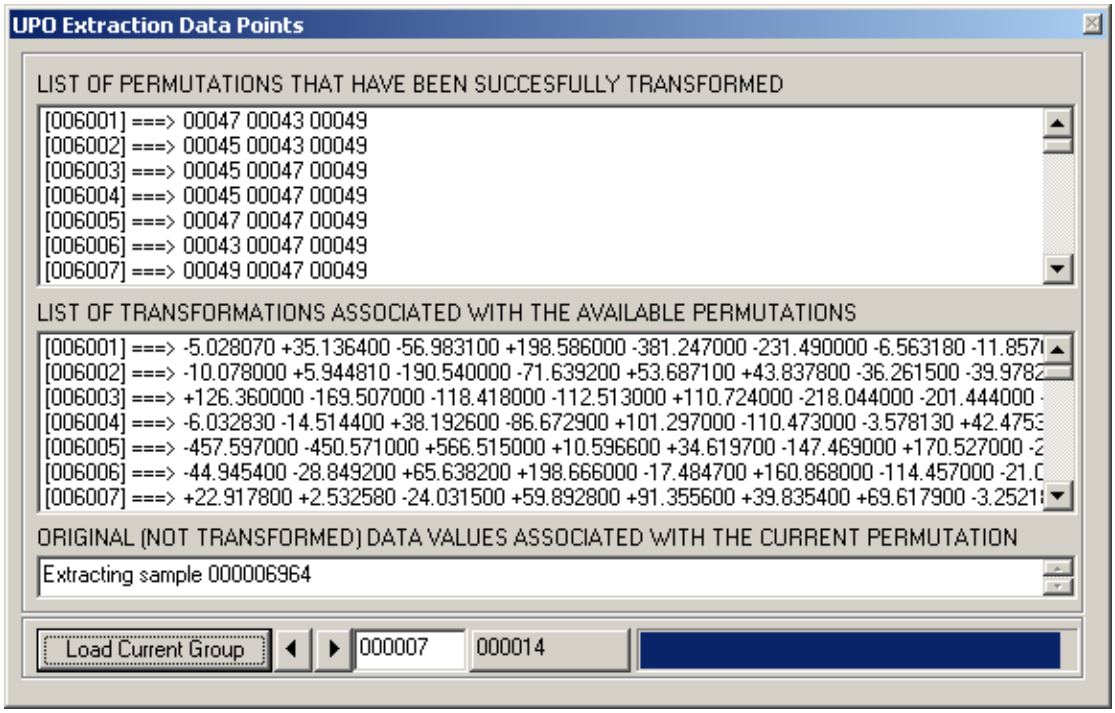

Figure 16. The preview of the permutations and the corresponding transformed points

The development of the dialog box used for the extraction of the periodic orbits is an active part of the research and the processing of the coordinates of the transformed points is currently performed using third party applications as Microsoft Excel. The new functions will perform actions such that the separation of the horizontal axis represent the transformed values into a set of bins of predefined size - the number of bins can be used as an alternate approach - and the counting of the transformed coordinates found in each one of those bins. After the execution of this procedure, the unstable periodic orbits can be extracted from sharp peaks in the histograms produced in this way.

8. Conclusions

The objective of this paper is the presentation of useful tools for the study of chaotic dynamical systems as well as the demonstration of key techniques for implementing similar applications. The current version of those tools include fundamental functions associated with the field of the chaotic dynamics, such as the reconstruction of the trajectory in the embedded space, the generation and study of bifurcation diagrams, the identification of the basin of attraction and the chaotic regions in the parameter space, as well as the extraction of unstable periodic orbits of the chaotic system under consideration. Some of those utilities can be used for the study of any chaotic system, while some others are 'hardwired' to specific examples in order to point out the design issues that have to be used to implement user friendly applications for studying such systems. Future work includes the parameterization of the hardwired features to work with any dynamical system, the optimization of the supported features to provide more capabilities, as well as the implementation of functions that are not currently supported by the application, such as utilities for detecting determinism in a time series and the complete identification of the stable and the unstable manifolds in the parameter space. A challenging aspect is also the parallelization of the developed algorithms using the appropriate tools - such as a message passing library - for building parallel algorithms that can be used to produce the same results in a cluster or a computational grid in a very small processing time.

References

Albano, A., Muench, J., Schwartz, C., Mees, A., & Rapp, P. (1988). Singular value decomposition and the Grassberger Procaccia algorithm. *Physical Review A*, *38*, 3017-3026.

Alligood, K., Sauer, T., & Yorke, J. (1996). *Chaos – An Introduction to Dynamical Systems*. Springer. http://dx.doi.org/10.1063/1.882006

Back, A., Guckenheimer, J., Myers, M., Wicklin, F., & Worfolk, P. (1992). DsTool: Computer Assisted Exploration of Dynamical Systems. *Notices of the American Mathematical Society, 39*(4), 303-309.

Bremen, H., Udwadia, F., & Proskurowski, W. (1997). An Efficient QR Method for the Computation of Lyapunov Exponents. *Physica D, 101*, 1-16. http://dx.doi.org/10.1016/S0167-2789(96)00216-3

Broomhead, D., & King, G. (1986). Extracting Qualitative Dynamics from Experimental Data. *Physica D, 20*, 217-236. http://dx.doi.org/10.1016/0167-2789(86)90031-X

Brown, R., Bryant, P., & Abarbanel, H. (1991). Computing the Lyapunov Spectrun of a Dynamical System from an Observed Time Series. *Physical Review A, 43*(6), 2787-2806. http://dx.doi.org/10.1103/PhysRevA.43.2787

Bryant, P., Brown, R., & Abarbanel, H. (1990). Lyapunov Exponents from Observed Time Series. *Physical Review Letters, 65*(13), 1523-1526. http://dx.doi.org/10.1103/PhysRevLett.65.1523

Chistianssen, F., & Rugh, H. (1997). Computing Lyapunov Spectra with Continuous Gram-Schmidt Orthonormalization. *Nonlinearity, 10*(5), 1063-1072. http://dx.doi.org/10.1088/0951-7715/10/5/004

Dellnitz, M., & Hohmann, A., (1997). A Subdivision Algorithm for the computation of unstable manifolds and global attractors. *Numerical Mathematics, 75*, 293-317.

Diakonos, F., Pingel, D., & Schmelcher, P. (2000). Analyzing Lyapunov Spectra of Chaotic Dynamical Systems. *Physical Review E, 62*(3), 4413-4416. http://dx.doi.org/10.1103/PhysRevE.62.4413

Doedel, E., Champneys, A., Fairgrieve, T., Kuznetsov, Y., Sandstede, B., & Wang, X. (1997). *AUTO97: Continuation and bifurcation software for ordinary differential equations.* Technical Report, Corondia University, 1997.

Engelborghs, K. (2000). *DDE-BIFTOOL: a Matlab package for bifurcation analysis of delay differential equations*. Technical Report, TW-305, Department of Computer Science, K. U. Leuven, Leuven, Belgium.

Feudel, U., & Jansen, W. (1992). CANDY/QA - A Software System for the Qualitative Analysis of Nonlinear Dynamical Systems. *International Journal of Bifurcation and Chaos, 2*(4), 773-794. http://dx.doi.org/10.1142/S0218127492000434

Haykin, S. (1994). *Neural Networks - A comprehensive foundation.* Prentice Hall, ISBN 0-02-352761-7.

Hegger, R., Kantz, H., & Schreiber, T. (1999). Practical Implementation of nonlinear time series methods: The TISEAN package. *Chaos, 9*(2), 413. http://dx.doi.org/10.1063/1.166424

Hegger, R. (1999). Estimating the Lyapunov Spectrum of Time Delay Feedback Systems from Scalar Time Series. *Physical Review E, 60*(2), 1563-1566.

Janaki, T., Rangarajan, G., Habib, S., & Ryne, R. (1999). Computation of the Lyapunov Spectrum for Continuous Time Dynamical System and Discrete Maps. *Physical Review E, 60*(6), 6614-6626.

Kennel, M., Brown, R., & Abarbanel, H. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A, 45*(6), 3403-3411. http://dx.doi.org/10.1103/PhysRevA.45.3403

Kuznetskov, Y., & Levtitin, V. (1997). *CONTENT: A multi-platform environment for continuation and bifurcation analysis of dynamical systems*. Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098, SJ Amsterdam, The Netherlands.

Liebovich, L., & Toth, T. (1989). A fast algorithm to determine fractal dimensions by box counting. *Physics Letters A, 141*(8), 386-390. http://dx.doi.org/10.1016/0375-9601(89)90854-2

Margaris, A., Kofidis, N., & Roumeliotis, M. (2009). A Detailed Study of the Wolf's Algorithm. *International Journal of Computer Mathematics, 86*(7), 1135-1148. http://dx.doi.org/10.1080/00207160701763040

Nusse, H., & Yorke, J. (1997). *Dynamics: Numerical Explorations.* Applied Mathematics Sciences, 101, Springer.

Oiwa, N., & Fielder-Ferrara, N. (1998). A Fast Algorithm for Estimating Lyapunov Exponents from Time Series, *Physics Letters A, 246*, 117-121. http://dx.doi.org/10.1016/S0375-9601(98)00476-9

Oiwa, N., & Fielder-Ferrara, N. (2002). Lyapunov Spectrum from Time Series Using Moving Boxes. *Physical Review E, 65*(3), 1-9.

Pasemann, F. (1993). Dynamics of a Single Model Neuron. *International Journal of Bifurcation and Chaos, 3*(2), 271-278. http://dx.doi.org/10.1142/S0218127493000210

Pasemann, F. (1997). A Simple Chaotic Neuron. *Physica D, 104*, 205-211. http://dx.doi.org/10.1016/S0167-2789(96)00239-4

Pyragas, K. (1997). Conditional Lyapunov Exponents from Time Series. *Physical Review E, 56*(5), 5183-5188. http://dx.doi.org/10.1103/PhysRevE.56.5183

Rosenstein, M., Collins, J. J., & de Luca, C. J. (1993). A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets. *Physica D, 65*, 117-134. http://dx.doi.org/10.1016/0167-2789(93)90009-P

Sano, M., & Sawada, Y. (1985). Measurement of the Lyapunov Spectrum from a Chaotic Time Series. *Physical Review Letters, 55*(10), 1082-1085. http://dx.doi.org/10.1103/PhysRevLett.55.1082

Sarraille, J., & Difalco, P. (1992). *FD3 software for calculating fractal dimensions and related documentation*. CSU, Stanislaus, Computer Science Department.

Scheid, F. (1968). *Numerical Analysis.* Schaum's Outline Series, McGraw-Hill.

So, P., Ott, E., Schiff, S., Kaplan, D., Sauer, T., & Grebogi, C. (1996). Detecting Unstable Periodic Orbits in Chaotic Experimental Data. *Physical Review Letters, 76*(25), 4705-4708. http://dx.doi.org/10.1103/PhysRevLett.76.4705

So, P., Ott, E., Sauer, T., Gluckman, B., Grebogi, C., & Schiff, S. (1997). Extracting Unstable Periodic Orbits from Chaotic Time Series Data. *Physical Review E, 55*(5), 5398-5417. http://dx.doi.org/10.1103/PhysRevE.55.5398

Sprott, J. (2010). *Numerical Calculation of the largest Lyapunov exponent.* in J.Sprott's techical notes, Retrieved from http://sprott.physics.wisc.edu/chaos

Stefansson, A., Concar, N., & Jones, A. (1997). A note on the Gamma Test. *Neural Computing and Applications, 5*, 131-133. http://dx.doi.org/10.1007/BF01413858

Syriopoulos, K., & Leontitsis, A. (2000). *Chaos - Analysis and Prediction of Time Series.* Anikoulas Publications, Greece.

Takens, F. (1981). Detecting Strange Attractors in Turbulence, in Rand, D. & Young L. S. (eds): *Dynamical Systems and Turbulence*, Warwick, 1980, published as Volume 898 of *Lecture Notes in Mathematics*, 366-381 Springer Verlag, Berlin.

Wolf, A., Swift, J., Swinney, H., & Vastano, J. (1985). Determining Lyapunov exponent from a time series. *Physica D, 16*, 285-317. http://dx.doi.org/10.1016/0167-2789(85)90011-9

Wright, J. (1984). Method for Calculating a Lyapunov Exponent. *Physical Review A, 29(*5), 2924-2927. http://dx.doi.org/10.1103/PhysRevA.29.2924