The Effect of Non-linear Dynamic Invariants in Recurrent Neural Networks for Prediction of Electrocardiograms

By
María del Pilar Gómez-Gil

A dissertation in Computer Science
Texas Tech University

Committee:
Dr. William J. B. Oldham (chairperson)
Dr. Donald Gustafson
Dr. Noe López-Benítez
Dr. Sunanda Mitra

TABLE OF CONTENTS

LIST OF TABLES

ABSTRACT

The possibility of automatic and accurate prediction of heart failures from the analysis of electrocardiograms (ECG) could be a breakthrough in medicine, because cardiologists can sometimes identify diseases and foresee catastrophic events, but they are not always successful. However, ECG, as many other biological rhythms, are the result of complex, non-linear dynamical systems, believed by many researches to be chaotic from a mathematical point of view. Chaotic signals are extremely dependent on initial conditions; they look random or noisy, but they are the result of bounded, deterministic systems. Therefore, prediction of ECG is a real challenge.

This research focused on the ambition of finding ways to model and predict electrocardiograms using artificial neural networks. It is known that point-by-point prediction is impossible for chaotic time series. However, we were looking for a predictability that could allow a network to model the attractor associated to ECG, rather than making it able to calculate accurately each value in the future. A prediction with such capabilities could foresee bifurcations in the dynamics and hence, predict catastrophic events.

We explored the use of Lyapunov exponents (an invariant measure of the divergence of several trajectories of a dynamical system) as an aid on the training of predictors based on complex neural networks (CNN). A CNN is a recurrent network built with harmonic generators, which are 3-node recurrent neural networks previously trained to reproduce a specific sine wave. Several predictors were designed training a feed-forward network to reproduce an ECG, using past signal values as external inputs; harmonic generators trained to reproduce the harmonic components of the signal. After that, these weights were embedded in a CNN, which trained until reaching a minimum. All predictors trained using the algorithm back-propagation through time.

We found that, embedding the Lyapunov exponents using the fashion described before is not enough to make the network fully capture the dynamics of the system, but it improved its short-term prediction. Besides, we found that harmonic generators control oscillations of the trajectories in long-term predictions. None of these characteristics are present in feed-forward networks or plain recurrent neural networks.

ACKNOWLEDGEMENTS

I would like to thank the members of my committee, Dr. Sunanda Mitra and Dr. Donald Gustafson for all the support and advice that I received from them during the development of this research. Special thanks are given to my advisor, Dr. Brian Oldham, for his continuous encouragement and friendship during the development of my doctoral studies and especially during this research. Also, I would like to express very special thanks to Dr. Noé Lopez-Benitez for his invaluable support, teaching and friendship. My sincere appreciation is expressed to the rest of the faculty of the department of Computer Science for all their assistance and patience when they had me as a student in their classes.

I would like to recognize the following institutions: "Universidad de las Américas-Puebla," " Consejo Nacional de Ciencia y Tecnología," and the Department of Computer Science of Texas Tech University for the financial support that I received during my doctoral studies.

I wish to express my sincere gratitude to my friends Sandy and Bob Crosier, who helped me during these three years with their invaluable advice, not only as an international student but as a friend. Sandy kindly and patiently proofread this document which I know was not fun. I also whish to express my most sincere thanks to my friends: Sara Pendley, Rafael Cedeño, Gilberto Zamora and Ginnett Rollins who have offered me their friendship and support when needed. I am specially grateful to my parents, Jorge and Lilia, who have always encouraged me to go "one step ahead," and to Dr. Juan Manuel Ramirez, who has for many years been "the wind beneath my wings."

Finally, I would like to dedicate this work to my brothers: Jorge and Roberto†, and their children: Cintya, Jorge and Mariana, as a tribute to their unconditional love.

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Since the seventies, advances in the theory of non-linear dynamics have encouraged the construction of models and predictors of non-linear time series that were previously considered intractable. At the same time, artificial neural networks have been widely used to model dynamic systems in applications of prediction, noise filtering and analysis of temporal sequences such as sunspot series, stock market data, disease behavior and speech signals.

Among other applications, the theory of non-linear dynamics has been recently used to model biological rhythms of the human body, such as blood pressure, heart beats and concentration of sugar in the blood. From a mathematical point of view, many of these rhythms have been found to be chaotic (Glass 1988). Chaotic signals are extremely dependent on initial conditions, and even though they look random or noisy, they are the result of deterministic systems, bounded and aperiodic. Due to these characteristics, chaotic signals are very difficult to model and consequently, to predict.

A biological rhythm of our particular interest is the electrical activity of the human heart. The electrocardiogram (ECG), a measure of this activity, is considered by several authors to be a chaotic signal (Glass 1987, 1991; Denton et al. 1990 (a); Albert 1990). The need to model this dynamical system lies in the fact that, even when experienced cardiologists can identify diseases and foresee some catastrophic events from the behavior of electrocardiograms, they are not always successful in predicting when it is going to occur. Their success depends upon many factors not yet clearly identified. Therefore, the possibility of an automatic and accurate prediction of heart failures from analysis of the ECG could be a breakthrough in medicine. However, it is difficult to model an ECG due to its chaotic characteristics and the high-frequency peaks present in it (points R and S at Figure 1.1).

The study of non-linear oscillations has been developed extensively since the contributions of Alekandrovech Andronov (1901-1952), who recognized observable oscillations using the abstract limit cycles defined by Poincaré (1854-1912). Andronov's main instrument was the two-dimensional phase space, a concept that led to the discovery of chaos. Chaotic signals result in strange but well defined attractors, which are clearly

identified in the phase space embedding the system. A return map is a representation of the phase space of the system (section 2.1). Figure 1.2 shows the return map of an electrocardiogram. A strange attractor with a dense injection region is noticed in the figure. Besides phase spaces and return maps, new concepts have been developed lately in non-linear dynamics: Poincaré sections, Lyapunov exponents, correlation dimension, fractal dimension and Kolmogorov entropy, among others.

One of the most popular tools to identify chaos is the calculation of Lyapunov exponents (section 2.7), which are an invariant quantitative measure of the divergence of several trajectories in a dynamic system. A positive maximum Lyapunov exponent characterizes a system that is very dependent of initial conditions, that is, one that is most probably chaotic. Lyapunov exponents are currently the standard metric used to identify chaos in time series generated by unknown dynamical systems. Several numerical algorithms have been developed for calculating these numbers.

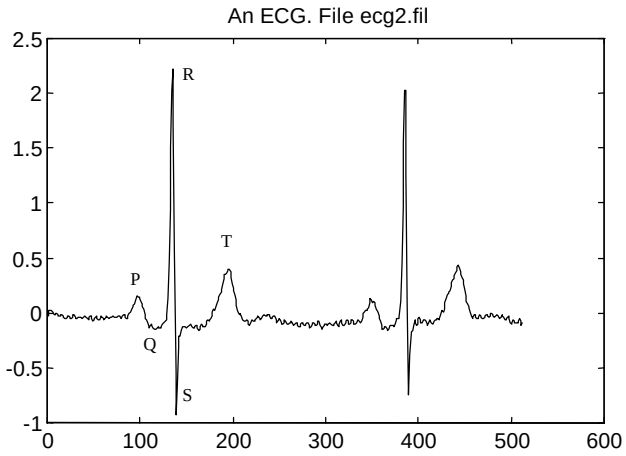


Figure 1.1. Components of an ECG

Artificial recurrent neural networks (RNN) (Figure 1.3) have shown to be promising tools for modeling non-linear time series. Appropriate topologies of RNN are believed to be able to acquire the dynamics imbedded in this kind of data. Due to its internal feedback connections, RNN contain memory, which make them very powerful and suitable for applications where information is coupled with time. Such a recurrence makes RNN to behave as complex non-linear systems, able to extract the invariant characteristics defining a

dynamical system. Therefore, RNN are a promising tool for long-term prediction of chaotic signal, an open problem at this time.

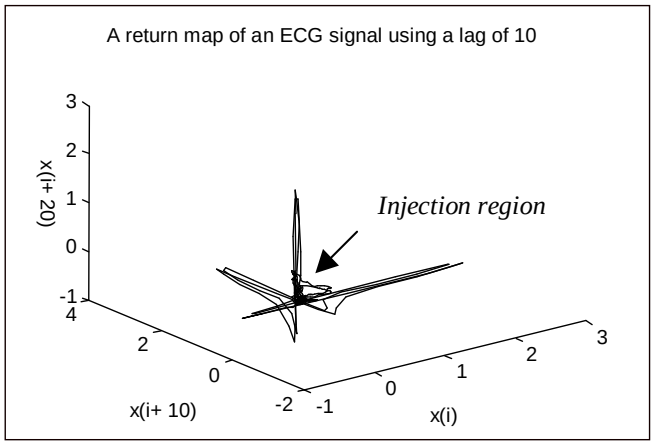


Figure 1.2. A plot of the strange attractor generated by an electrocardiogram.

The ability of an artificial neural network to model a system is completely related to its topology, training algorithm and the data used for teaching it. Therefore, to search for a solution using a neural network implies finding the appropriate topology, using the right training algorithm, with enough and adequate training data. It is clear that, as more information about the characteristic of the problem is embedded in the topology of the network and in the training data, the network will have improved chances for solving the problem.

This research focuses on the ambition of finding ways to model and predict electrocardiograms using both neural networks and the theory of non-linear dynamics. This work is based upon several previous studies in areas of artificial neural networks, non-linear dynamic systems, chaos, forecasting of time series and digital signal processing.

It is known that long-term point-by-point prediction is not possible for chaotic time series, due to its divergence characteristics (Wang and Alkon 1993). However, in this research we are looking for a kind of predictability that could allow a network to model the behavior of the attractor associated to electrocardiograms, rather than making it able to calculate accurately each value in the future of such a signal. A predictor with such

capabilities could foresee bifurcations in the dynamic systems; consequently, many of the catastrophic failures of the heart that kill hundreds each year could be diagnosed.

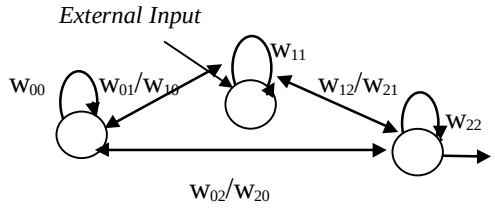


Figure 1.3. A small fully-connected recurrent neural network.

The main objective of this research was to explore the use of information about the non-linear characteristic of the ECG signal as an aid on the training of recurrent neural networks, and to evaluate the prediction abilities of such networks.

With this ultimate goal in mind, we implemented and evaluated several network predictors, considering the concepts of Lyapunov exponents and harmonic generators. We call a harmonic generator a 3-node recurrent neural network that has been trained to produce a sine wave with a specific frequency and amplitude (section 3.3.1.4). Such a trajectory is generated without receiving any external inputs each time that a prediction is performed; the only external information given to the network for point-by-point prediction consists of the initial conditions of the system at time zero (section 3.1). This kind of predictor is called autonomous. A recurrent neural network using these harmonic generators, called the complex network, showed promising results in previous works (Oldham and Gómez 1997).

The Lyapunov exponents were imbedded in the predictors designed in this research by training a feed-forward neural network to generate a signal with Lyapunov exponents similar to the original training signal. After that, the weights of this network were used as initial weights of hybrid or recurrent neural networks constructed over this feed-forward network. Several versions were tested, including the use of external inputs, feedback for prediction, and hybrid networks combined with harmonic generators. In addition, we explored the impact of the use of the techniques of teacher forcing (William and Zipser 1989) and constants weights (Pearlmutter 1990) when training the neural networks using the

algorithm back propagation through time. Also the effect of filtering and decimation of the training signals was analyzed.

The main contribution of this work was the analysis and evaluation of the amalgam of two ideas: the ability of Lyapunov exponents to characterize chaos and the ability of recurrent neural networks to represent complex systems. Such a mixture was done in a search for improving the long-term capabilities of predictors built using complex networks. As a result of this research we found that the embedding of Lyapunov exponents in complex networks, using the version described before, is not enough to allow the network fully capture the dynamic of the chaotic system generating the heart activity, and therefore, still not able to foresee catastrophic events. However, it was found that the embedding of this invariant information does improve the short-term ability to predict, and that the harmonic generators are a powerful tool to control the oscillations of the trajectories in long-term prediction.

Additionally, we found the algorithm back-propagation through time a versatile tool for training networks composed of a mixture of recurrent and feed-forward connections. However, we also discovered a few practical problems in the implementation described by Pearlmutter (1990), particularly with the definition of the integration step required for such implementation. Other interesting results of this work included finding that adjustable time-constants (section 5.1.3) proved to improve the performance of these predictors, and that the application of teacher forcing (section 3.3.1.2) did not generate any improvement in the training of the complex network.

This document is divided as follows: Chapter II presents the basic concepts related to non-linear dynamic systems. It contains the background needed to fully understand the ideas and terminology used during the rest of the document. Chapter III describes the state of the art of recurrent neural networks when applied to chaos and oscillatory systems. It also contains basic definitions related to forecasting, and a description of linear and non-linear predictors. This chapter also includes a detailed description of the recurrent neural network called the complex model, which was used as the starting point of some of the predictors designed in this work. Chapter IV enumerates the methodologies used in this research to evaluate the performance of the predictors, as well as the algorithms applied to pre-process the signals used to train the networks. The performance metrics included the

Mean Square Error, the Lyapunov exponents and visual inspection of return maps. The pre-processing of training signals involved filtering, sampling rate conversion and harmonic decomposition. Chapter V presents the results obtained by nine predictors designed applying the concepts of Lyapunov exponents and harmonic generators in feed-forward, hybrid, and fully-recurrent neural networks. The predictors are grouped as follows: modifications to the original complex model, predictors based on hybrid neural networks, predictors based in feed-forward networks and predictors based in hybrid networks using harmonic generators and external inputs. Chapter VI presents a detailed discussion of our conclusions and some recommendations for future work. Appendix A contains a numerical algorithm for the calculation of Lyapunov exponents from a time series. Appendix B describes the training algorithm back-propagation through time. Appendix C contains the connection matrices defining the topologies of the hybrid neural networks used in the experiments reported in Chapter V. Appendix D describes the electrocardiogram signals used for the experiments reported in this work.

CHAPTER II
NONLINEAR DYNAMICAL SYSTEMS

This chapter summarizes basic concepts of non-linear dynamical systems that are needed to fully understand the context in which this thesis is developed. For a complete review of this topic see: Tong (1993), Parker and Chua (1998), Epstein (1997), Glass and Mackey (1988), Denton (1990) (a).

2.1 General Concepts

A non-linear system may be described by a set of differential equations:

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{F}(\mathbf{y}(t)), \quad \mathbf{y}(t) = [y_1(t), y_2(t), \ldots y_d(t)], \quad \mathbf{y}(0) = \mathbf{y}_0. \tag{2.1}$$

where the vector field **F** is nonlinear. Here d is the dimension of the system.

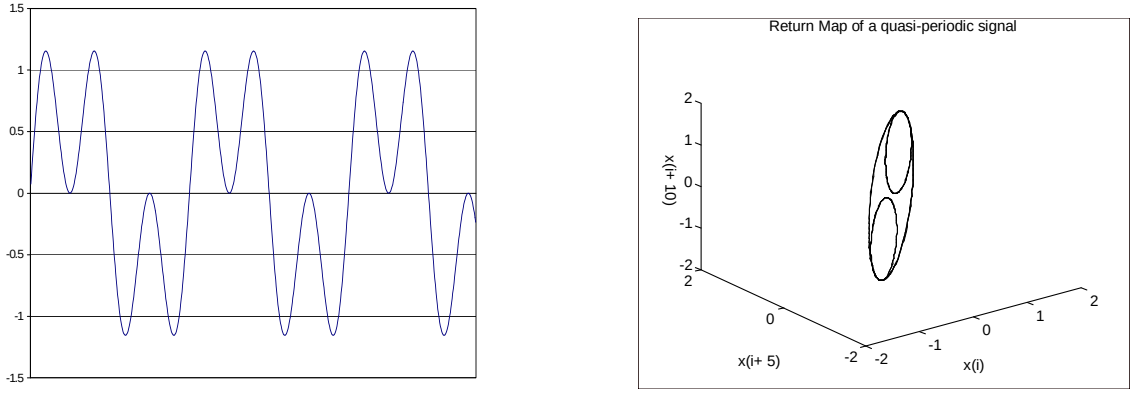
Equation (2.1) describes the motion of a point in a d-dimensional state space known as *phase space*. A *trajectory or orbit* of the system is a curve of $\mathbf{y}(t)$ drawn in the phase space, showing the solution of (2.1) for a specific initial condition. A family of trajectories for different initial conditions is called a *phase portrait*.

The behavior of a dynamical system can be visualized using a plot called *return map*, which is a representation of the state space of the system. A return map shows the relation between a given point in a time series with other points further away in time. Each point is plotted in a different coordinate. The dimension of the plot depends on how many points are being related. The temporal difference between the points being plotted is called the *time lag*. Figure 2.1 (b) shows a return map of the time series plotted at Figure 2.1(a).

2.2 Fixed Points and Limit Cycles

The asymptotic behavior of a dynamical system as $t \to \infty$ is known as its *steady state*. An steady state y* is a set of values of the variables of a system for which the system does not change as time proceeds. That is:

$$\frac{d\mathbf{y}(t)}{dt} = 0. \tag{2.2}$$



(a) (b)

Figure 2.1. The concept of return map. (a) A quasi-periodic time series. (b) Its return map for a time lag = 5

When the steady state is a point, it is called a *fixed point* or *equilibrium point*. For example, consider the system (Greenberg 78):

$$\dot{x} = -ax + y, \qquad \dot{y} = -ax - ay\,. \tag{2.3}$$

Changing to polar form $x = \cos\theta,\ y = \sin\theta$, it reduces to $\dot{r} = -ar,\ \dot{\theta} = -1$ whose trajectories are logarithmic spirals that approach a singular point if a > 0 and depart from it if a < 0. Figure 2.2 shows one of such trajectories for a > 0 in polar coordinates.

Some systems do not reach fixed points as $t \to \infty$, rather they oscillate. The solution to their differential equation is periodic, known as a *limit cycle*. A *limit set* is defined as the set of points in the state space that a trajectory of the system repeatedly visits. The points of the limit cycle form the limit set.

An example of an oscillator is the Duffing equation, defined as:

$$\ddot{x} + \alpha x + \beta x^3 = 0\,. \tag{2.4}$$

The solution of this system is given by the family of curves:

$$y^2 + \alpha x^4 + \frac{\beta x^4}{2} = C\,, \tag{2.5}$$

where C is a constant. Some trajectories of these curves for $\beta > 0$ are shown at Figure 2.3

An *attractor* is a set of points toward which a trajectory approaches after its transient state dies out. Equilibrium points or fixed point and limit cycles are attractors. A *basin of*

attraction is a set of initial conditions for which the system approaches the same attractor as $t \to \infty$.

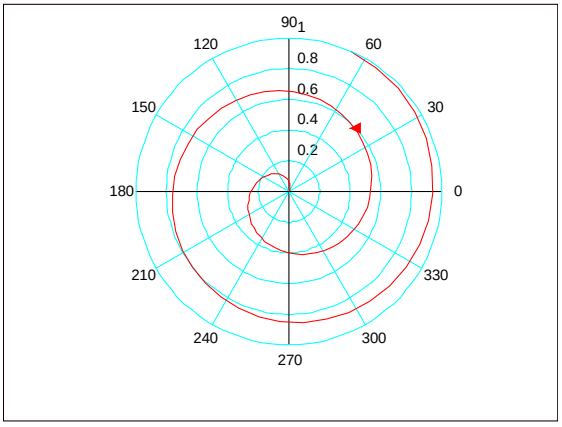


Figure 2.2. A trajectory of equation 2.3 traveling to a fixed point.

For example, the system:

$$\frac{dr}{dt} = ar(1-r) \quad \text{a} > 0$$
$$\frac{d\Phi}{dt} = 2\pi, \tag{2.6}$$

has a limit cycle at $r = 1$. A two-dimensional radial symmetric differential equation like this is called a *Poincaré oscillator*. Any initial condition except $r = 0$ will approach the trajectory to a specific cycle. See Figure 2.4, where two trajectories starting from different initial conditions will approach the circle of radius 1.

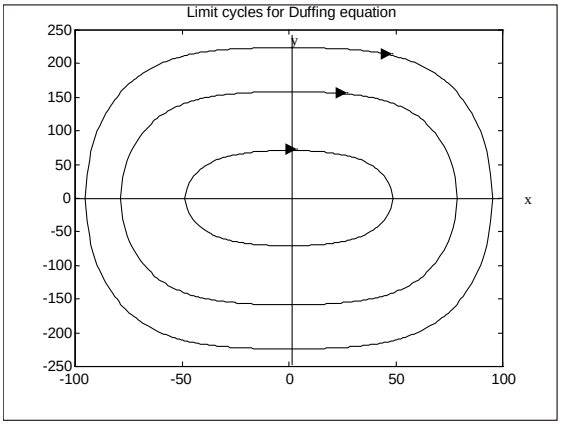


Figure 2.3. Examples of limit cycles for Duffing equation.

Limit cycles are not possible in linear systems or in one-dimensional ordinary differential equations.

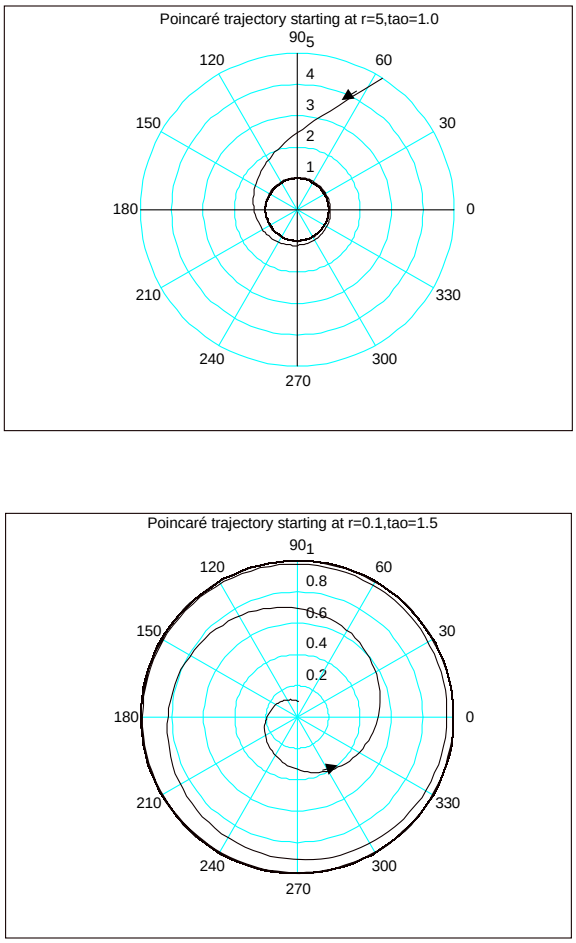


Figure 2.4. Two examples of stability in Poincaré oscillator.

2.3 Chaos

The study of non-linear oscillations has been developed extensively since the contributions of Alekandrovech Andronov (1901-1952), who identified observable self-sustained oscillations using the abstract limit cycles defined by Poincaré (1854-1912) (Tong 1993). Andronov's main tool was the two-dimensional phase plane, a concept that led to the discovery of strange attractors or chaos.

When the steady state behavior of a system is bounded but not an equilibrium point or a limit cycle then the system is said to be *chaotic*. The geometrical object in state space to

which chaotic trajectories are attracted is called a *strange attractor*. The geometry of strange attractors normally is very complicated. Strange attractors possess fractional dimension, that is, a dimension that is not integer. The dimension of an attractor shows a lower bound on the number of state variables needed to describe its steady-state behavior (Parker and Chua 1989). An strange attractor presents one or more *injection regions,* which are zones of the phase space with an very large number of trajectories crossing along them. Such closeness in the trajectories make the system to "jump" from one trajectory to other, generating what is known as chaos.

For example, consider the Hennon map, given by the difference equations:

$$\begin{aligned} x_{t+1} &= 1 - ax_t^2 + y_t \\ y_{t+1} &= bx_t . \end{aligned} \tag{2.7}$$

Figure 2.5 shows the variable x plotted as time goes on for $a=1.4$ and $b=0$. For these values of a and b the Hennon map presents a chaotic behavior. Its strange attractor can be visualized in the return map of Figure 2.6

Another example of a chaotic system is given by the Mackey-Glass equation (Glass 1988), which has being widely used to model many biological rhythms. It is given by the differential equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) . \tag{2.8}$$

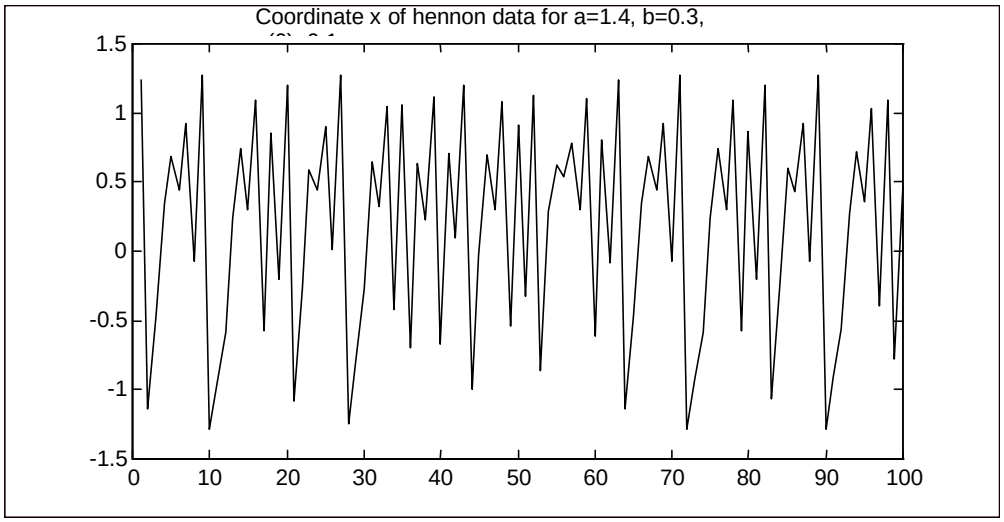


Figure 2.5. Coordinate x of Hennon map for a =1.4, b = 0.3 and $x(0)$ = 0.1

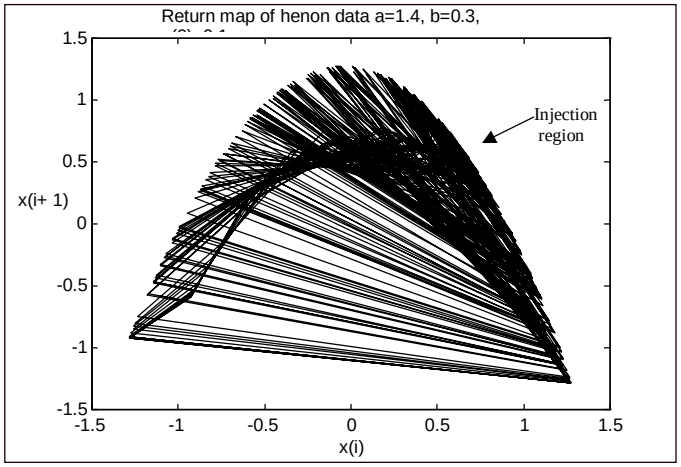


Figure 2.6. A strange attractor of Hennon map for $A = 1.4$, $b = 0.3$ and $x(0) = 0.1$

For $a = 0.2$, $b = 0.1$ and $\tau = 17$, this equation results in chaotic behavior. Figure 2.7 shows a numerical solution of the Mackey-Glass equation and Figure 2.8 is its corresponding return map.

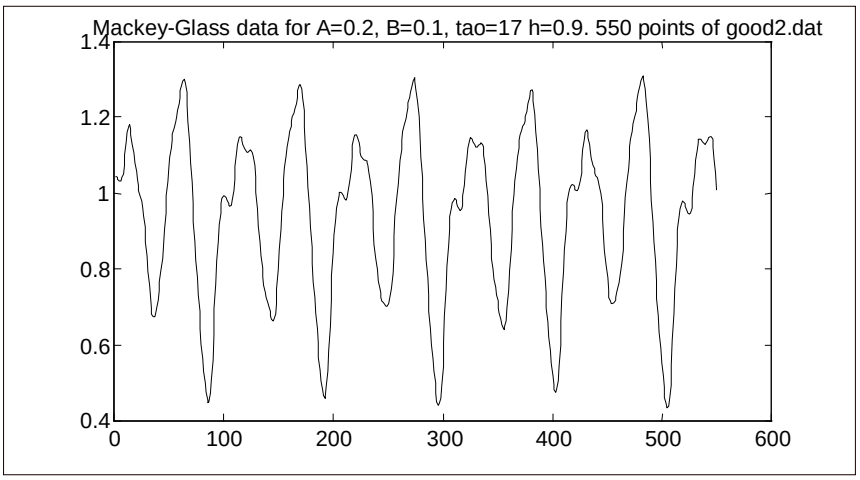


Figure 2.7. Mackey-Glass data. Equation (2.8)

Through the years, the existence of chaos has been characterized in time series using several methods, among others: analysis of Fourier spectra, fractal power spectra, entropy,

fractal dimension, and calculation of Lyapunov exponents. However, several of these methods have proven not to be very efficient. In recent years, the calculation of Lyapunov exponents has been a common way to determine if a time series resulting from a unknown dynamical system is chaotic. Section 2.7 describes with detail Lyapunov exponents.

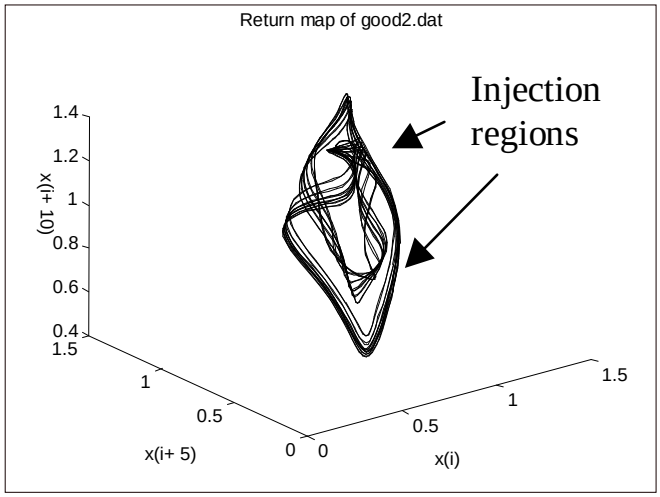


Figure 2.8. A return map of data at Figure 2.7 with a lag = 5

2.4 Stability

A system is said to be *locally stable* if after a small perturbation away from the steady state it returns to its steady state as $t \to \infty$. If a small perturbation induces a change in the dynamics so that the original dynamics is not reestablished, then the steady state or limit cycle is said to be *unstable*.

For example, in the Poincaré oscillator (equation 2.5) $r = 0$ is a steady state; at that point $\frac{dx}{dt} = 0$. However, any small perturbation away from $r = 0$ will lead the path to the stable limit cycle at $r = 1$. Therefore, the steady state at r=0 is unstable. We can see at Figure 2.4 two examples of such stability.

To determine stability and asymptotic stability in nonlinear systems it is common to apply the *direct method of Lyapunov*. This method is based on the two Lyapunov's theorems that state:

"Theorem 1: The equilibrium state $\bar{\mathbf{x}}$ is *stable* if, in a small neighborhood of $\bar{\mathbf{x}}$, there exists a positive definite function V(x) such that its derivative with respect to time is negative semi-definite in that region" (Haykin 1994, pp. 547).

"Theorem 2: The equilibrium state $\bar{\mathbf{x}}$ is *asymptotically stable* if in a small neighborhood of $\bar{\mathbf{x}}$ there exists a positive definite function V(x) such that its derivative with respect to time is negative definite in that region" (Haykin 1994, pp. 547).

A scalar function V(x) that satisfies these requirements is called a *Lyapunov function* for the equilibrium state $\bar{\mathbf{x}}$. Recall that a function V(x) is *positive definite* in the estate space $\mathcal{L}$ if for all x in $\mathcal{L}$ it satisfies the following requirements:

1. The function V(x) has continuous partial derivatives with respect to the elements of the state vector x.
2. $V(\bar{\mathbf{x}})=0$.
3. $V(x)>0$ if $\mathbf{x} \neq \bar{\mathbf{x}}$.

Using this definition and according to Theorem 1, the equilibrium state $\bar{\mathbf{x}}$ is *stable* if:

$$\frac{d}{dt}V(\mathbf{x}) \leq 0 \quad \text{for } \mathbf{x} \in U - \bar{\mathbf{x}}, \text{ where } U \text{ is a small neighborhood around } \bar{\mathbf{x}}.$$

According to Theorem 2, the equilibrium state $\bar{\mathbf{x}}$ is *asymptotically stable* if:

$$\frac{d}{dt}V(\mathbf{x}) < 0 \quad \text{for } \mathbf{x} \in U - \bar{\mathbf{x}}.$$

It is important to point out that stability in the sense of Lyapunov, as described before, is applied only to fixed-point attractors, and it cannot be applied to the particular cases of nonlinear dynamical systems exhibiting chaotic behavior (Haykin 94). In fact for a chaotic dynamic, if perturbed, the system finds a new trajectory with new initial conditions.

2.5 Bifurcation

Any value of a parameter in which the number and/or stability of steady states changes is called a *bifurcation point*, and the involved system is said to undergo a *bifurcation.* At bifurcation points, systems are *structurally unstable*; that is, the main qualitative features of the system change. For example, a biological system can be approximated by a *quadratic map*:

$$x_{i+1} = a x_i (1 - x_i) \qquad 0 \leq a \leq 4. \tag{2.9}$$

Its steady state x^* is a value for which $x_i = x_{i+1} = x^*$; that is, $x^* = 0$ and $x^* = (a-1)/a$. Figure 2.9 shows an example of a time series for the quadratic map with $a = 2.5$. It converges to the fixed point $(a-1)/a = 0.6$. As the value of parameter a increases in the range of $3.0<a<3.75$, successive period-doubling occurs in the time series (see Figures 2.10 and 2.10) and the steady states are limit cycles now.

Figure 2.12 shows the quadratic map for $a = 4.0$. Now its behavior is chaotic. Notice that the time series seems to be random, however it was generated by a perfectly well defined equation.

Kaplan and Cohen (1990) as well as others summarize the following as characterizations of deterministic chaos:

- Chaotic trajectories are aperiodic and deterministic.
- Chaotic systems are extremely dependent on initial conditions.
- The chaotic behavior is bounded and presents a strange attractor.
- There is a particular pattern associated with chaotic behavior.

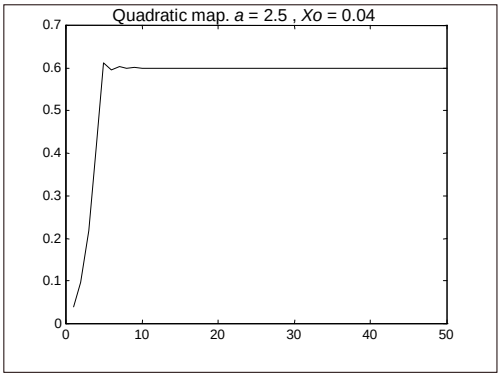


Figure 2.9. Quadratic map for $a = 2.5$

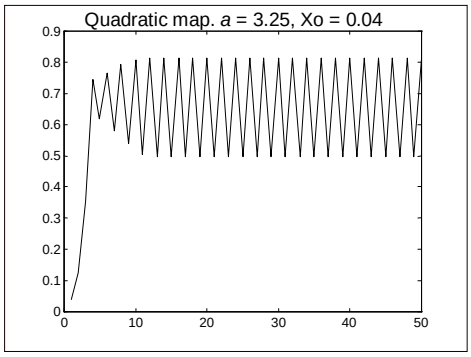


Figure 2.10. Quadratic map for $a = 3.25$

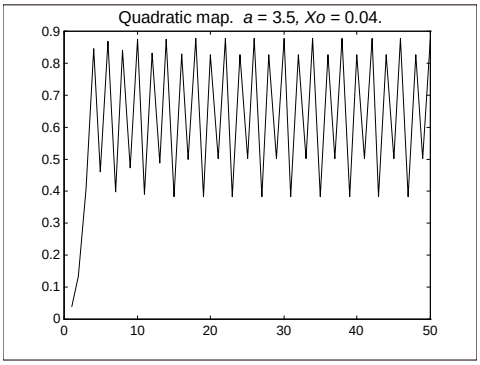


Figure 2.11. Quadratic map for $a = 3.5$

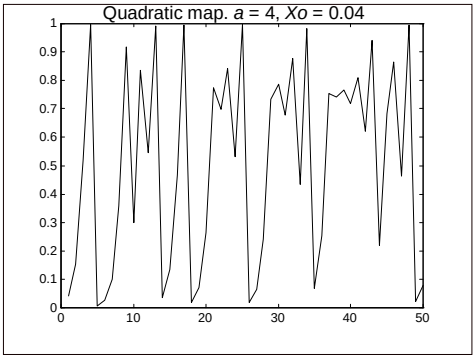


Figure 2.12. Quadratic map for $a = 4.0$

2.6 Time Series and Phase-Space Reconstruction

In most of the cases, the only information available of an unknown non-linear, d-dimensional system is a one-dimensional time series $\{x(t),\ t = 1, 2, \ldots T\}$, $x(t) \in \Re$. According to the embedding theory defined by Takens (1981), a space of dimension $M \geq 2d+1$ can be reconstructed from these one-dimensional observations. The properties of the attractor of the real system of dimension d may be observed in the reconstructed space of dimension M. This can be done by defining the vector:

$$\mathbf{y}(t) = (x(t), x(t+\tau), x(t+2\tau)...x(t+(M-1)\tau)), \tag{2.10}$$

for some set of time lag τ. $\mathbf{y}(t) \in R^M$.

For $M \geq 2d+1$, the dynamical properties of a map

$$\mathbf{\Phi}(\mathbf{y}(t)) = \mathbf{y}(t+1) \quad \mathbf{\Phi} : R^M \rightarrow R^M, \tag{2.11}$$

are topologically the same as the unknown system. M is called the *embedding dimension* (Kaashoek and Van Dijk 1994).

There are several methods to calculate an appropriate value for the embedding dimension M. Indeed, the value of the time lag τ need to be selected. Abarbanel et al., (1990) describe in detail a method to reconstruct the embedding space. In their method they choose a value of τ based on the auto-correlation function of the original scalar measurements. The calculation of M is an iterative process involving the calculation of the correlation function for increasing values of M, starting at one. They choose the value of M when the structure of the correlation dimension does not change with an increment of M. These methods are applicable

if the dimension is less or equal to seven. At present, there is no technique for systems with larger dimensions.

2.7 The Lyapunov Exponents

Since the 1970's, the study of non-linear dynamics has focused on concepts like the KAM theorem, Lyapunov exponents, Smale's horseshoe, Feigenbaum constants and fractal dimensions (Tong 1993). Several techniques have been developed to detect and analyze non-linear behavior; among them are: return maps, Poincaré sections, Lyapunov exponents, correlation dimension, fractal dimension, Kolmogorov entropy and spectral analysis. Probably today the most popular tool to detect chaos in an unknown dynamical system from a time series are the Lyapunov exponents (LE for short).

Given a system:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}) \qquad \mathbf{x}(t) \in R^d \quad , \quad \mathbf{x}(t_o) = \mathbf{x}_o \,, \tag{2.12}$$

where f is a vector field that does not depend on time, i.e., autonomous, the *Lyapunov exponents* $\lambda_1, \lambda_2 \ldots \lambda_d$ are the average rates of expansion ($\lambda_i > 0$) or contraction ($\lambda_i < 0$) near a limit set of the dynamical system (Parker and Chua 1989). In other words, the LE are a quantitative measure of the divergence of several trajectories in the system. The subscript "*i*" refers to the *i-th* direction of the *d*-dimensional phase space where the system is embedded. The variable *d* represents the Lyapunov dimension or phase space dimension. The LE are *invariant measures* that characterize the attractors of a dynamical system; that is, LE do not change when the initial conditions of a trajectory are modified or if some perturbation occurs in the system.

The values of the LE give us information about the type of a limit set because they measure the exponential attraction or separation in time of two adjacent trajectories in phase space with different initial conditions. Table 2.1 summarizes the interpretations of LE for different kinds of systems (Parker and Chua 1989). If at least one LE is positive, the system presents chaotic motion; hence it is very dependent on initial conditions. If that is the case, the magnitude of the LE reflects the time scale on which the dynamics of the system become unpredictable.

According to Brown et al. (1991), the Lyapunov exponents may be determined by observing the evolution of small deviations in an orbit **w**(k). This orbit is defined in the phase space of the system, that is, $\mathbf{w(k)} \in R^d$ for k = 1,2,…N. It may be assumed that such an orbit is a map; that is, it is a discrete version of the system flow. The orbit should satisfy:

$$\mathbf{w}(k+1) = f(\mathbf{w}(k)). \tag{2.13}$$

A perturbation in the orbit can be described as:

$$\delta\mathbf{w}(k+1) = \underline{D}\mathbf{f}(\mathbf{w}(k))\delta\mathbf{w}(k), \tag{2.14}$$

where $\underline{D}\mathbf{f}(\mathbf{w})$ is the $d \times d$ Jacobian matrix evaluated along the orbit.

The Lyapunov exponents are the logarithms of the eigenvalues of the matrix defined by:

$$\lim_{K \to \infty} \left[(\underline{D}\mathbf{f}^K)^T (\underline{D}\mathbf{f}^K)^{1/2K} \right], \tag{2.15}$$

where $\underline{D}\mathbf{f}^K = \underline{D}\mathbf{f}(K) \cdot \underline{D}\mathbf{f}(K-1) \cdots \underline{D}\mathbf{f}(1)$, and $\underline{D}\mathbf{f}(K) = \underline{D}\mathbf{f}(\mathbf{w}(K))$.

Normally, the LE are referred in order of their numerical size, that is: $\lambda_1 \geq \lambda_2 \geq \ldots$ LE are expressed in terms of bits of information/s for continuos systems and bits/iteration for discrete systems.

The *Lyapunov spectrum* is closely related to fractional dimensions associated with the attractor. Examples of fractional dimension-like quantities are: fractal dimension, information dimension and correlation exponent. For example, the information dimension d_f is related to LE as:

$$d_f = \frac{j + \sum_{i=1}^{j} \lambda_i}{\left|\lambda_{j+1}\right|}, \quad \text{where } j \text{ is defined by the condition: } \sum_{i=1}^{j} \lambda_i > 0, \text{ and } \sum_{i=1}^{j+1} \lambda_i < 0. \tag{2.16}$$

For chaotic systems, the magnitude of the LE reflects the time scale on which the system dynamics become unpredictable.

Table 2.1. Values of Lyapunov exponents for different

types of limit sets (Parker and Chua 1989)

Steady State	*Flow*	*Lyapunov exponents*
Equilibrium point	Point	$0 > \lambda_1 \geq \ldots \geq \lambda_n$
Periodic	Circle	$\lambda_1 = 0$ $0 > \lambda_2 \geq \ldots \geq \lambda_n$
Two-periodic	Torus	$\lambda_1 = \lambda_2 = 0$ $0 > \lambda_3 \geq \ldots \geq \lambda_n$
K-periodic	K-torus	$\lambda_1 = \ldots = \lambda_k = 0$ $0 > \lambda_{k+1} \geq \ldots \geq \lambda_n$
Chaotic	Cantor-like	$\lambda_1 > 0$ $\sum \lambda_i < 0$

2.7.1 Numerical Methods to Calculate Lyapunov Exponents

A numerical method is needed to calculate the LE from a set of observations of the system when equations describing it are not available. In 1985, Wolf et al., proposed one of the first practical methods to calculate the maximum LE from a time series. Since then, many others have been proposed; examples are presented at Parker et al. (1989), Brown et al. (1991), Abarbanel et al. (1990), Rosenstein et al. (1993), Banbrook et al. (1997), Barna and Tsuda (1993), among others. There are numerical methods available for both the calculation of the maximum LE or calculation of the LE spectra.

Following is the description of two numerical methods for calculation of LE: the one proposed by Wolf et al. (1985) which has historical importance, and the one proposed by Gencay and Dechert (1992) which is of fundamental importance in this research.

2.7.1.1. The Method of Wolf et al.

This method can calculate non-negative LE. The authors have published versions of their method to calculate the LE spectra or only the largest positive exponent. This algorithm is easy to use and requires fewer amounts of data than other methods available at the time that it was created. However, as in most of these numerical methods, its reliability depends mainly on the good sense of the experimenter when choosing or calculating several parameters needed by the algorithm, as well as in the accuracy and size of the data set. Appendix A shows

a structured version of the algorithm to calculate the maximum LE using fixed evolution time as described at (Wolf et al. 1985).

The main idea of this algorithm is to monitor the long-term evolution of a single pair of nearby orbits of the system in order to estimate λ_1, the largest LE. The algorithm attempts to approximate the local tangent space about a fiducial orbit of the system. The process starts calculating the time-delay reconstructed coordinates **y** as described at section 2.6. After that, it finds the nearest neighbor in the reconstructed space to the first point in the orbit. The magnitude of the difference vector is recorded. Subsequently, the point evolves along its trajectory a given number of steps. The magnitude of the final separations is determined, and a contribution to λ_1 is calculated as the logarithm of the final separation divided by the initial separation. All contributions are averaged over the length of the time series. If the distance between neighbors becomes too large, the algorithm abandons this point and searches for a new neighbor.

This algorithm requires the following input parameters:

1. The number of points in the time series (*N*). Wolf et al. (1985) suggest that at least 10^d points are needed, but Abarbanel et al. (1991) suggest 20^d as the right amount of data to use with this algorithm. Greater accuracy requires a longer time series.
2. The embedded dimension of the system (*d*). The determination of this value requires the application of other numerical methods. Wolf et al. (1985) and Abarbanel et al. (1991) suggest some ways to calculate it. The behavior of the algorithm is very dependent on the accuracy of this value. A too large value increments noise; a too short value produces loss of information.
3. Reconstruction time delay (τ_d). This value is chosen in a way to make the *d* components of the system as "orthogonal" as possible. A popular way to calculate it is finding the first zero of the auto-correlation function of the time series, but experimentation is sometimes needed to find a correct value.
4. Time between successive measurements in the time series (T_s). This is the inverse of the sampling rate for the data.

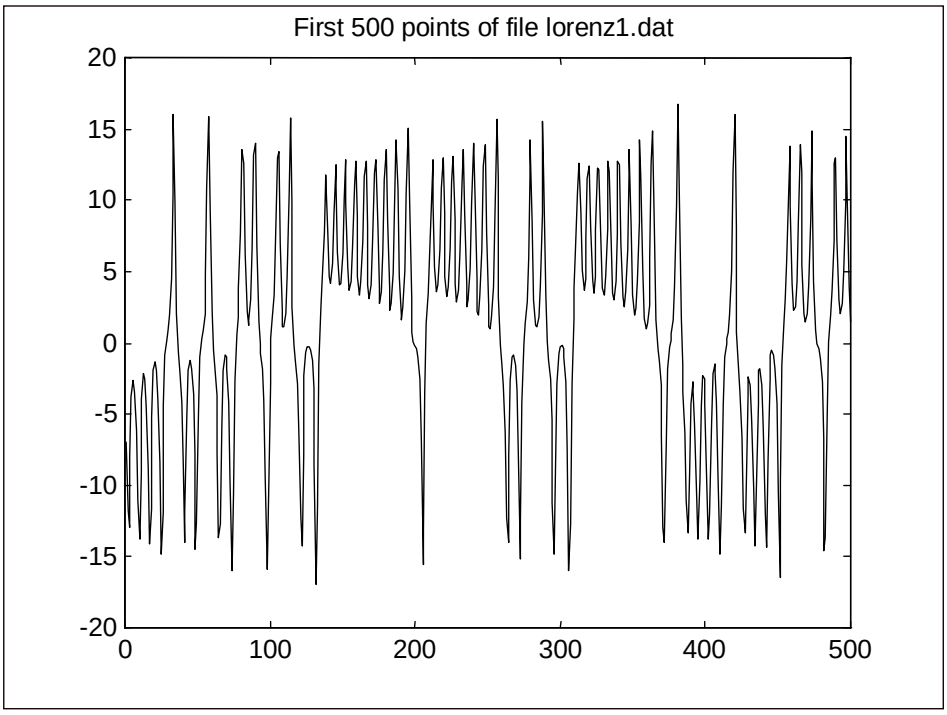
5. Maximum distance that the algorithm will look for neighbors (S_{max}). Making this variable smaller will increase accuracy. The authors suggest it should be less than 1% of the macro-scale of the attractor, and to experiment with its value.
6. Minimum distance that the algorithm will look for neighbors (S_{min}). This variable is used to avoid noise.
7. Evolution time (T_E). Given in time series steps, it is the time that a given pair of neighbors are allowed to evolve before replacement. This variable affects greatly the accuracy of calculation. Experimentation is required to find a correct value.

The well-known Lorenz system:

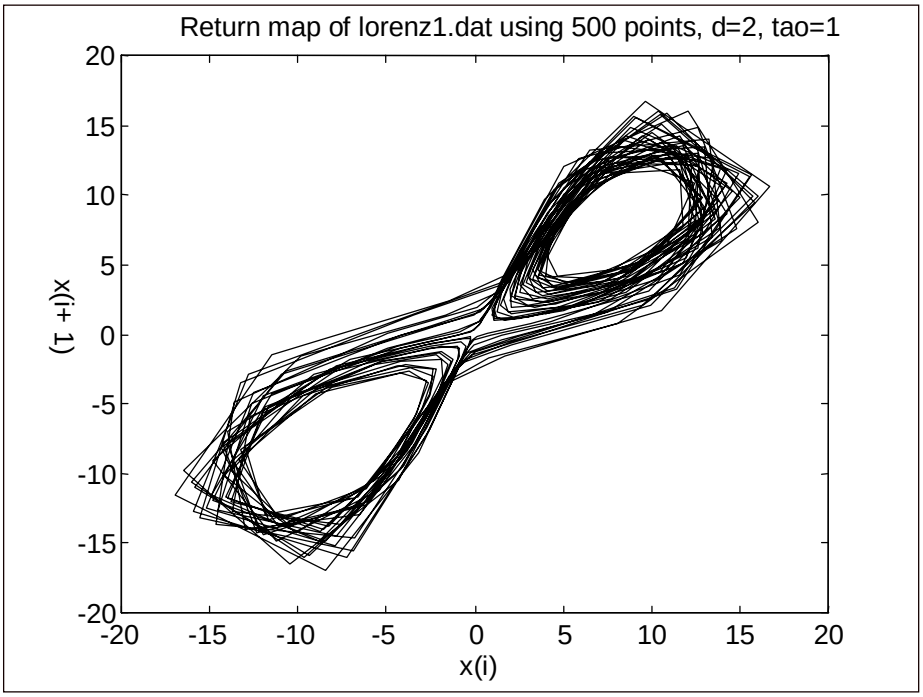
$$\begin{aligned} \dot{x} &= a(x - y), \\ \dot{y} &= x(b - z) - y, \\ \dot{z} &= xy - cZ, \end{aligned} \tag{2.17}$$

presents chaotic behavior when a = 16.0, b = 45.92, c =4.0. Figure 2.13 (a) shows 500 points of a numerical solution to the Lorenz system, and (b) the return map for the same data showing the strange attractor of the system. The maximum LE of this time series was calculated using the Wolf et al. algorithm with the following parameters: number of points: 4,500; embedded dimension: 3; time delay: 13; time period of data: 0.1; maximum distance to look for neighbors: 0.4; minimum distance to look for neighbors: 0.00001 and evolution time: 20. Figure 2.14 shows the convergence of LE , getting as the last result 1.315. It is known that the true value of the maximum LE for the Lorenz system is 1.5.

Wolf et al. suggest running the algorithm for different values of evolution time and choosing the one where λ_1 presents some stationary. Figure 2.14 plots the obtained values for the Lorenz system for evolution times between 2 and 60. Notice that a plateau is around 12 and 21. The average λ_1 on this range is 1.3518.



(a)



(b)

Figure 2.13. Lorenz system. (a) Data. (b) Return map with lag = 1.

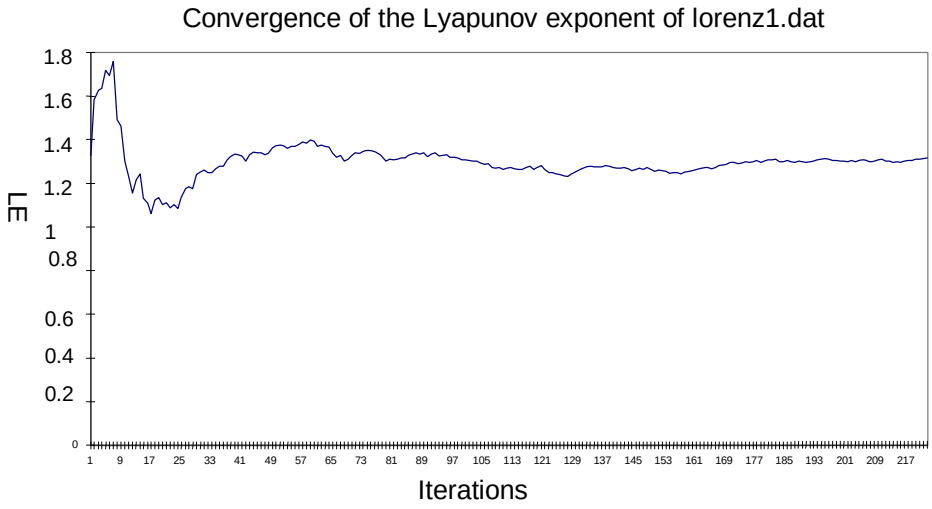


Figure 2.14. Convergence in the calculation of Lyapunov Exponent.

The main drawback of this algorithm is its strong dependence on the size of the data set and the accuracy of the input parameters, which by themselves are hard to determine in real life cases. This is the case of electrocardiograms, which are believed to be chaotic signals. Figure 2.15 shows a sample of an electrocardiogram signal. This is a fraction of record No. 123 of the database produced by Harvard (1992). The signal was sampled at 360Hz. The Wolf's algorithm was used with 37,500 points of this record, an embedded dimension of 6 (as suggested by Babloyantz & Destexhe [1988]), time delay of 15 points (calculated using the auto-correlation function), time period of 0.028, a maximum distance to look for neighbors of 0.008 and a minimum distance of 1.0e-5. Figure 2.16 shows a plot of the convergence of λ_1 as the algorithm goes over the data (only the first 150 points are plotted). Notice that the plot is rougher than that in Figure 2.14.

Figure 2.17 shows the obtained values of λ_1 for evolution times from 80 to 384. Notice that λ_1 does not converge to a well-defined stable value. Some stability is shown at the range from 254 to 384, where the mean of λ_1 is 0.0931. The true value of the maximum exponent of an ECG has not been determined. Several numerical calculations, using different data sets of ECG signals, have being reported: Babloyantz and Destexhe (1988) obtained values of $\lambda_1 = 0.38 \pm 0.08$ for ECG; Karanam (1996) obtained calculated lower limits of λ_1

from 0.11 to 0.27; Casaleggio et. al (1995) obtained values ranging from 7.6 to 29.1 for different samples of ECG.

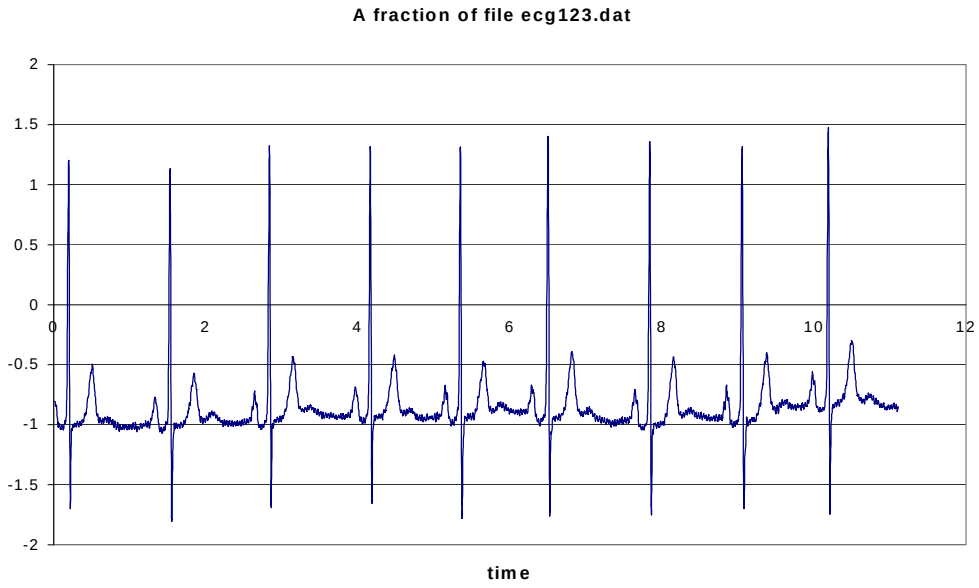


Figure 2.15. A fraction of file ecg123.dat

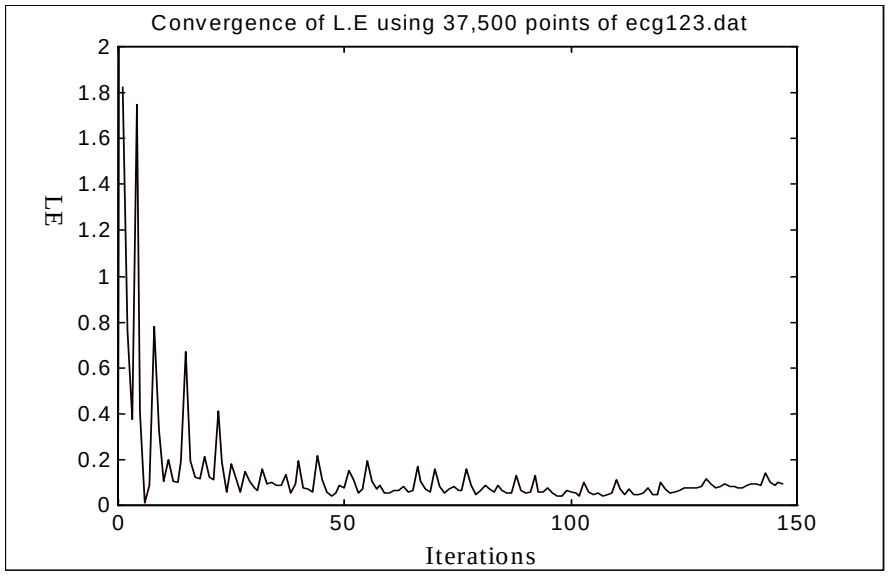


Figure 2.16. Convergence of Wolf's algorithm for ecg123.dat

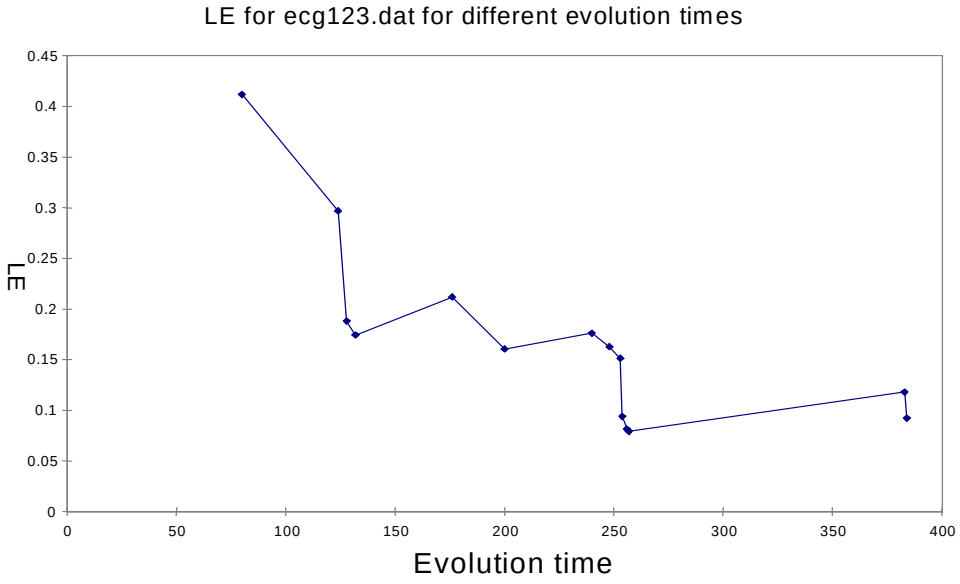


Figure 2.17. Maximum LE of the ECG signal
for different evolution times

2.7.1.2. Estimation of Lyapunov Exponents using
Feed-forward Neural Networks

Gencay and Dechert (1992) developed an algorithm to estimate all the Lyapunov exponents of an unknown dynamical system using a technique based on a multivariate feed-forward neural network. The same algorithm has been used by several authors (Sattin 1997, González et al. 1995, Kaashoek and Van Dijk 1994) for different applications. The main idea of this algorithm is to estimate a function from the observations of the system that is topologically equivalent to the one describing the system, then calculate the LE of that function. As stated by equations 2.13 to 2.15, the Lyapunov exponents of a dynamical system can be calculated by evaluating the Jacobian of the function f (see equations) that describes its trajectory. Multi-layer feed-forward neural networks can approximate a function and its derivatives to any degree of accuracy; therefore, they are used in this algorithm as a non-parametric estimation technique. The algorithm works as follows:

Suppose a dynamical system $f : \Re^n \to \Re^n$ with a trajectory

$$x_{t+1} = f(x_t), \quad t = 0,1,2,\ldots \quad . \tag{2.18}$$

Associated with this system there is a measurement function $h:\Re^n \to \Re$ that generates the sequence of observations $\{y_t\}$:

$$y_t = h(x_t). \tag{2.19}$$

The time series $\{y_t\}$ is all the information available about the system. As described at section 2.3, a map of dimension m with equivalent properties to the original system can be constructed from the one-dimensional observations y_t as:

$$y_{t+1}^m = g(y_t^m), \tag{2.20}$$

where $y_t^m = (y_{t+m-1}, y_{t+m-2}, ..., y_t)$ and $y_{t+1}^m = (y_{t+m}, y_{t+m-1}, ..., y_{t+1})$.

The dynamical properties of f and g are the same. The map g to be estimated can be represented as:

$$g:\begin{pmatrix} y_{t+m-1} \\ y_{t+m-2} \\ \vdots \\ \vdots \\ y_t \end{pmatrix} \to \begin{pmatrix} v(y_{t+m-1}, y_{t+m-2} \cdots y_t) \\ y_{t+m-1} \\ \vdots \\ \vdots \\ y_{t+1} \end{pmatrix}. \tag{2.21}$$

The function v can be estimated using a feed-forward network with one hidden layer (see Figure 2.17)

$$v_{N,m}(x;\beta,w,b) = \sum_{j=1}^{L} \beta_j k(\sum_{1}^{m} w_{i,j} x_i + b_j), \tag{2.22}$$

where:

$$k(x) = \frac{\beta}{1+\exp(-wx-b)}, \tag{2.23}$$

the w_{ij} are weights from input nodes to hidden nodes, β_j are weights from hidden to output node, L is the number of hidden nodes, m it he number of inputs, corresponding to the imbedded dimension and x are external inputs to the network. The optimization criteria consists on minimizing the square-error cost function over the whole trajectory of size T:

$$E(\beta_j, w_{i,j}) = \sum_{t=0}^{T-m-1} \frac{1}{2}(y_{t+m} - v(y_t^m;\beta,w,b). \tag{2.24}$$

The estimation of parameters can be carried out using the well-known algorithm back-propagation (Rumelhart et al. 1986) or, as suggested by Gencay and Dechert, using the Polak-Ribiere conjugate gradient method (Press et al. 1988). The latter requires one to use the partial derivatives of the cost function which are:

$$\frac{dE}{d\beta_l} = \sum_{t=0}^{T-m-1} k(\sum_{i=1}^{m+1} w_{il} x_i)(-y_{t+m} + \sum_{j=1}^{L} \beta_j k(\sum_{i=1}^{m+1} w_{i,j} x_i)), \tag{2.25}$$

$$\frac{dE}{dw_{l,k}} = \sum_{t=0}^{T-m-1} (\beta_k x_l k'(\sum_{i=1}^{m+1} w_{i,l} x_i))(-y_{t+m} + \sum_{j=1}^{L} \beta_j k(\sum_{i=1}^{m+1} w_{i,j} x_i)), \tag{2.26}$$

for $l = 1...(m+1),\ k = 1..L$,

where the substitutions $b_j = w_{m+1,j}$ and $x_{m+1} = +1$, have been made to simplify the equations.

Once that g is known, a method as the one described by Parker and Chua (1989) can be used to calculate the Lyapunov Exponents based on the Jacobian Matrix of g, which is given by:

$$(Dg)_{y_n^m} = \begin{pmatrix} v_m & v_{m-1} & v_{m-2} & \dots & v_2 & v_1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & 1 & 0 \\ & & & & & \end{pmatrix}, \tag{2.27}$$

where:

$$v_m = \frac{dv}{dy_{t+m-1}}, \tag{2.28}$$

and the Lyapunov exponents λ_i are computed as:

$$\lambda_i = \frac{1}{T} \sum_{t=0}^{T} \ln \left\| y_t^j \right\|,\ i = 1...m. \tag{2.29}$$

The ability of equation (2.22) to approximate arbitrary functions has proven to be very good (Hornik et al. 1990), even for a chaotic series. Figure 2.19 shows the result of approximating a time series of a Hennon map (see equation 2.7) using 2 input nodes and 8 hidden nodes.

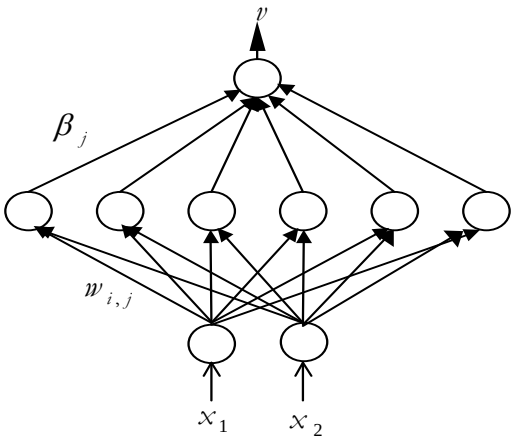


Figure 2.18. A feed-forward net with two external inputs.

The network was trained using the standard back-propagation algorithm. This one-step prediction is so good that the plots of original and calculated series can not be distinguished at the resolution of Figure 2.19. The Mean Square Error of the cost function after training of the network was 7.8E-4. Similar results were obtained when training the network using the Polak-Ribiere conjugate gradient method, with the observation that even though the convergence was much faster, the final value of the error was not better than with back-propagation.

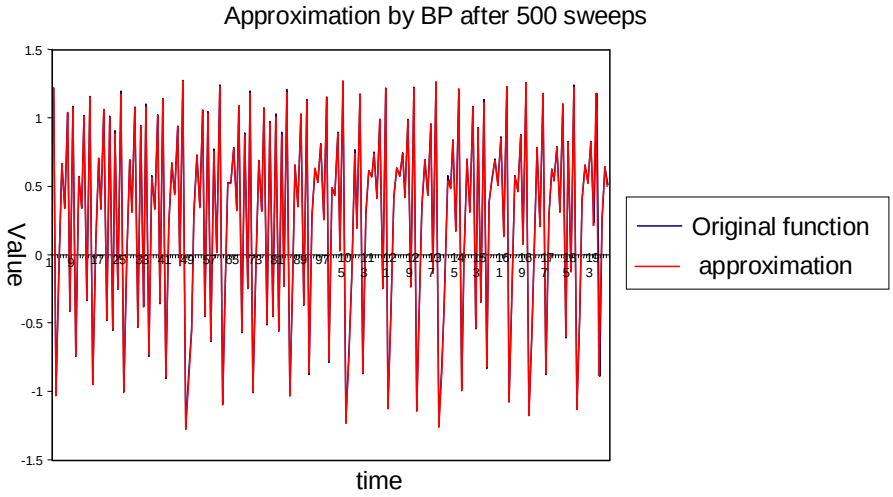


Figure 2.19. Aproximation of a Hennon series using a feed-forward net with 2 inputs and 6 hidden nodes. LE = 0.6064.

This kind of approximation or single-point prediction (see section 3.1) also works well for ECG signals. Figure 2.20 shows a plot of the first 1,000 points for the estimation of 3,978 points of a series. For this example 5 input nodes and 10 hidden nodes were used. The Mean Square Error was 3.29E-4.

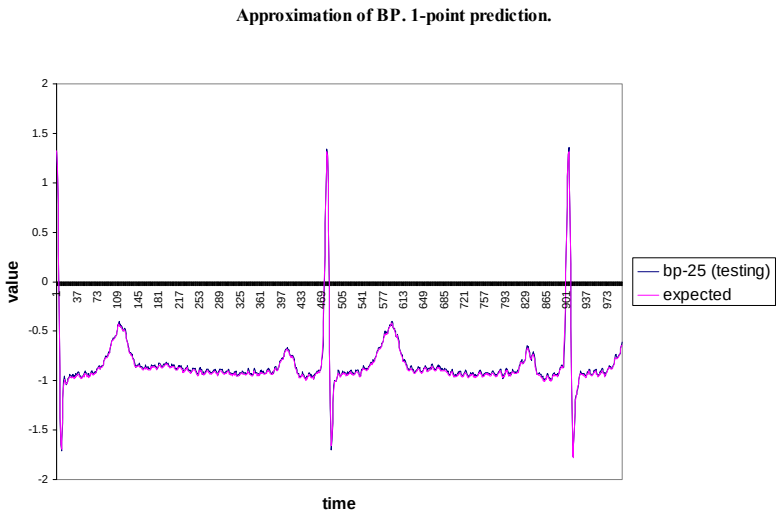


Figure 2.20. Approximation of an ECG signal using a feed-fordward net with 5 inputs and 10 hidden nodes. LE = 0.1823

CHAPTER III
PREDICTION AND ARTIFICIAL NEURAL NETWORKS

The possibility to foresee future events has always been of great interest for all disciplines. Tong (1993) identifies three basic activities in the construction of a forecasting model: first determine the main characteristics of the data, next construct a model using the available theory and such characteristics, and finally verify if the model is able to represent the features of the system. Such activities may need to be executed several times.

However, this is not an easy task, particularly for the cases where the unknown dynamical model is non-linear or chaotic, because the simple observation of the outputs gives few clues to determine the model that will represent the dynamics accurately.

Despite the difficulties of forecasting, many techniques to predict time series have been proposed. According to Brockell (1991), the problem of forecasting consists of the evaluation of future values of a time series, x_{T+h}, $h \geq 1$, based on the observations of its past values $x_1, x_2 ... x_T$. In general, prediction may be seen as a function approximation problem (Príncipe et al. 1997) which consists of defining a complicated function $f(\mathbf{y})$ using other function $\widetilde{f}(\mathbf{y})$ that is a combination of simpler functions:

$$\widetilde{f}(\mathbf{y}) = \sum_{i=1}^{N} a_i \varphi_i(\mathbf{y}). \tag{3.1}$$

If the bases functions $\varphi_i(\mathbf{y})$ are a linear combination of past outputs and/or inputs, the model is said to be *linear*. When the bases are nonlinear with respect to the past signal, the model is *non-linear*. For many years linear models have been the most popular tools to predict, but obviously they are not able to represent accurately any non-linear dynamical system.

3.1 Types of Prediction

A *predictor map* defined by (3.1) can be written as:

$$\mathbf{y}(t+1) = \mathbf{F}(\mathbf{y}(t), \mathbf{a}). \tag{3.2}$$

Single-step prediction takes place when several observations of past values are used by the predictor to calculate the next point. This is also called *next-point prediction* or *one-point*

prediction. The predictor map is applied once: $\mathbf{y}(t+1) = \mathbf{F}(\mathbf{y}(t), \mathbf{a})$. This kind of prediction is normally carried out by *non-autonomous predictors*, where each prediction of time t requires as external inputs observations at times *t-1, t-2… t-d+1*. It is also possible to carry out single-step prediction using an *autonomous predictor*, which is able to fully represent the solution to the dynamic of the system, and therefore it only requires as external input the initial conditions of the system. From time *t=0* the predictor will calculate the output at any time t without any external inputs.

Point-by-point prediction takes place when, in a iterative way, the predictor calculates outputs at times *t, t+1 , t+2 … .* Here the predictor map is applied several times: $\mathbf{y}(t+k) = \mathbf{F}(\mathbf{y}(t+k-1), \mathbf{a}) = \mathbf{F}(\mathbf{F}(\mathbf{y}(t+k-2), \mathbf{a}), \mathbf{a}) = ... = \mathbf{F}^k(\mathbf{y}(t), \mathbf{a})$. A non-autonomous predictor will require feedback from its own predictions to calculate new values when the value to be predicted is such that there are no more available observations. Point-by-point prediction is required for long-time prediction.

3.2 Linear Models for Prediction.

A linear predictor can be seen as a filter, as pictured at Figure 3.1 (Burrs 1991). Given a discrete time signal with unknown parameters, future outputs can be estimated as a linear combination of input values, output values, or both. In its most general form, the estimation $\tilde{y}(n)$ is given by:

$$\hat{y}(n) = -\sum_{i=1}^{N} a_i y(n-i) + \sum_{j=1}^{M} b_j x(n-j), \qquad (3.3)$$

where $x(n)$ and $y(n)$ are inputs and outputs of the system respectively; a_i and b_j represent estimated values. Normally the only known values are the outputs of the system.

This system can be described as:

$$A(z)Y(z) = B(z)X(z), \qquad (3.4)$$

then:

$$\hat{H}(z) = \frac{Y(z)}{X(z)} = \frac{B(z)}{A(z)}. \qquad (3.5)$$

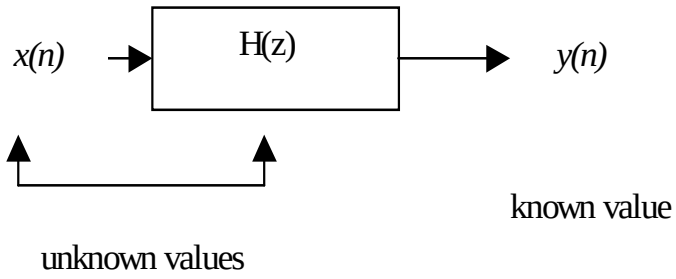


Figure 3.1. A linear prediction model

The obtained parameters are estimations of the poles and zeroes of the system. This pole-zeroes model is known as the ARMA model, initials standing for *Auto-Regressive, Moving-Average*.

It is also possible to define simpler models which take in account only past inputs or past outputs. They are the all-zeroes model, defined as:

$$\hat{y}(n) = \sum_{j=1}^{M} b_j x(n-j), \tag{3.6}$$

also called the MA model, initials standing for *Moving Average*, and the all-poles model, defined as:

$$\hat{y}(n) = -\sum_{i=1}^{N} a_i y(n-i), \tag{3.7}$$

also called the AR model, initials standing for *Auto Regressive*.

3.2.1 Implementation of an AR Model

AR models are very popular because in most applications only the output values of the system are known. Besides they are easy to implement, but they require heavy numerical calculations.

In this model, it is required to solve the equation:

$$\hat{y}(n) = -\sum_{i=1}^{M} a_i y(n-i),$$

in a way that the error defined by the function $E = < e^2(n) >$ is minimized.

For any set $\{a_i\}$, the error when estimating $y(n)$ is given by:

$$e(n) = y(n) - \hat{y}(n) = a_0 y(n) - \sum_{i=1}^{N} a_i y(n-i) = \sum_{i=0}^{N} a_i y(n-i), \tag{3.9}$$

where $a_0 = 1$.

According to the orthogonal principle, the minimum error is obtained when $\{a_i\}$ is chosen such that E is orthogonal to $y(n-i)$ for all $i=0..N$, that is,

$$\langle e(n)\, y(n-j) \rangle = 0 \quad j=1, 2 \ldots N, \tag{3.10}$$

which is equivalent to:

$$\sum_{i=0}^{N} a_i \langle y(n-i)\, y(n-j) \rangle = 0 \quad j = 1,2,\ldots N. \tag{3.11}$$

If $y(n)$ is a stationary signal, the auto-correlation function defined by:

$$r_{i-j} = \langle y(n-i)\, y(n-j) \rangle, \tag{3.12}$$

can by substituted in equation 3.11 which becomes:

$$\sum_{i=0}^{N} a_i r_{i-j} = 0 \quad j = 1,2..N. \tag{3.13}$$

The minimum-squared error is given by the equation:

$$E = \sum_{i=0}^{N} a_i r_i. \tag{3.14}$$

Combining equations (3.13) and (3.14), the following equation system is obtained:

$$\begin{bmatrix} r_0 & r_1 & r_2 & \cdots & r_p \\ r_1 & r_0 & r_1 & \cdots & r_{p-1} \\ r_2 & r_1 & r_0 & \cdots & r_{p-2} \\ & & & \cdots & \vdots \\ r_p & r_{p-1} & r_{p-2} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} a(0) \\ a(1) \\ a(2) \\ \vdots \\ a(p) \end{bmatrix} = \begin{bmatrix} E_p \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{3.15}$$

where $a(i) = a_i$, and p is the number of coefficients to calculate or *degree* of the predictor. E_p is the minimum error obtained when using p coefficients.

There are many ways to solve this system. For example, N. Levinson constructed a recursive algorithm to calculate the coefficients $\{a_i\}$ based on the fact that in a stationary signal the correlation matrix is symmetric and "Toeplitz," that is, its j-th. row counting from bottom is always the inverse of its j-th. row counting from the top. For a detailed description of the Levinson method see Oppenheim (1989).

A non-autonomous, single-point predictor of degree 10 was trained using the Levinson method to learn the ECG signal given at Figure 3.2 (a). Figure 3.2 (b) shows the results obtained when predicting the next point. The predictor takes 10 past values of the original signal to predict the next point, obtaining satisfactory results. However, it is important to point out that single-point prediction is neither very difficult nor useful in most cases. Almost any kind of predictor could obtain satisfactory results for one-point prediction.

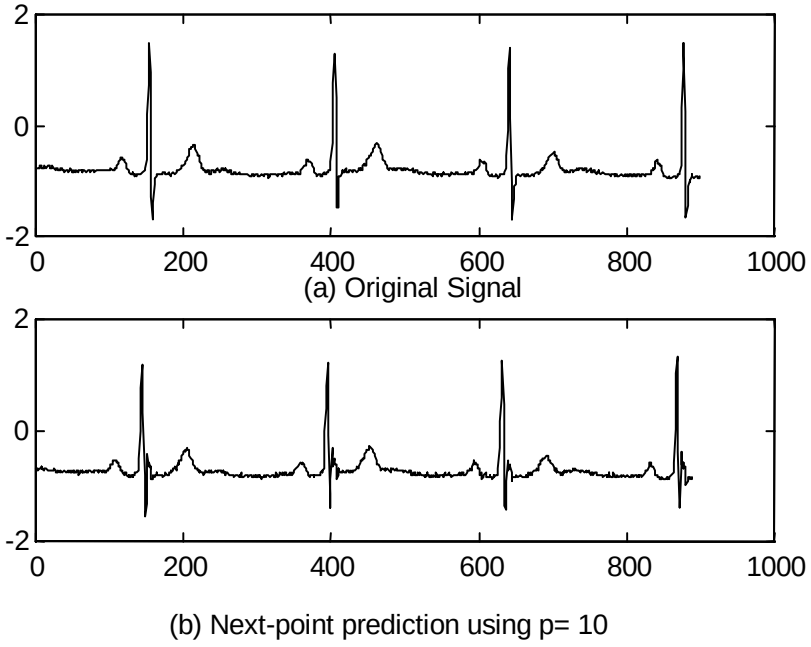


Figure 3.2. An example of next-point prediction of an ECG signal.

Figure 3.3 (b) shows the results obtained by the same predictor when working as a point-by-point predictor and trained with *p=200*. Two hundred points of the original signal were given, and 50 points ahead were predicted. To predict point 201, 200 original points

were given as external inputs. Feedback of the calculated values was required to predict from point 202 ahead. For prediction of the last point, 151 original points and 49 predicted values were used. Point by point prediction was not successful after just a few points.

3.3 Non-linear Prediction

A non-linear model can not be described by a transfer function, but a model equivalent to the linear case can be constructed for non-linear systems using the embedding theory of Takens (Príncipe 1997). This model is designed by first reconstructing an embedding space from the time series with characteristics equivalent to the original system (see section 2.6) and then defining a map that transforms from the current reconstructed state of the trajectory to the next state.

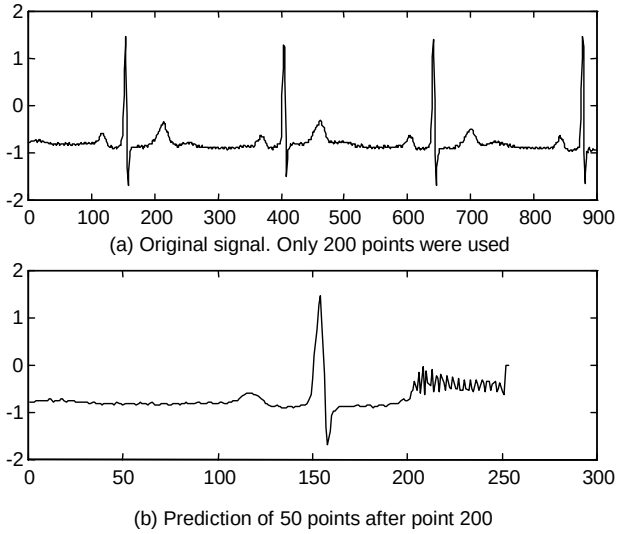


Figure 3.3. An example of point-by-point prediction of an ECG signal

Given $\mathbf{y}(n) = (x(n), x(n+\tau), x(n+2\tau)...x(n+(d-1)\tau)$ (where $x(n)$ is the time series, d is the embedding dimension, and τ the time lag), a map $\mathbf{F} : \Re^{\mathbf{d}} \rightarrow \Re^{\mathbf{d}}$, parameterized by $\mathbf{a} = (a_1, a_2, a_3...a_p)$, can be constructed such that

$$\mathbf{y}(n+1) = \mathbf{F}(\mathbf{y}(n), \mathbf{a}). \tag{3.16}$$

As in the linear case, the estimation of coefficients a_i is carried out such that a cost function like:

$$E(\mathbf{a}) = \sum_{n=1}^{N-1}\left[\sum_{m=1}^{d}\left[y_m(n+1) - F_m(\mathbf{y}(n),\mathbf{a})\right]^2\right], \tag{3.17}$$

is minimized.

The map $\mathbf{F}$ is required not only to produce accurately $\mathbf{y}(n+1)$ from $\mathbf{y}(n)$, but also to produce $\mathbf{y}(n+2)$ after two applications of the map to $\mathbf{y}(n)$, $\mathbf{y}(n+3)$ after three and so on.

As pointed out at section 2.7, data resulting from a non-linear dynamical system contain invariant information that is essential to describe the geometrical structure of its attractor. A way to construct a predictor map is calculating such invariants from the data and then imposing them as constraints on the calculation of parameters, in a way that the dynamical system defined by $\mathbf{F}$ has similar invariants to the unknown system. Abarbanel et al. (1990) constructed a predictor for chaotic series based on this concept obtaining good results for data coming from the Hennon map and the Lorenz system. They found that the parameter values that minimize the least-square criterion do not in general reproduce the invariants of the dynamical system, while the maps that reproduce the values of the invariants are not optimum in the least-square sense.

It is important to remark that reliable point-to-point prediction of chaotic systems with unknown dynamics is impossible (Wang and Alkon 1993). Indeed, there is not a theory for recognizing whether a constructed predictor has been able to truly identify the original system.

3.3.1 Neural Networks for Non-linear Prediction

Applications of artificial neural networks are countless. Both feed-forward and recurrent neural networks have proved to be well suitable for many problems where approximations to functions are needed; also several models have been designed to emulate oscillations and other time-dependent sequences.

There are several studies related to modeling and prediction of non-linear time series using neural networks. Wang and Alkon (1993) present a good summary of some of these studies. Recurrent neural networks have shown to be crucial for activities involving non-

linear dynamics and specially for chaos. Following is a brief description of some of these models.

3.3.1.1. The Oscillatory Network of Hayashi

Hayashi (1994) analyzed the behavior of an oscillatory network with external inputs. His network is made of excitatory and inhibitory neural groups. Each excitatory cell is connected to an inhibitory cell and to other excitatory cells. The dynamic equations for the cells are:

$$\dot{x}_i = -x_i + G(\sum_{j=1}^{N} W_{ij}x_j - K_{EI}^{(i)} y_i + I_i), \tag{3.18}$$

$$\dot{y}_i = -y_i + G(K_{IE}^{(i)} x_i), \tag{3.19}$$

$$G(z) = \frac{2}{\pi}\arctan(\frac{z}{a}), \tag{3.20}$$

$$W_{ij} = \frac{1}{N}\sum_{\alpha=1}^{M} \xi_i^{\alpha}\xi_j^{\alpha} + \delta_{ij} \quad (\mathrm{N} \geq \mathrm{M}), \tag{3.21}$$

where x_i and y_j are the averaged pulse density for excitatory and inhibitory cells, N represents the number of node groups, M is the number of memory patterns, $\xi_i^{\alpha}(\alpha = 1,2 \ldots \mathrm{M})$ are the training patterns and δ_{ij} is the Kronecker's delta. Hayashi observed that, when the external inputs to the network were similar to a memory pattern, the network generated a limit cycle near such a pattern. For an input far from the memory patterns, a chaotic orbit was generated.

3.3.1.2 Real Time Recurrent Learning

Williams and Zipser (1989) proposed a network called Real Time Recurrent Learning (RTRL). This learning algorithm is based on minimization of error, as in Back Propagation Through Time (BPTT). The authors claim that the principal advantage of their algorithm over BPTT is that it does not require an epoch length. The computation time of this algorithm is of $O(n^4)$. The dynamics of their system is represented by the equations:

$$S_k(t) = \sum_{l \in U} w_{kl} Y_l + \sum_{l \in I} w_{kl} X_l = \sum_{l \in U \cup I} w_{kl} Z_l(t), \tag{3.22}$$

$$Y_k(t+1) = f_k[S_k(t)]\,. \tag{3.23}$$

There are *n* nodes and *m* external input lines per node. $\overline{X}(t)$ is an *m*-tuple of external inputs at time *t*, $\overline{Y}(t)$ is an *n*-tuple of outputs at time *t*, U represents the set of outputs and I represents the set of inputs. The vector **Z** is defined as:

$$Z_k(t) = \begin{cases} X_k(t) & \text{if } k \in I \\ Y_k(t) & \text{if } k \in U \end{cases}\,. \tag{3.24}$$

William and Zipser also proposed the concept of *teacher forcing,* which basically consists of using the desired value of the signal, if available, instead of the actual output $Y_k(t)$ when computing the rest of the outputs of the network. Teacher forcing has shown to improve learning in some applications but not in all cases.

3.3.1.3 Time Delay Neural Network with Global Feedback Loop

Príncipe and Kuo (1994) studied a dynamic modeling of chaotic time series using a recurrent neural network with a global feedback loop. Their network was trained using back- propagation through time. They proposed to use dynamic invariants as a measure of the success of the predictor, instead of a global error. This network is called the Time Delay Neural Network with Global Feedback Loop (TDNNGFL). The dynamic net is seeded with a set of input samples; next, the input is disconnected and the predicted sample is fed back to the input for *k* steps. The mean square error between the predicted and true sample is used as a cost function. Because the time series is chaotic, the authors weighted the error function according to the largest Lyapunov exponent of the signal. The cost function used in this network is:

$$E = \sum_{j=0}^{r} \sum_{i=2m+1}^{k} h(i) \cdot dist(x(i + jq + 1) - \tilde{x}(i + jq + 1))\,, \tag{3.25}$$

where *r* is the number of training sequences, *m* is the estimated dimension of the dynamic system, *k* is the length of the trajectory, *q* is the number of samples that overlap the sequences of length *k, x* is the real data and $\tilde{x}$ is the predicted data. The function *h* is defined as:

$$h(i) = (e^{\lambda \max \Delta t})^{-(i-2m-1)}\,, \tag{3.26}$$

where λ_{max} is the largest Lyapunov exponent and Δt is the sampling interval. Using this cost function, the weighting of errors for later iterations has less credit than in former iterations.

3.3.1.4 The Complex Network

A 3-node fully-connected recurrent neural network (RNN) is able to oscillate, hence it may capture the dynamics of sine waves and work as an autonomous predictor. Once trained, these kinds of networks can accurately predict point-by-point fairly well during long periods of time, using no external inputs except the initial point of the signal (Figure 3.4). We call this a *harmonic generator*.

Based on this oscillation ability, Oldham (1997) developed a predictor model which consists of a fully-connected RNN pre-loaded with information about the Fourier components of the signal to be learned. Such a model is known as the complex network.

The frequency information of the signal is embedded in the network in the following fashion: the seven first sine harmonics of the signal to be modeled are used to train the weights of seven three-node recurrent networks, called "the sub-networks" or harmonic generators. After training, these sub-networks are embedded in a bigger RNN, the complex network.

Figure 3.5 shows a complex network with 3 harmonic generators, (named *A, B* and *C*). *No* is the output node that produces the predicted signal. The sub-networks are connected in a fully recurrent fashion to every other node in the complex network, but, for simplicity, not all connections are drawn in the figure. The internal weights of the harmonic generators are held fixed during the training of the rest of the weights. Bold lines in the figure represent these fixed weights. There are no nodes with external inputs; therefore, the network acts as an autonomous predictor. The signal is fed to the network during training as the desired value of node *No*. The model also may include other "output nodes," which are trained to learn the value of the signal in delayed times (S_0 and S_1 in the figure). These are called "pseudo-output" nodes. The rest of the nodes are hidden nodes. This network is trained using an implementation of the algorithm back-propagation through time developed by Pearlmutter (1989). Appendix B contains a detailed description of this algorithm.

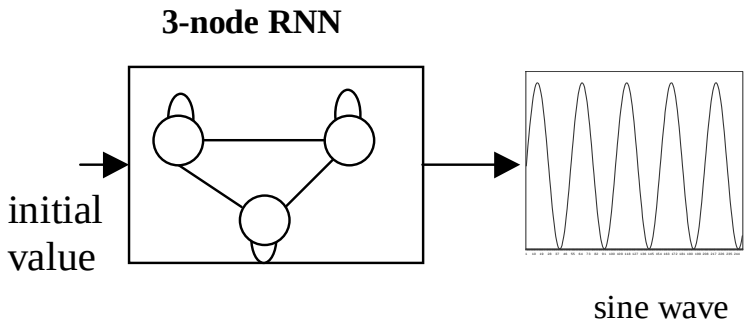


Figure 3.4. A harmonic generator: an autonomous predictor of sine waves.

Figure 3.6 shows the results obtained by a complex network trained to model an ECG signal. The network has a total of 33 nodes; 4 of them are pseudo-output nodes, one is the output node and 21 correspond to 7 sub-networks trained to the first seven harmonics of the original signal. This network has 1,026 modifiable weights out of 1,089 total.

The training time series has 512 points corresponding to approximately 2 beats or cycles of the ECG. The point-by-point prediction is carried out up to point number 1,500. Both the original and predicted signals are shown in Figure 3.6; the original signal ends at point 512. Figure 3.7 shows the corresponding return map. These results were obtained after 31,000 epochs, obtaining a final mean square error of 0.0071. More training did not result in any significant improvement in the error. Notice that the highest output value obtained by the network is around 0.2 while the original signal has a maximum at 0.75.

Figure 3.8 shows the prediction obtained when the harmonic weights were allowed to be modified, and the network trained 15,000 more epochs. The final mean square error by point was 0.0068. These results are similar to the ones showed at Figure 3.6.

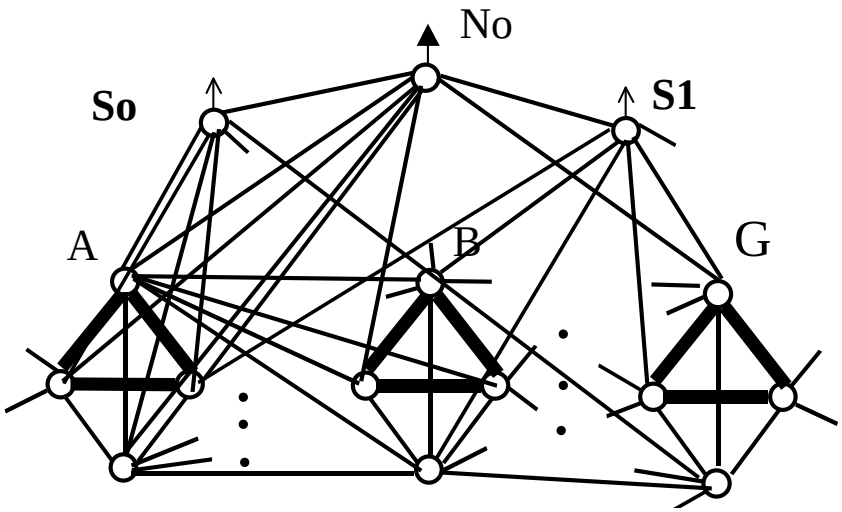


Figure 3.5. The complex network. A, B and C are harmonic generators; No. is the output node.

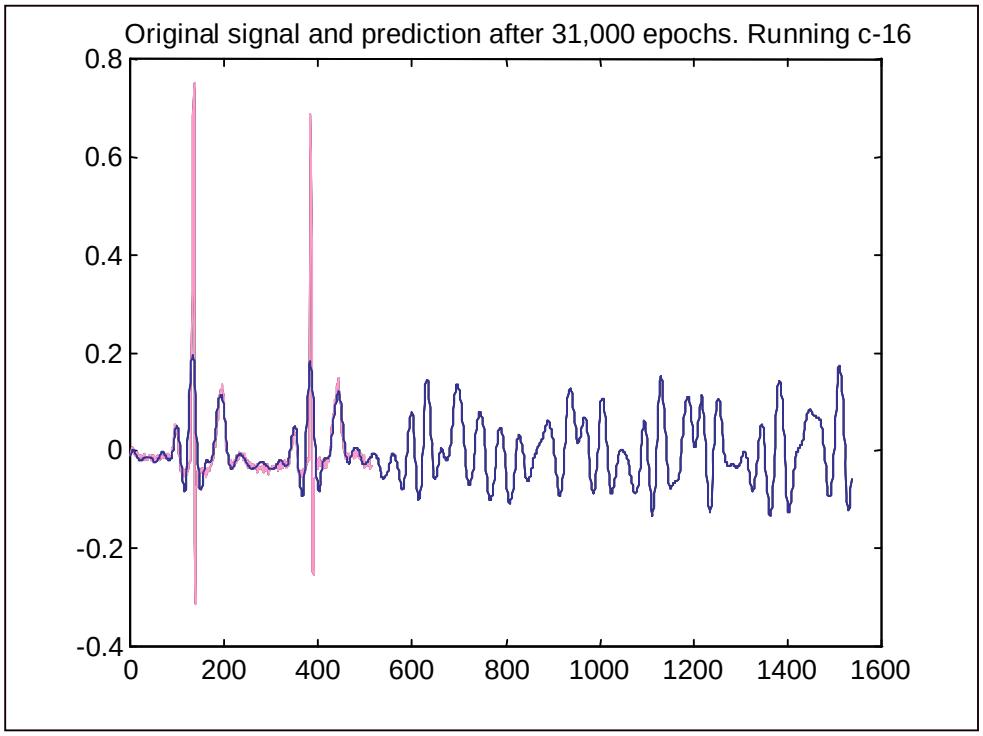


Figure 3.6. A prediction obtained by a complex network after 31,000 epochs.

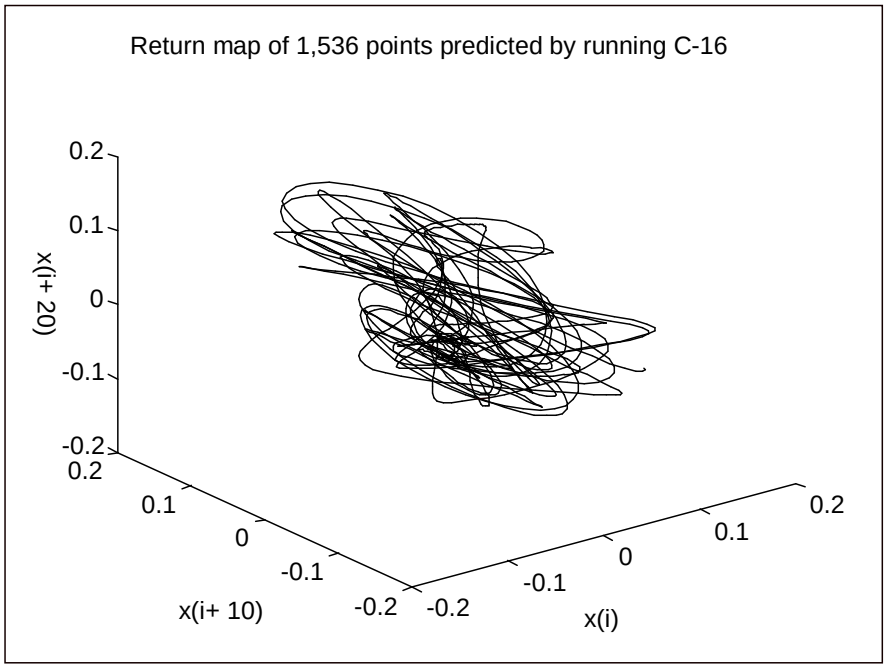


Figure 3.7. Return map generated by the prediction shown at Figure 3.6

Figure 3.9 shows a surface plot of the 33-by-33 weight matrix of the network after 31,000 epochs. This matrix $\mathbf{W} = w_{i,j} \;\; i, j = 0...32$ contains the connections from node *i* to node *j*. Nodes are ordered as follows: the first 21 nodes belong to sub-networks; the next 7 are hidden nodes; the next 4 are pseudo-output nodes; and the last is the output node. Notice in the figure that most of the activity is found at the weights located in the inverted diagonal, which correnspond to the 3-node sub-networks. Almost no activity is shown in the connections corresponding to recurrence among the hidden nodes.

Even though this point-to-point prediction is not satisfactory, the network shows an oscillatory, aperiodic output for large periods of time.

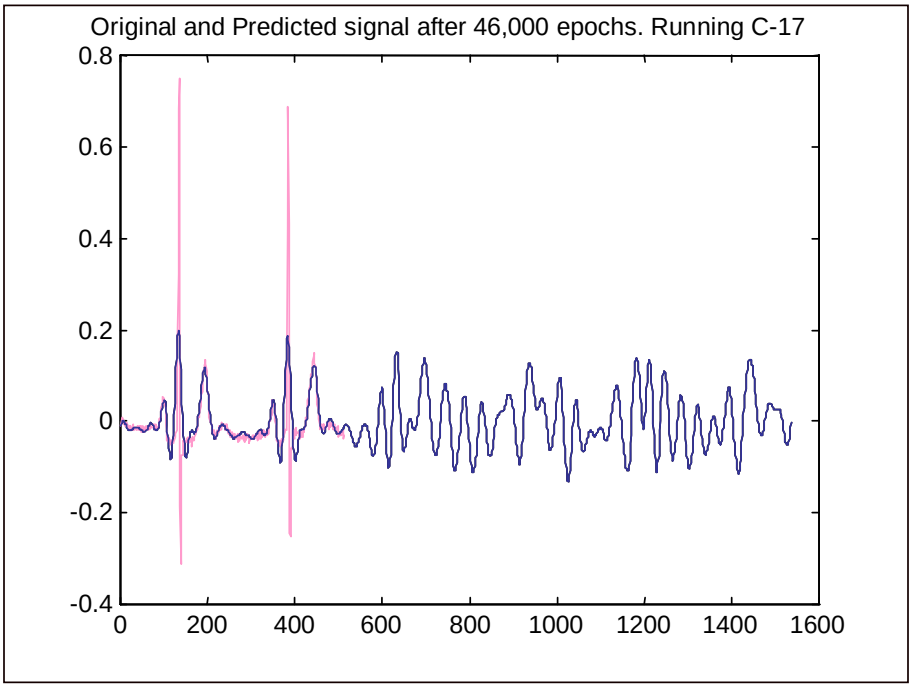


Figure 3.8. Prediction after 46,000 epochs training all weights

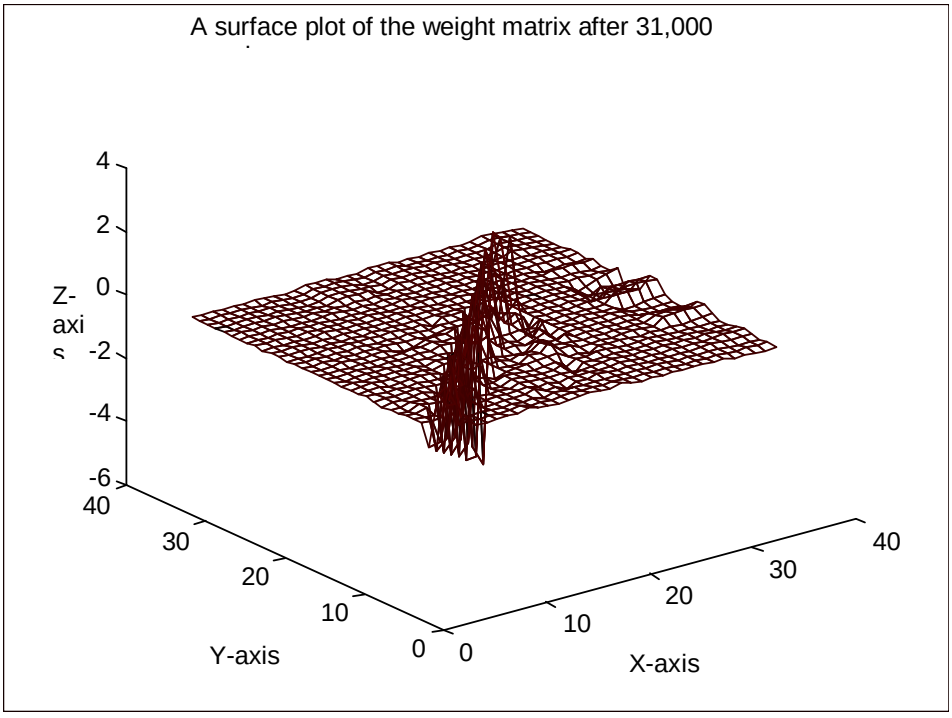


Figure 3.9. The weight matrix after 31,000 epochs.

CHAPTER IV
PERFORMANCE METRICS AND PREPROCESSING OF TRAINING SIGNALS

This chapter presents the methodologies used to measure and compare the performance of the predictors constructed in this work. It also includes a brief description of the pre-processing algorithms that were applied to the training signals before using them to feed the predictors. See also Appendix D for a listing of these metric values for all the training signals used in this research.

4.1 Methodologies to Evaluate the Predictors.

The performance of each predictor was measured using one or more of the following: the Mean Squared Error, the maximum Lyapunov exponent of the output signal and a visual inspection of the return maps generated by the output signal. Next is a description of such metrics.

4.1.1. Mean Squared Error.

The *Mean Squared Error (MSE)* generated by the a neural net predictor is calculated as:

$$MSE = \frac{1}{S}\sqrt{\sum_{t=0}^{S-1}\left(y_n(t) - d_n(t)\right)^2}\ , \tag{4.1}$$

where S is the size of the trajectory being evaluated; $y_n(t)$ is the output of the node predicting the signal; and $d_n(t)$ is the desired output of the predictor.

Even though MSE is a popular metric in approximation problems, it is not considered the best performance measure for modeling of chaotic systems (Príncipe & Kuo 1994, Abarbanel et al. 1990). A chaotic system may generate an infinite number of trajectories depending upon its initial conditions; therefore, even when the predictor is learning the correct information about the dynamic system, it is possible that it is not reproducing the same trajectory that was used for training. For this reason, this metric should be used with discretion.

4.1.2. Lyapunov Exponents

The *Lyapunov exponents (LE)* are invariant measures of non-linear systems; they do not change when the initial conditions of the trajectory are altered or due to small perturbations (see section 2.7 for a full description of Lyapunov Exponents). It is expected that a predictor map, if getting the essential characteristics of a dynamic system, will reproduce signals with invariants similar to the training signal, even if its output signal does not look similar to the training signal.

The method proposed by Wolf et al. (1985) was used in this work to calculate the maximum LE of the signals generated by the predictors (see section 2.7.1 and Appendix A for details). As pointed out in section 2.7.1, this numerical method needs to be used with caution, because its correct convergence is very sensitive to input parameters, noise and the amount of data available. This problem is a characteristic of most of the methods currently available to calculate LE from data.

4.1.3 Return Maps

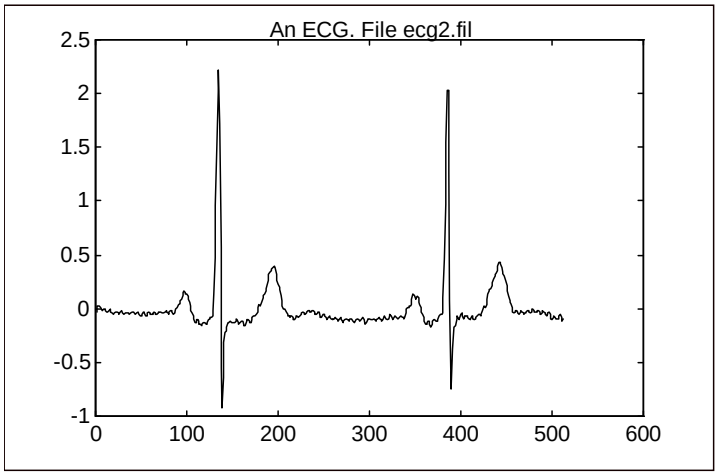
A *return map* is a plot representing the state space of the system (section 2.1 for a detailed explication of return maps). Chaotic signals generate return maps with well-defined but strange attractors, while random signals generate return maps without a defined shape. The return maps of chaotic signals show parts where trajectories are infinitely close to each other, known as *injection regions*. Return maps may be considered an invariant characteristic of chaotic time series, because different trajectories of the same dynamic system generate similar return maps.

Figure 2.7 shows a time series generated with the Mackey-Glass equation (equation 2.8), and Figure 2.8 shows its corresponding three-dimensional return map. Figure 4.1 (a) shows an electrocardiogram and 4.1 (b) its corresponding return map.

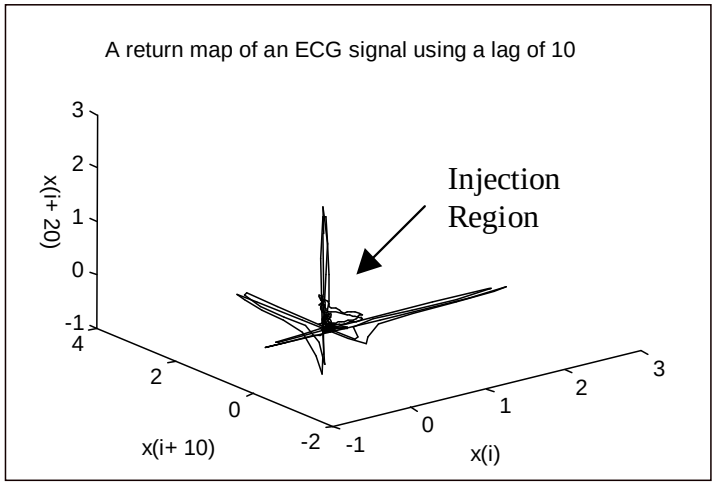
4.2 Preprocessing of Training Signals

The signals input to the predictors were modified for at least one of the following procedures: normalization of their amplitudes to get a mean equal zero or a specific maximum and minimum value; filtering of the signals to a specific frequency bandwidth; and conversion of the signals to a smaller sampling rate. All of these procedures were applied in order to simplify the work of the predictors. Also, the harmonic components for some signals were

calculated in order to use this information in the design of some predictors. Following is a brief explanation of how some of these procedures were carried out.



(a)



(b)

Figure 4.1. Return map of an electrocardiogram. (a) An ECG and (b) its corresponding return map with a time lag = 10.

Appendix D contains the return maps of all the signals used to train the experiments described at Chapter 5. All of these return maps were calculated using a time lag of 10.

4.2.1 Filtering

Filtering is the process of converting a signal to another signal where a range of frequencies has been removed. Following the recommendations of Abarbanel et al. (1993), a band-pass filter with finite-duration impulse response (FIR filter) was applied to the signal in order to reduce noise (see Proakis and Manolakis (1996) for a detailed description of FIR filters).

Electrocardiograms are signals with a broadband Fourier spectrum. Figure 4.3 shows the frequency spectrum of the ECG shown at Figure 4.2, which was sampled to 360 Hz. This signal will be referred as *ecg2.dat* (see Appendix D for a full description of all signals referred to in this chapter).

Ecg2.dat was filtered by a FIR filter of order 40 with cutoff frequencies lying at 0.5 and 105 Hz. This upper limit was chosen after observing that, in ecg2.dat, the magnitude of frequencies above 100 Hz. is less than 0.5. The filtered signal, called *ecg2.fil*, is shown at Figure 4.4; Figure 4.5 shows its frequency spectrum. The fundamental frequency of both ecg2.dat and ecg2.fil is 1.4062 Hz.

Figure 4.6 shows the signal ecg4.fil, which is the result of normalizing the magnitudes of ecg2.fil to a minimum of –0.3122 and a maximum of 0.75.

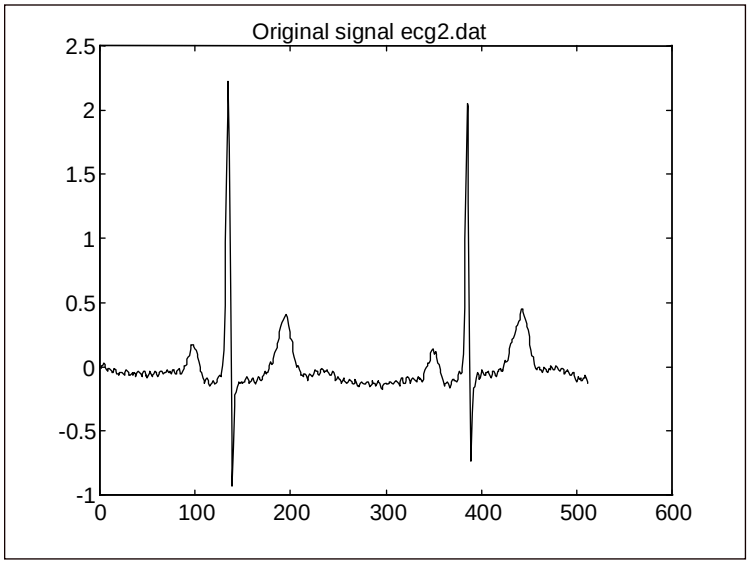


Figure 4.2. An electrocardiogram (ecg2.dat) .

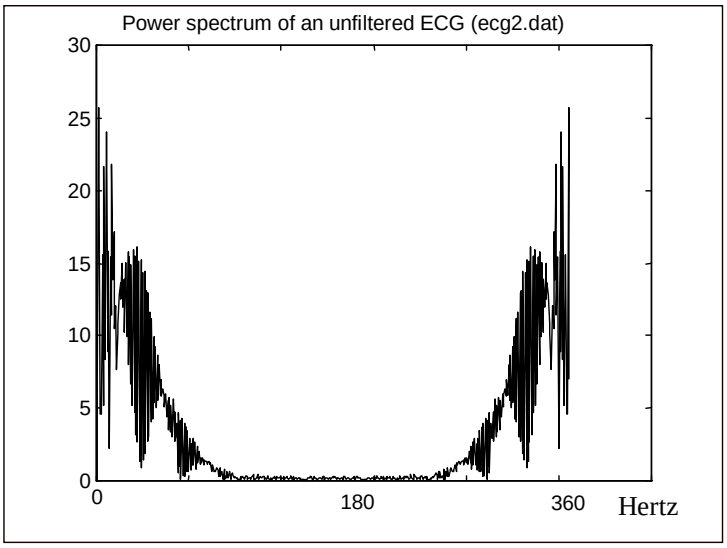


Figure 4.3. The frequency spectrum of ECG in Figure 4.2.

4.2.2 Sampling Rate Conversion

Sampling Rate Conversion is the process of changing a signal from a given sampling rate F_x to another sampling rate F_y. The ratio $\frac{F_y}{F_x}$ should be rational when using a digital method for the conversion. If the sampling rate is reduced the process is called *decimation*.

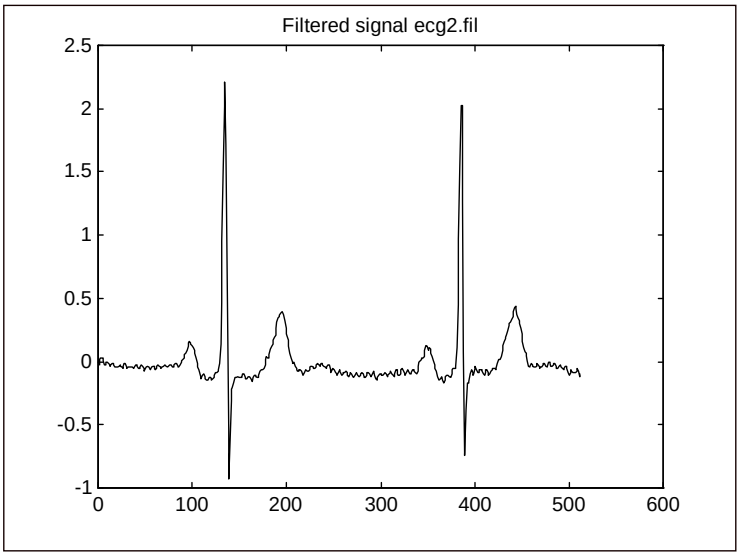


Figure 4.4. A filtered version of ecg2.dat (Ecg2.fil)

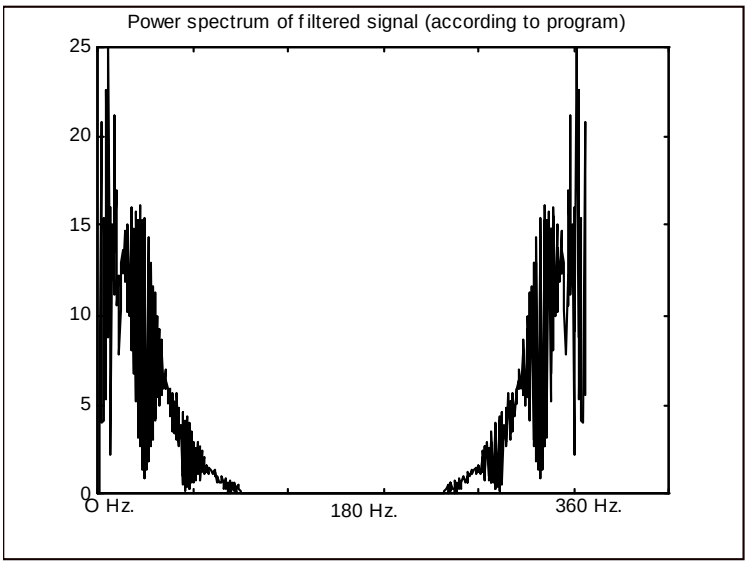


Figure 4.5. The frequency spectrum of ecg2.fil

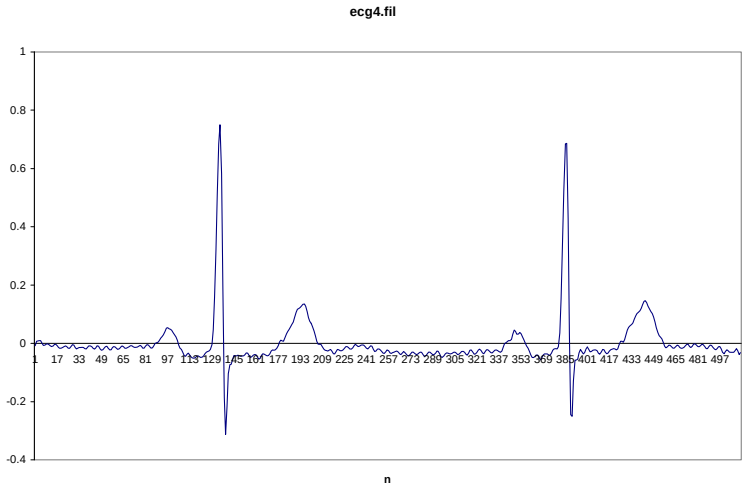


Figure 4.6 Ecg4.fil: the signal ecg2.fil normalized in magnitude to the range [-0.3122,0.75]

When decimating a signal *x(n)* with spectrum X(w) by an integer factor of D, the bandwidth of *x(n)* must first be reduced to:

$$F_{MAX} = \frac{F_x}{2D}, \tag{4.2}$$

to avoid the phenomenon of aliasing in the resulting signal (Proakis and Manolakis 1996).

To decimate by a factor of 4 the time series ecg2.dat (Figure 4.2) which was originally sampled to 360 Hz., the signal needs to be filtered to :

$$F_{MAX} = \frac{F_X}{2D} = \frac{360}{2*4} = 45 \text{ Hz.},$$

before the down sampling. Figure 4.7 shows the resulting signal, called ecg6n.su4. Ecg2.dat had 512 points; ecg6n.su4 contains 128 points.

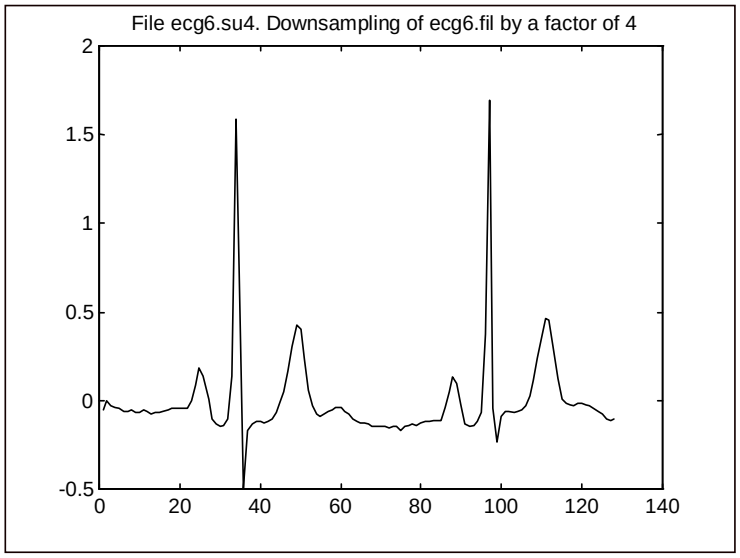


Figure 4.7. Ecg6n.su4: ecg2.dat converted to a sampling frequency of 90 Hz.

4.2.3. Harmonic Decomposition

Spectral decomposition of a discrete-time aperiodic signal is carried out using the *Discrete Fourier Transform (DFT*), defined for a signal $x(n)$ with N components as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad \text{for k=0, 1, 2,... } N - 1. \tag{4.3}$$

Its corresponding *Inverse* (IDFT) is defined by:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{-j2\pi kn/N} \qquad \text{n= 0, 1, 2,...N-1.} \tag{4.4}$$

The magnitude of *X(k)* is called the *Power Spectrum Density* (PSD) of the signal.

Knowing the fundamental frequency of a signal, its harmonic components are defined as sine and cosine functions with their frequencies being a factor of the fundamental. The amplitudes and phases can also be obtained from the PDS. The addition of all harmonic components of a signal will lead to the exact signal if the number of frequency components is infinite. Figures 4.8 shows the signal obtained when adding 30 harmonics of ecg2.fil. Notice that this figure roughly resembles the original signal; therefore, modeling ECG signals using harmonics is not accurate enough to allow any prediction, unless that the number of harmonics were large.

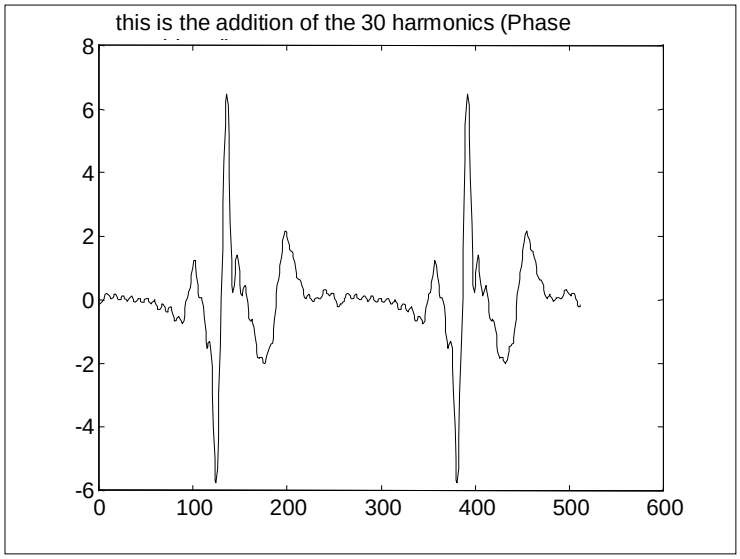


Figure 4.8. Approximation of ecg4.fil using 30 harmonics.

CHAPTER V
RESULTS

This chapter describes the results obtained by several neural-net predictors built applying some of the ideas detailed in previous chapters. Four kinds of predictors were analyzed: modifications to the complex network model, predictors based on hybrid neural networks, predictors based on feed-forward networks with feedback, and predictors based on hybrid networks using harmonic generators and external inputs. Other experiments were run, but these are used to summarize the results.

All the neural networks (feed-forward, hybrid and recurrent types) were trained using an implementation of the algorithm back propagation through time described by Pearlmutter (1990); Appendix B contains a detailed description of this algorithm. In all cases, the training processes were stopped when the decrement in the cost function during the last 100 epochs became less than 1.0E-05.

All programs were implemented using the programming language C++. Some of the pre-processing software was developed using MatLab™. The electrocardiograms used in the experiments were taken from the database produced by the Harvard-MIT Division of Health Science and Technology Biomedical Engineering Center (Harvard 1992). Appendix D contains a full description of all signals used in the experiments. This includes values of their maximum Lyapunov Exponents, and plots of the time series, Fourier transforms and return maps.

Table 5.1 describes at-a-glance each case; Table 5.2 contains a detailed description of the input parameters used for training each predictor; Table 5.3 shows the calculations of the performance metrics maximum Lyapunov Exponent (LE) and Mean Squared Error (MSE) for most of the cases. In that table, as well as in other places, the term *segment* is used to refer a trajectory of predicted points with the same size as the trajectory used to train the network. Appendix C contains the connection matrices that fully specify the topology of the hybrid and feed-forward networks used in some of the experiments. The results obtained by the original complex model, described at section 3.3.1.4, will be referred to as Case C.0 in this and next chapters. Case C.0 is used as a comparison point for some of the cases reported in this chapter.

Table 5.1 A summary of the cases analyzed in this work.

CASE NUMBER	DESCRIPTION
C.0	The original complex network (section 3.3.1.4): a fully recurrent network with harmonic generators. No external inputs.
C.1	The same complex network defined at C.0 but using a down-sampled signal (section 4.2.2) for training.
C.2	The same complex network model defined at C.0, adding teacher forcing (section 3.3.1.2) during training.
C.3 A and B	The complex network defined at C.0, adding time constants weights (section 4.3.3). Part A used the same signal as C.0 for training; part B used a down-sampled version of the signal.
H.1	A hybrid network using harmonic generators and pseudo-output nodes, with no recurrent connections except in the harmonic generators. The topology of this network is defined in appendix C by the connection matrix A.
F.1 A and B	A feed-forward network that uses feedback during its prediction state. Its topology is defined by connection matrix B in appendix C. Part A predicts a sine function; part B predicts an electrocardiogram.
F.2	Addition of recurrent connections to the network used at case F.1.
K.1	A hybrid network built with harmonic generators and external inputs, that uses feedback during prediction. No hidden layer is included. Connection matrix C defines its topology.
K.2 A and B	A hybrid network built with harmonic generators and external inputs, that uses feedback during prediction. A hidden layer is included. Connection matrix D defines its topology. Part A predicts an ECG; part B predicts Mackey-Glass data.

Table 5.2 Specifications of the neural net predictors implemented for each case.

CHARAC-TERISTIC	CASE C.0	CASE C.1	CASE C.2	CASE C.3.A	CASE C.3.B	CASE H.1
Type of connections	Fully recurrent	Fully recurrent	Fully recurrent	Fully recurrent	Fully recurrent	Hybrid
I.D. of connection topology (appendix C)	--	--	--	--	--	A
Total number of weights to modify.	1,089	1,089	1,089	1,089+33= 1,122	1,089+33= 1,122	245+33 = 278
Total number of nodes	33	33	33	33	33	33
Number of input nodes	0	0	0	0	0	0
Number of output nodes	5	5	5	5	5	5
Sigmoid coefficient	0.5	0.3	0.3	0.1	0.3	0.1
Delta t	0.1	0.6	0.6	0.3	0.1	0.3
Training file	Ecg4.fil	Ecg6n.su4	Ecg6n.su4	Ecg4.fil	Ecg6n.su4	Ecg4.fil
Size of trajectory	512	128	128	512	128	512
Number of harmonics	7	7	7	7	7	7
Total of epochs executed	31,000	31,000	31,000	61,000	61,000	45,000
Final MSE	7.1E-3	1.47E-2	1.51E-2	7.4E-3	1.59E-2	7.95E-3

Table 5.2 (continuation). Specifications of the neural net predictors implemented for each case.

CHARAC-TERISTIC	CASE F.1.A	CASE F.1.B	CASE F.2	CASE K.1	CASE K.2.A	CASE K.2.B
Type of connections	Feed-forward with feedback	Feed-forward with feedback	Feed-forward, then recurrent	Hybrid with external inputs	Hybrid with external inputs	Hybrid with external inputs
I.D. of connection topology (appendix C)	B	B	B	C	D	D
Total number of weights to modify.	60	60	First 60; next 256	First 189+27 = 216; next 729+27 = 756	First 154+34 = 188; next 1,156+34 = 1,190	First 154+34 = 188; next 1,156+34 = 1,190
Total number of nodes	16	16	16	27	34	34
Number of input nodes	5	5	5	5	5	5
Number of output nodes	1	1	1	1	1	1
Sigmoid coefficient	1.0	1.0	1.0	0.1	0.1	0.5
Delta t	1.0	1.0	1.0	0.3	0.3	0.5
Training file	H360_7_5.dat (Sine)	A0310z.fil (ECG)	A0310z.fil (ECG)	Ecg4.fil	Ecg4.fil	Good8. dat (Mackey)
Size of trajectory	104	475	475	512	512	210
Number of harmonics	0	0	0	7	7	7
Total of epochs executed	10,000	50,000	100,000	50,000	30,000	31,000
Final MSE	2.0E-3	1.4E-3	8.4E-4	3.10E-3	2.35E-3	8.82E-1

Table 5.3 Performance metrics MSE and LE for selected cases.

CHARAC-TERISTIC	CASE C.0	CASE C.1	CASE C.3.	CASE H.1	CASE F.1B	CASE F.2	CASE K.1	CASE K.2.A	CASE K.2.B
Training file	Ecg4.fil	Ecg6n.su4	Ecg4.fil	Ecg4.fil	A0310z.fil	A0310z.fil	Ecg4.fil	Ecg4.fil	Good8.dat (Mackey-Glass)
Size of trajectory	512	128	512	512	475	475	512	512	210
MSE after Training	7.1E-3	1.47E-2	7.4E-3	7.95E-3	1.4E-3	8.4E-4	3.10E-3	2.35E-3	8.8E-1
LE of training signal	3.23 ±0.27	10.77 ±2.94	3.23 ±0.27	3.23 ±0.27	19.34 ±5.25	19.34 ±5.25	3.23 ±0.27	3.23 ±0.27	0.0334 ±0.003
LE of first predicted segment *	6.53 ±4.54	38.64 ±10.53	7.76 ±2.09	12.53 ±0.85	19.7 ±5.12	18.05 ±5.66	4.47 ±0.33	4.88 ±2.21	0.0374 ±0.006
LE of first 4 predicted segments *	1.63 ±0.75	1.90 ±1.04	3.92 ±1.11	2.96 ±0.52	INFINITE	INFINITE	5.09 ±1.14	7.52 ±1.95	0.0845 ±0.005

* A *segment* is a trajectory of predicted points with the same number of points as the trajectory used to train the signal

5.1 Modifications to the Complex Model

Several modifications were tried for Case 0, in an attempt to improve its learning speed and prediction ability. These modifications include: training the network with a down-sampled signal, the use of teacher forcing during training, and adjusting the time-constant weights. Each case is described next.

5.1.1. Case C.1. Down-sampling of the training signal

The training algorithm backpropagation through time has a complexity of $O(n^2 \cdot s)$, where n is the number of nodes in the network and s is the size of the training set (trajectory). Therefore, a reduction in the size of the trajectory could lead to faster training, provided that no information is lost. Besides, it was hoped that removing noise and retraining only the important features of the signal would make it easier for the network to learn.

Considering this, the training signal used at case C.0 (ecg4.fil) was down-sampled by a factor of 4 using the procedures described at section 4.2. Then the experiment was repeated keeping the rest of parameters similar (see Table 5.2). After 31,000 epochs the MSE was 1.47E-2. This resulted in a value larger than the MSE obtained at case C.0 (7.1E-3).

Figure 5.1 shows the first segment of the predicted signal (128 points), obtained after that the training was stopped; and it compares it with the expected one. Figure 5.2 shows a plot of the long-term, point-by-point prediction obtained at this case; Figure 5.3 shows a return map of such a prediction.

The LE of the first predicted segment was 38.64 ± 10.53. This value is far away of the LE of the corresponding segment in the training signal, which is 10.77. The same difference is noticed with the LE of 4 segments of prediction, which was 1.90 ± 1.04. Therefore, comparing the MSE, the LE and the return maps of cases C.0 and C.1, no improvement is noticed due to down-sampling.

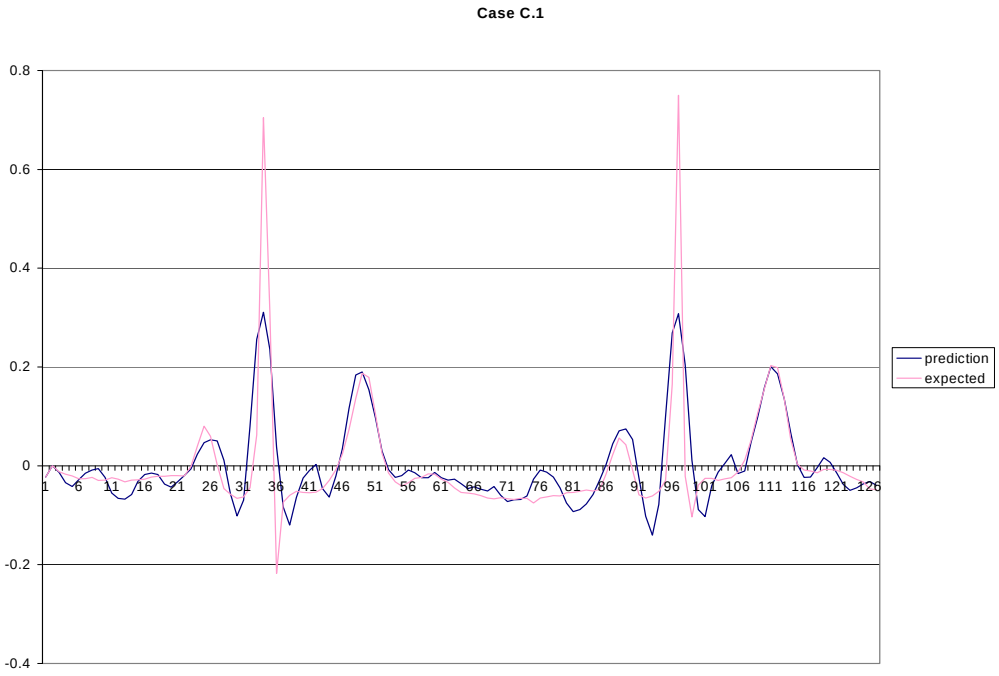


Figure 5.1. First 128 points predicted after learning in case C.1, compared with training signal. MSE = 7.1E-3.

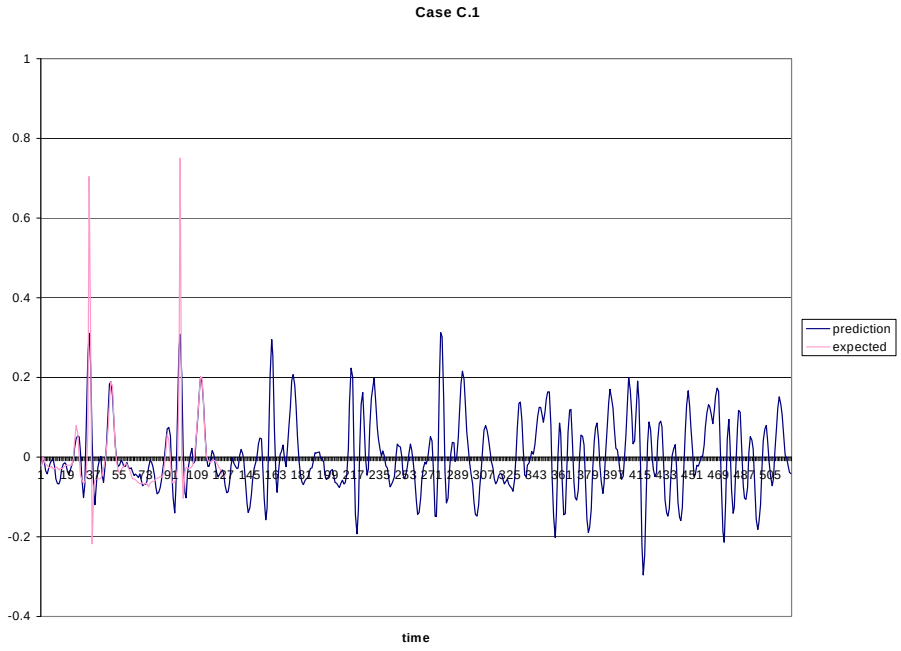


Figure 5.2. Prediction of 520 points obtained at Case C.1., compared with training signal.

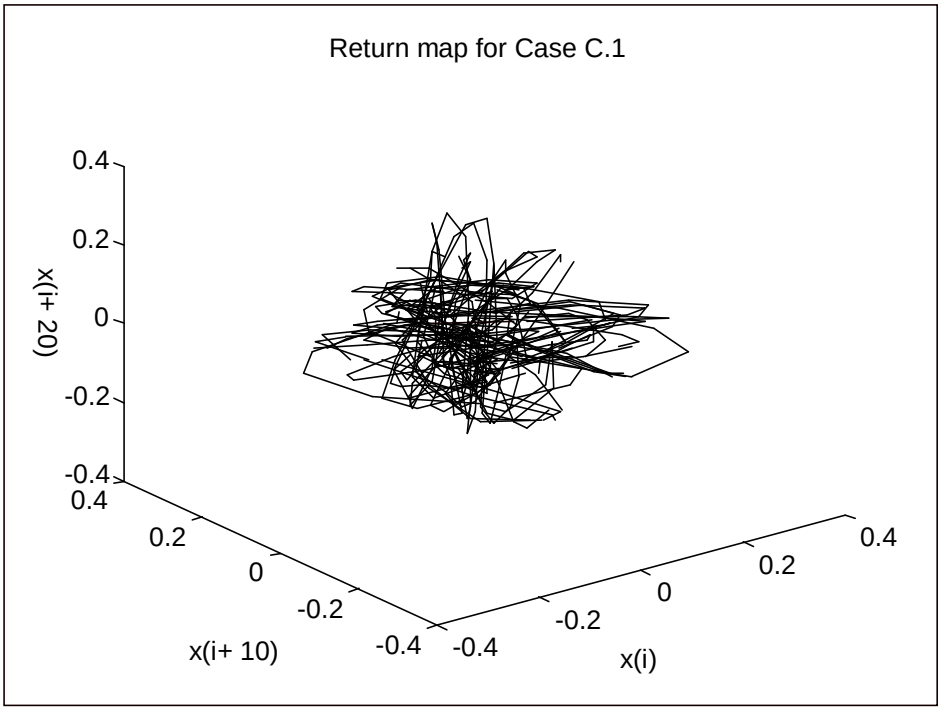


Figure 5.3. Return map of 4 segments of prediction generated at case C.1.

5.1.2 Case C.2. Training Using Teacher Forcing

As described in Appendix B, the dynamic of the neurons used at the complex network is defined by:

$$\frac{dy_i}{dt} = -y_i + \sigma(x_i) + I_i \,, \tag{5.1}$$

where:

$$x_i = \sum_j w_{j,i} y_j \,. \tag{5.2}$$

Williams and Zipser (1989) (section 3.3.1.2) proposed that during training, the calculation of the total input to neuron *i* (equation 5.2) should be carried out using the desired value for node *i*, if available, instead of the actual output of such node.

This change was implemented for the complex network described at case C.1 using the down-sampled signal ecg6n.su4 for training. After 31,000 epochs, the MSE was 1.51E-2, slightly larger than in case C.1 (1.47E-2) where the same signal was used.

Figure 5.4 shows the prediction of the first segment after training; Figure 5.5 shows the long-term prediction of 4 segments; and Figure 5.6 shows the return map generated by this prediction.

Comparing the MSE and return maps of cases C.1 and C.2, no improvement is noticed due to teacher forcing.

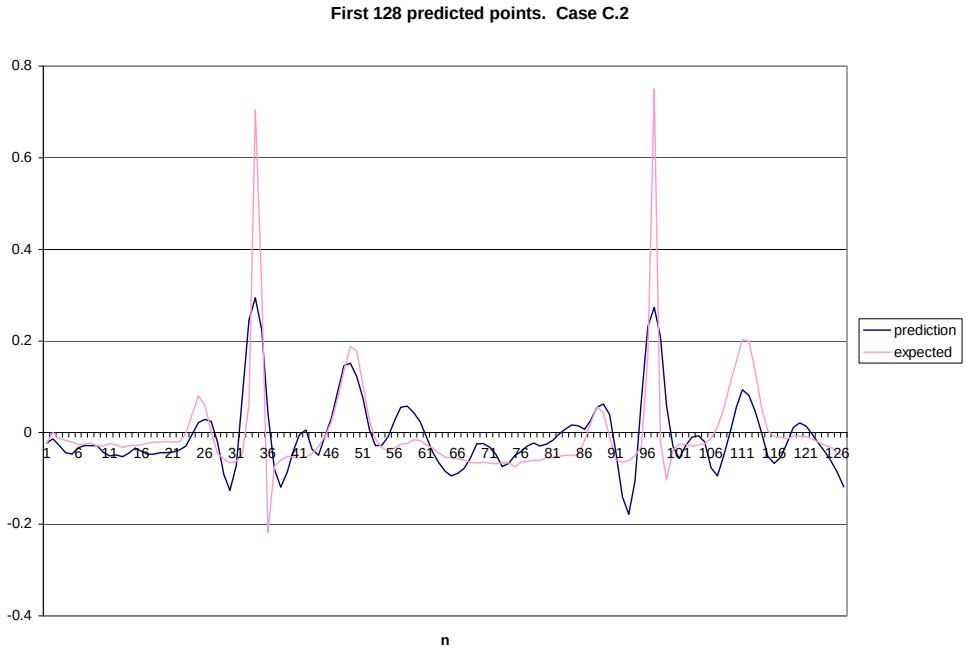


Figure 5.4. First 128 points predicted after learning for case C.2, compared with training signal. MSE = 1.47E-2.

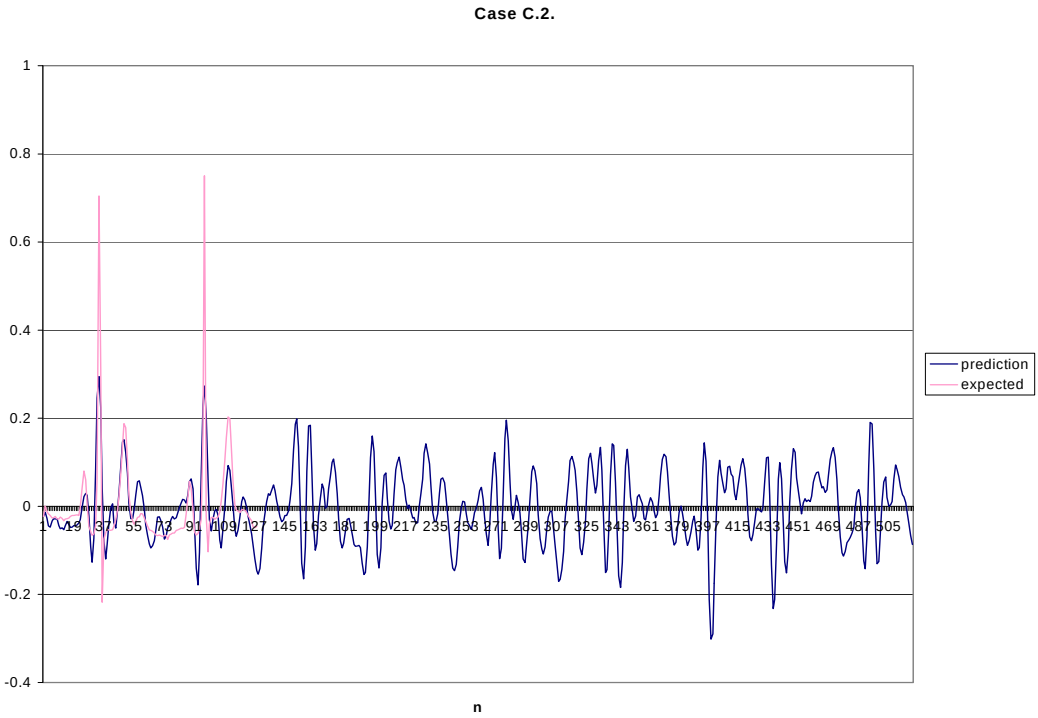


Figure 5.5. Prediction of 4 segments obtained at Case C.2, compared with training signal.

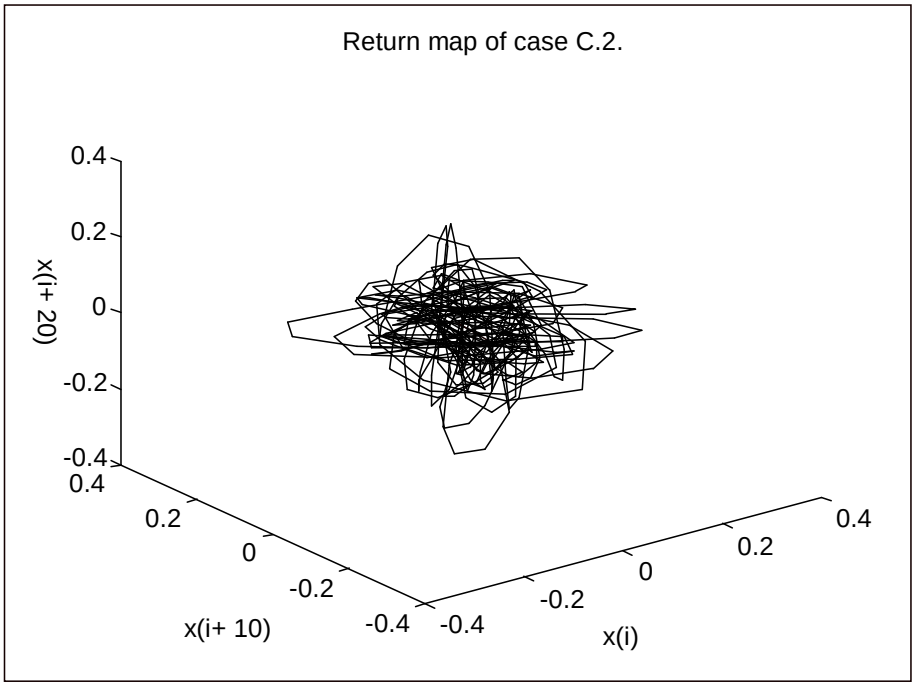


Figure 5.6 A return map generated by the prediction of case C.2.

5.1.3. Case C.3. Time-Constant Weights

Pearlmutter (1989) pointed out that the change of a neuron y_i over time (equation 5.1), can be driven by an adapting parameter called time constant. Equation (5.1) becomes:

$$T_i \frac{dy_i}{dt} = -y_i + \sigma(x_i) + I_i, \tag{5.3}$$

which, solving for y_i and written in discrete form is:

$$y_i(t) = \left[1 - \frac{\Delta t}{T_i}\right] y_i(t - \Delta t) + \frac{\Delta t}{T_i}\sigma(x_i(t)) + \frac{\Delta t}{T_i} I_i . \tag{5.4}$$

Using this dynamic, the equations to train the weights $w_{i,j}$ and T_i using back propagation through time (see Appendix B) are:

$$T[i] := T[i] - \frac{\eta \Delta t}{T[i]} \sum_t z[i][t] * \left[\frac{y[i][t] - y[i][t - \Delta t]}{\Delta t}\right], \tag{5.5}$$

$$w[i][j] := w[i][j] - \frac{\eta \Delta t}{T[j]} \sum_t y[i][t] * \sigma'(x[j][t]) * z[i][t], \tag{5.6}$$

$$z[i][t] = \Delta t e[i][t + \Delta t] + \left[1 - \frac{\Delta t}{T_i}\right] * z[i][t + \Delta t] + \Delta t \sum_j \frac{w_{ij}}{T_j} \sigma'(x[j][t + \Delta t]) z[j][t + \Delta t], \tag{5.7}$$

for all $i, j = 0$ to n-1; where n is number of neurons in the network. These changes were implemented for the complex network described at case C.0, using signal ecg4.fil (Experiment A) and signal ecg6n.su4 (Experiment B).

In experiment A, an MSE of 7.4E-3 was reached after 61,000 epochs. Figures 5.7 to 5.9 plot the obtained prediction of first segment, the prediction of four segments and the return map of the long-term prediction.

The MSE of C.3.A is slightly larger than the MSE of C.0 (7.1E-3). However, the attractor in the return map of case C.3.A (Figure 5.9) presents a better defined geometrical shape than the one in the return map of C.0 (Figure 3.7). Notice also in Figure 5.8 (a) that the time series of points 512 to 1,024, which correspond to the third and fourth segments of prediction, have a shape more similar to a ECG than the corresponding segments at case C.0 (Figure 3.6). This uniformity is also well noticed in the return maps of each segment:

Figures 5.10 (a) to (d) plot the return maps of segments of 512 points each, corresponding to the prediction obtained by C.3.A. Compare these figures with Figures 5.11 (a) to (d), which show the same segments obtained at case C.0. Notice that the fourth segment in both figures is completely different to the first segment, due to the error accumulated by the long-term prediction.

Figure 5.8 (b) shows a grid with vertical lines separated the same distance as the first R-peaks of the predicted signal. Notice that almost 5 of the R-peaks[1] of the prediction signal keep a constant time period; then the shape of ECG is lost.

The maximum LE of the first predicted segment was 7.76 $\pm$ 2.9; and the one corresponding to four segments was 3.92$\pm$1.11. The maximum LE of the training signal is 3.23 $\pm$ 0.27. When analyzing these numbers, case C.3.A. seems to have captured the dynamic of the system better than case C.0, given the fact that its LE for one and four segments are nearer to the expected value.

For case C.3.B where a down-sampled signal was used, an MSE of 1.59E-2 was obtained after 61,000 epochs. Figures 5.12 and 5.13 show the prediction of one and four segments. Figure 5.14 shows the corresponding return map. The return map does not present any uniformity in the geometrical shape of the attractor neither did the long-term prediction yield better results than in case C.0.

Therefore, the use of adaptive time constants in the complex model greatly improved the performance of its long-term prediction, when training was done using the original signal. However, it did not show any improvement when using a down-sampled signal.

[1] See Figure 1.1 to identify the peaks of an ECG

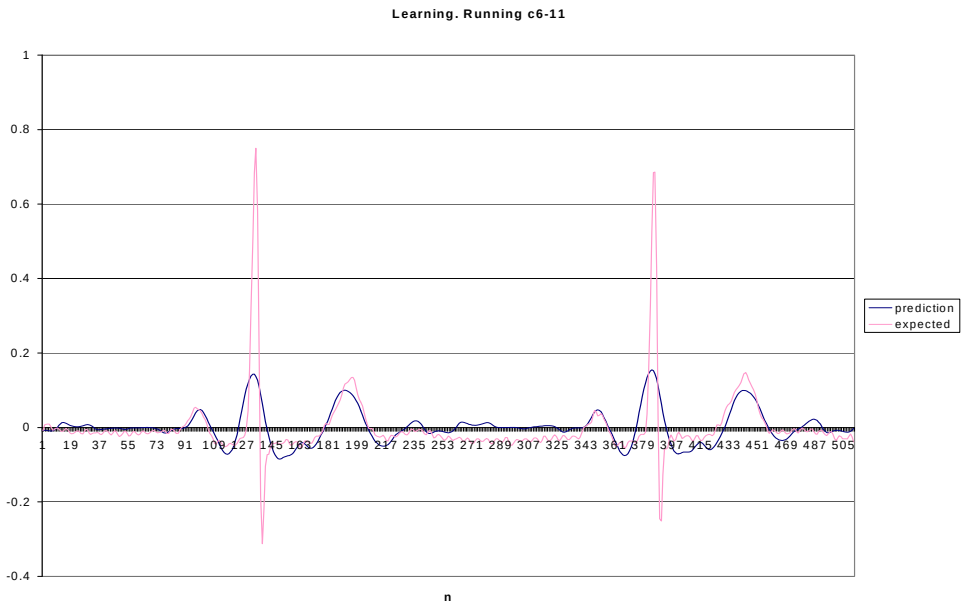


Figure 5.7 First 507 points learned for case C.3.A., compared with training signal

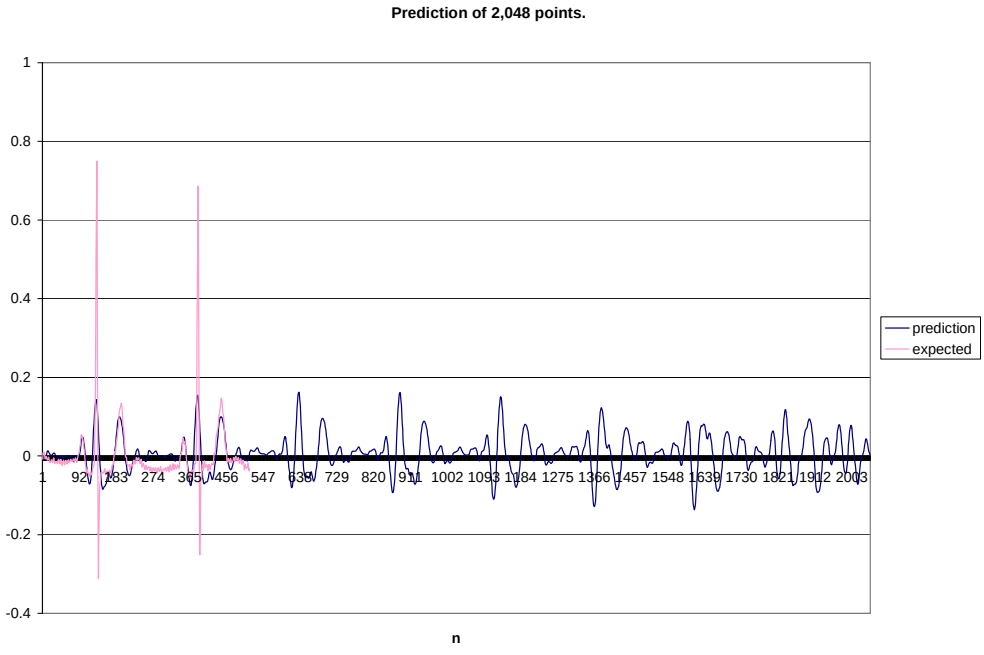


Figure 5.8 Results of Case C.3.A. (a) Prediction of 2,048 points obtained at Case C.3.A, compared with training signal

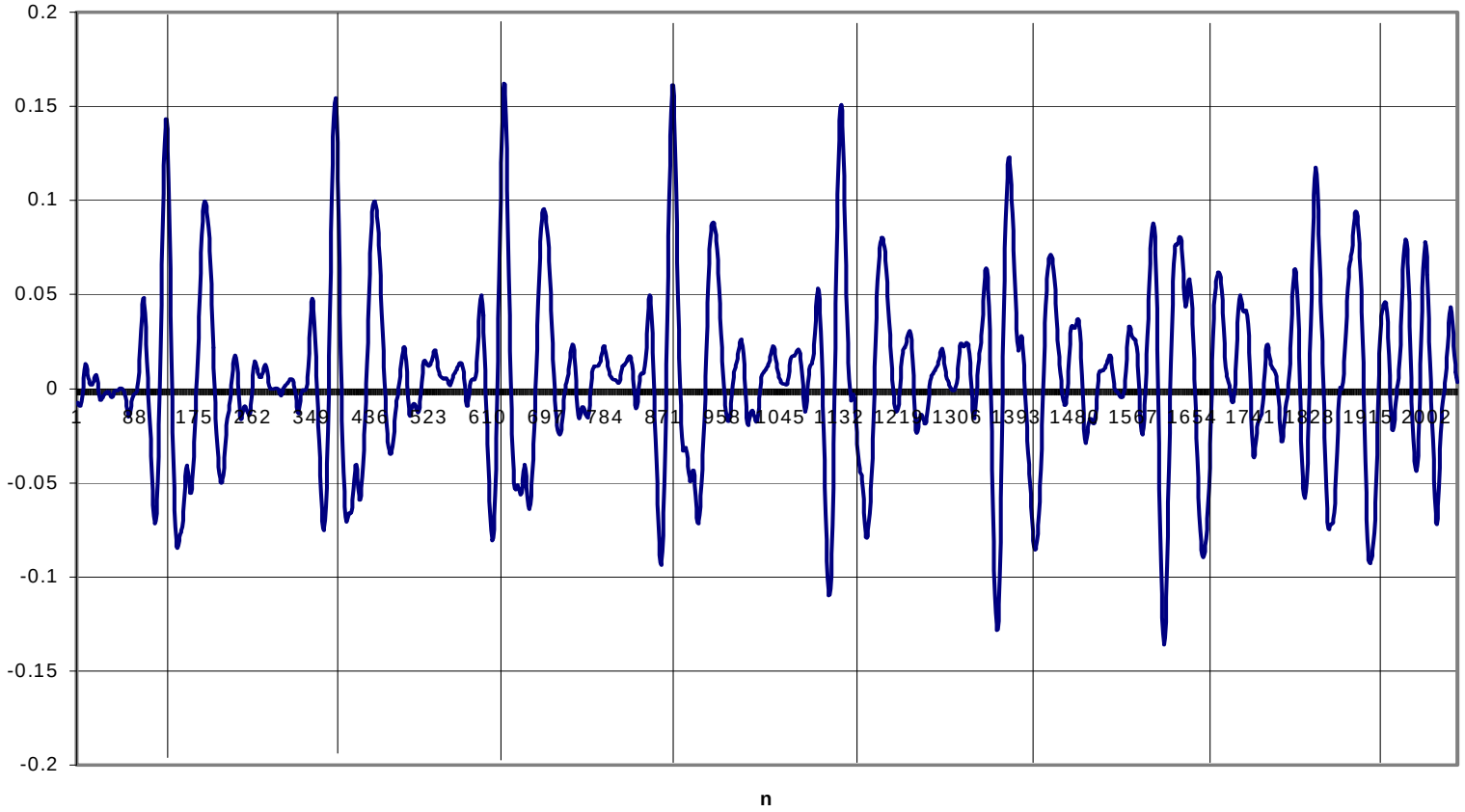


Figure 5.8. (b) Time periods in the prediction obtained at Case C.3.A.

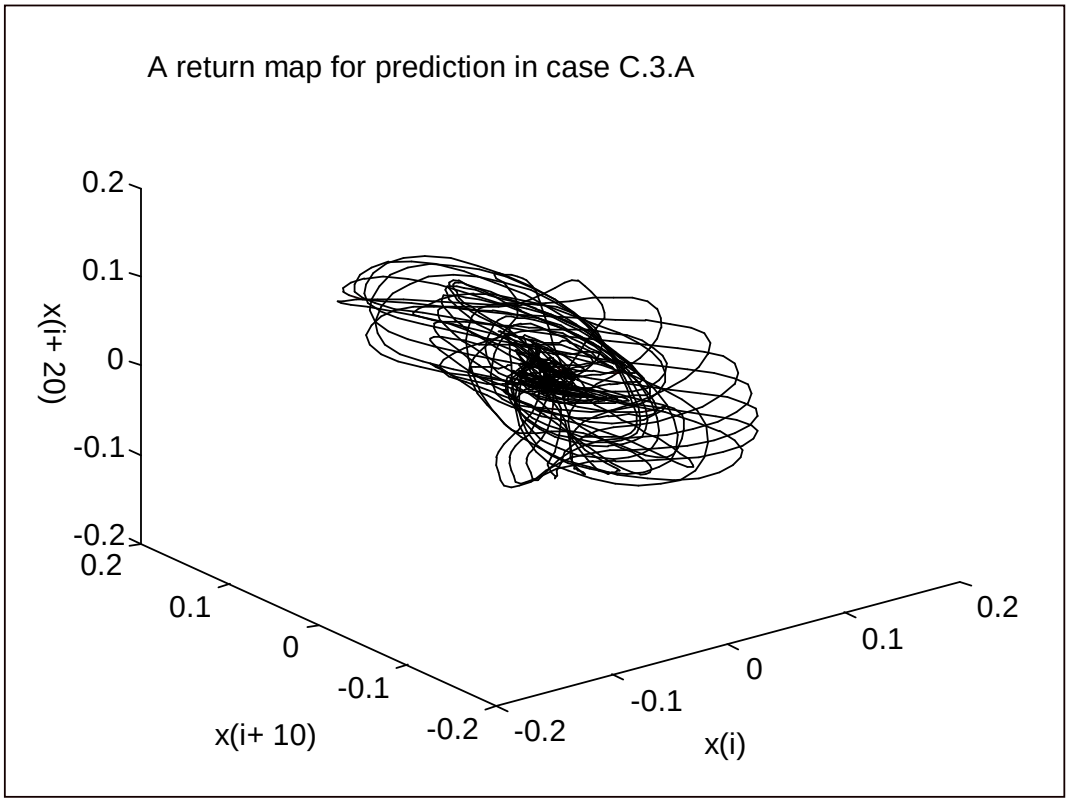


Figure 5.9. A return map generated for prediction at case C.3.A

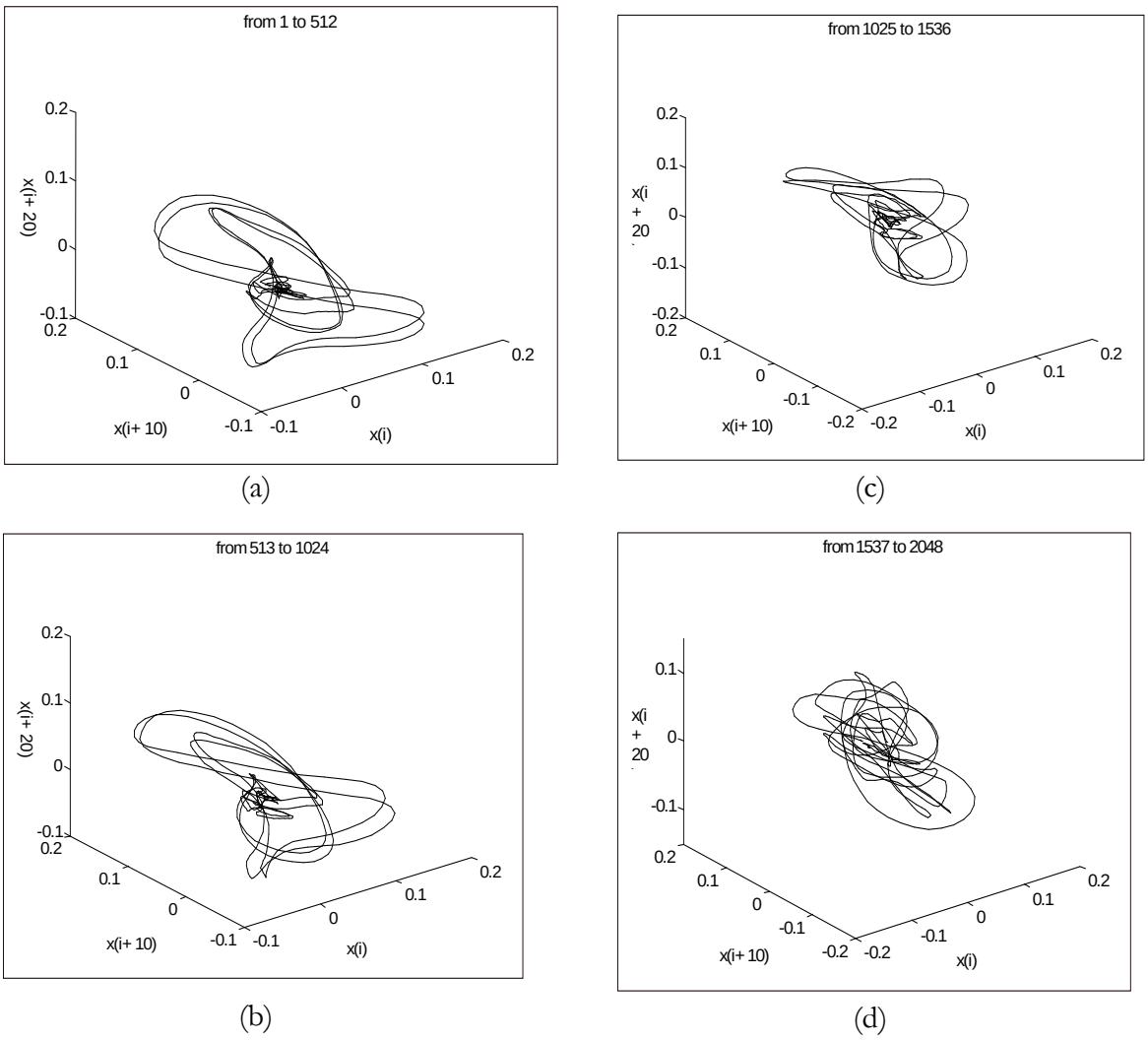
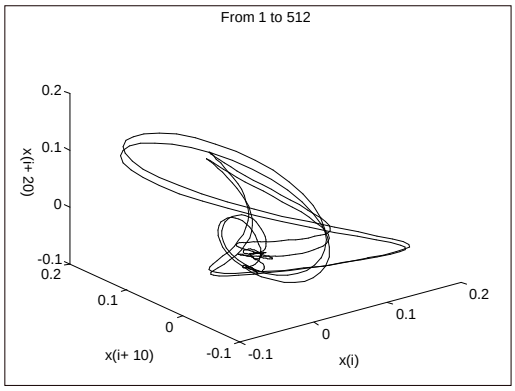
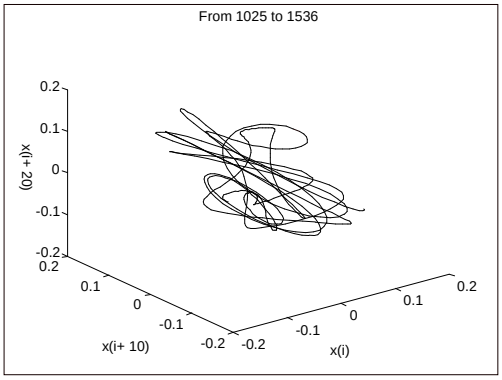


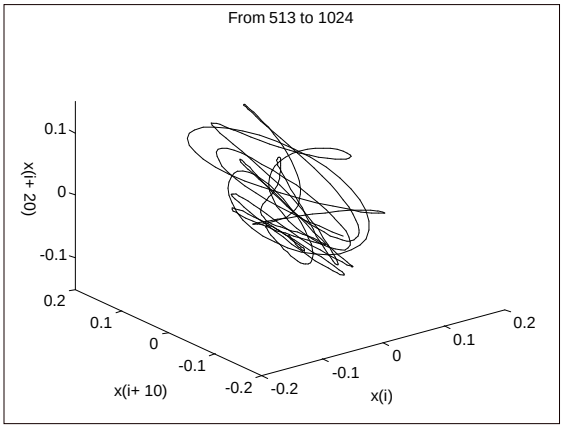
Figure 5.10. Return maps of segments of prediction generated by case C.3.A. (a) first segment, points 1 to 512; (b) second segment, points 513 to 1024; (c) third segment, points 1,025 to 1,536; (d) fourth segment, points 1,537 to 2,048



(a)



(c)



(b)

Figure 5.11. Return maps of segments of prediction generated by case C.0, (a) first segment; (b) second segment; (c) third segment.

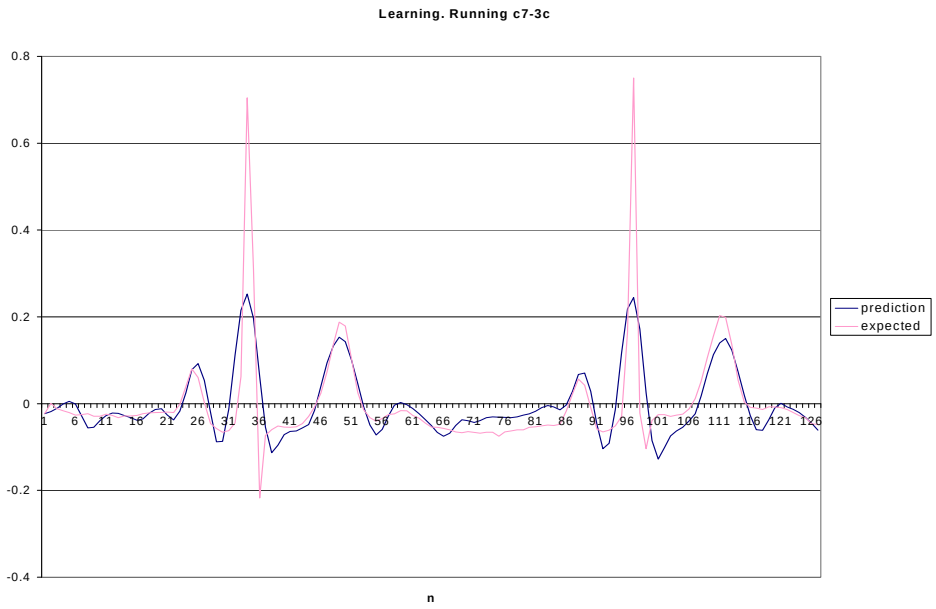


Figure 5.12. First 123 points learned for case C.3.B. MSE = 1.59E-2.

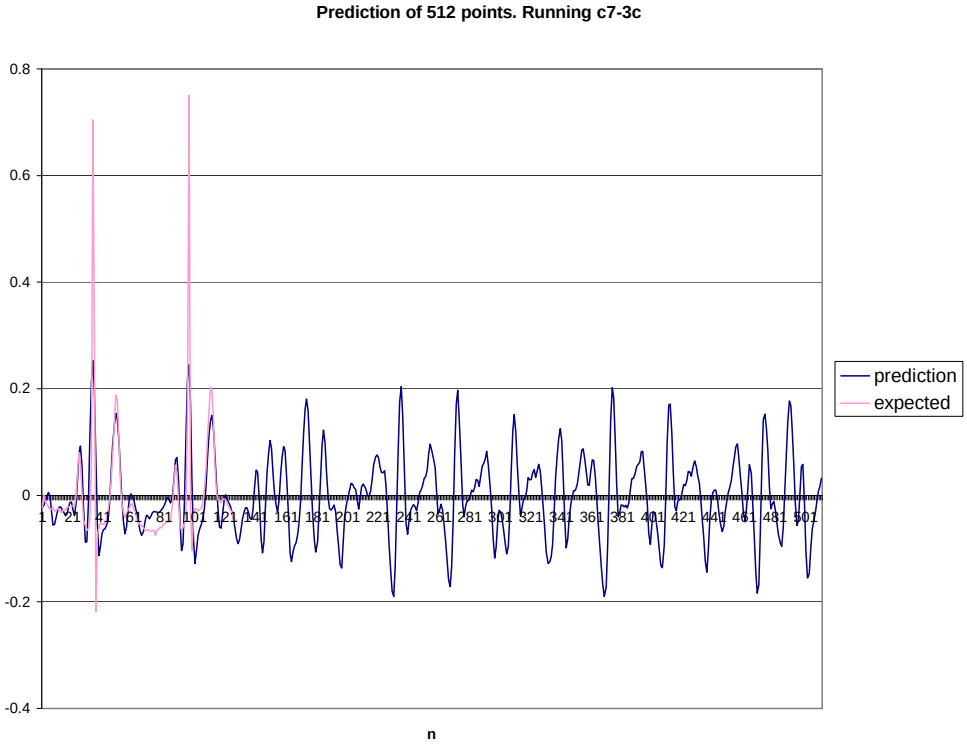


Figure 5.13. Long-term prediction of 512 points obtained at Case C.3.B.

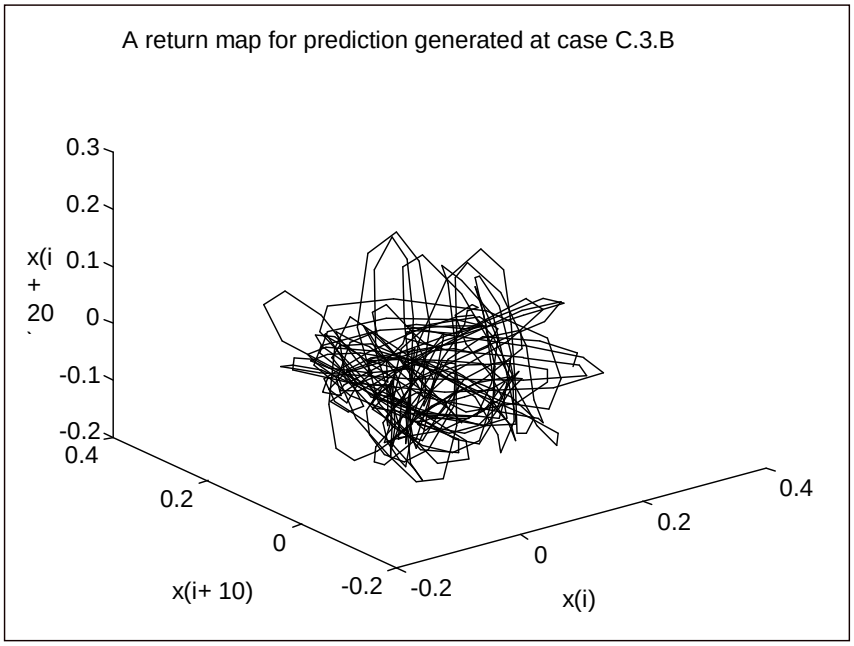


Figure 5.14 A return map generated for the prediction at case C.3.B.

5.2. A Predictor Based on a Hybrid Neural Network (Case H.1)

As specified at Table 5.2, the complex network has 1,089 weights to be adapted by the training algorithm. This number is large and hence difficult to adjust considering the amount of information available in the training signal.

Figure 5.15 shows a surface map of the weight matrix obtained after training at case C.3.A. Notice that the activity in the recurrent connections between nodes in the same layer and between harmonic generators and other nodes is minimal, compared to the rest of weights (see section 3.3.1.4 for a description of the weights organization). Based on this observation, and in an attempt to find a model with less adaptive parameters, a hybrid network was designed with harmonic generators in the first layer, feed-forward connections from first to hidden layer, and feed-forward connections from hidden to last layer. No recurrent connections were included, except the ones in the harmonic generators. The network is called hybrid because it contains both feed-forward and a few recurrent connections. As in the complex model, the last layer contains several pseudo-output nodes and one output node. There are no external inputs.

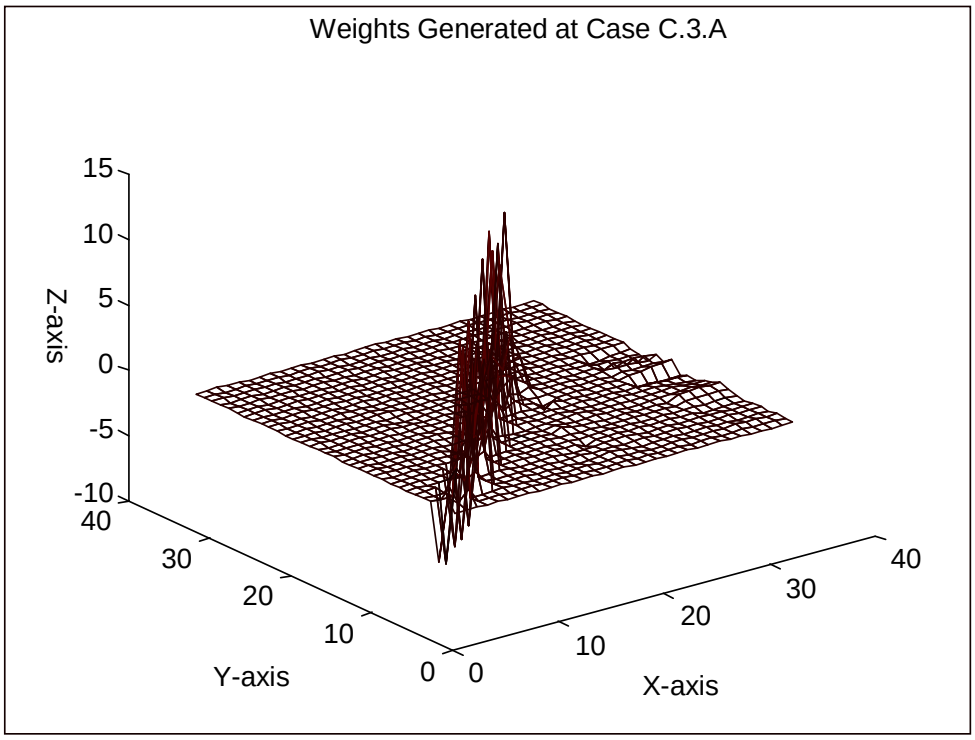


Figure 5.15. A surface map of the weights after training at case C.3.A.

Figure 5.16 shows this hybrid network; for simplicity, not all the connections are drawn there. Section C.1 contains the connection matrix fully describing this topology. As in the complex network, the harmonic generators are trained to produce the first 7 harmonics of the training signal.

A hybrid network with 7 harmonic generators, 7 hidden nodes, 4 pseudo-output nodes and 1 output node was trained to learn the signal ecg4.fil. After 45,000 epochs, this network reached a final MSE of 7.95E-3. Figures 5.17 and 5.18 show the results obtained for short and long-time prediction for this case. Figure 5.19 shows a return map of 2,048 points obtained by this architecture.

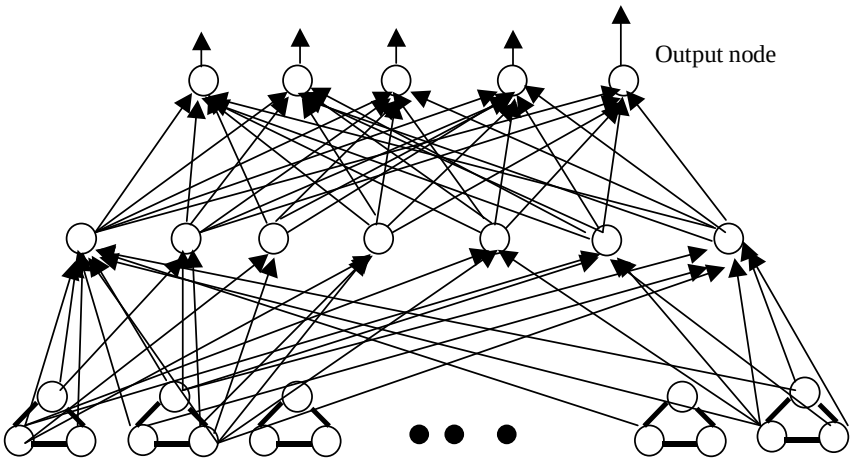


Figure 5.16. The hybrid network used at case H.1, with 7 hidden nodes, 4 pseudo-output nodes and 1 output node.

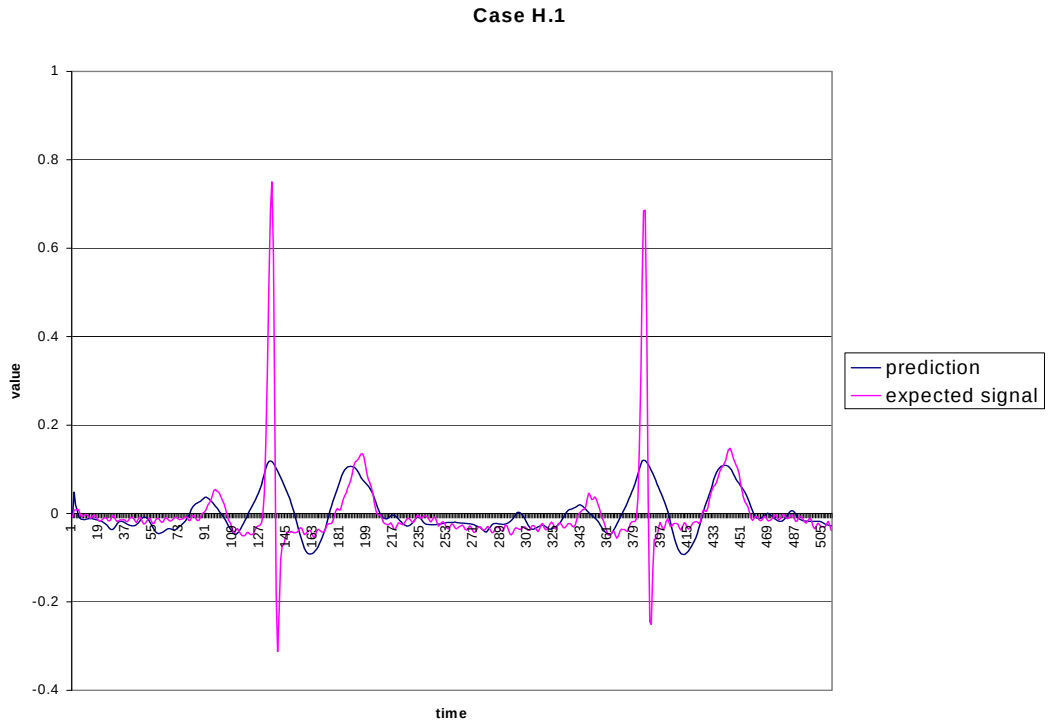


Figure 5.17. The first segment generated after learning for Case H.1. MSE = 7.9E-3

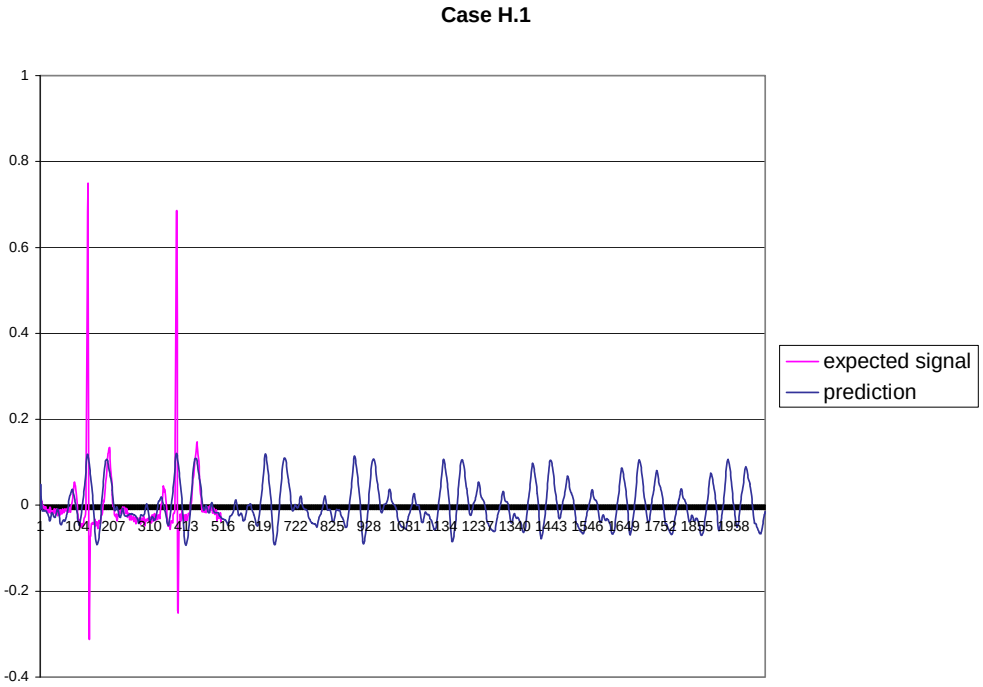


Figure 5.18. Prediction of 4 segments obtained at Case H.1

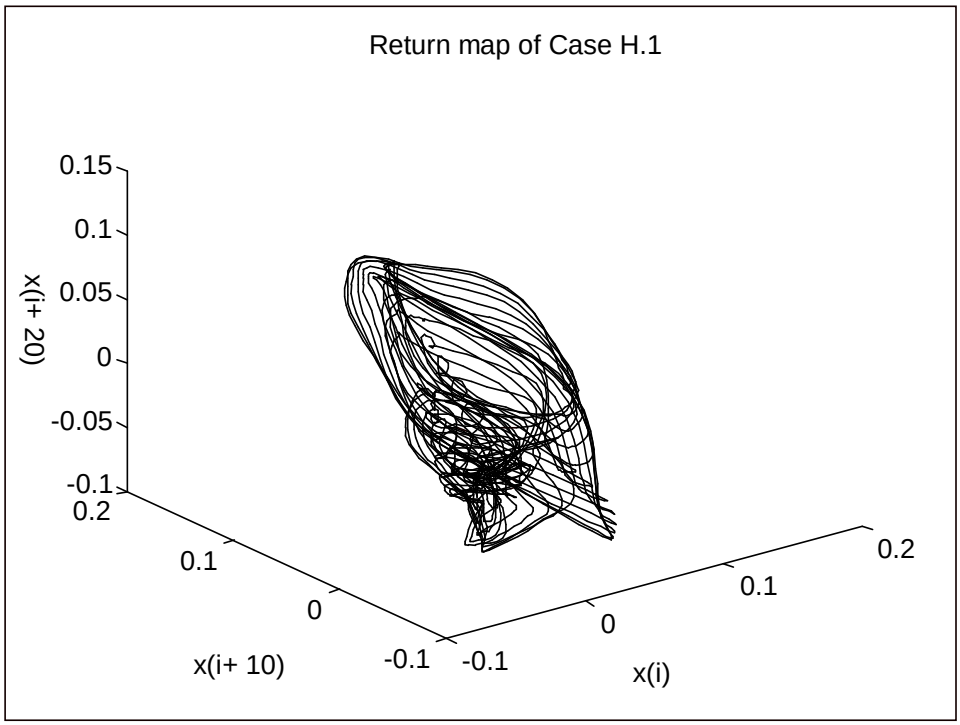


Figure 5.19. A Return Map of the prediction obtained at Case H.1

When comparing these results with the ones obtained by case C.3.A, the following is noticed: the MSE of case H.1 is slightly larger than the one in case C.3.A. The long-term prediction obtained by case H.1 (Figure 5.18) looks more "periodical," and with all their R and T-peaks with similar amplitudes, which resembles little the ECG shape. The attractor shown at the return map of case H.1 (Figure 5.19) does not resemble the expected shape for an ECG.

The maximum LE of the first predicted segment was 12.53±0.85; for four segments was 2.96±0.52; the one for the training signal is 2.23±0.27. Notice the strong difference between the values obtained for the training and first predicted signal. It is important to point out that the reason for this difference, besides the fact that the network is not representing correctly the dynamic of the system, may be also that the numerical algorithm calculating the LE is reaching its end before that a convergence state is reached. This is a drawback found in this method for calculation of LE.

As explained before, this model contains information about harmonics but no other kind of recurrence. The poor results obtained here, compared with the ones obtained by cases C.0 and C.3.A, show that recurrent connections play a very important role in modeling a dynamical system, even when their values are small in magnitude. This is because the weights associated to recurrent connections allow the network to learn how to "correct itself" from its own mistakes generated at past times.

5.3 Predictors Based in Feed-Forward Networks

Section 2.7.1.2 mentioned the ability of feed-forward neural networks to approximate a function from a set of observations. Such approximation, as shown by Gencay and Dechert (1992), is topologically similar to the dynamical system generating the observations; it can be used to calculate the Lyapunov exponents of such a system. The system imbedded in this kind of neural network has the same invariants as the original system, provided that such representation is accurate enough.

Using this idea, a multi-layer, feed-forward network with external inputs (Figure 5.20) was used to construct a predictor; Topology B at section C.2 fully describes their connections. This network was trained to approximate a signal; then its ability for long-term prediction was evaluated. The predictor works as follows: during the learning state, it

receives as external inputs points of the training signal at times *t-1, t-2 … t-k,* and it is trained to approximate the value of point *s(t)* (Figure 5.21 (a)). During the prediction state the network may receive external inputs coming from observations of the system or it may receive feedback of its own past predictions when no more observations are available (Figure 5.21 (b)). Notice that this is not a recurrent network, because neither any weights are attached to the feedback, nor any other information about the past output values of the nodes is involved during training.

This architecture was tested first with a very simple signal, a sine function, and then with an ECG. The network used in both experiments had 5 input nodes, 10 hidden nodes and 1 output node.

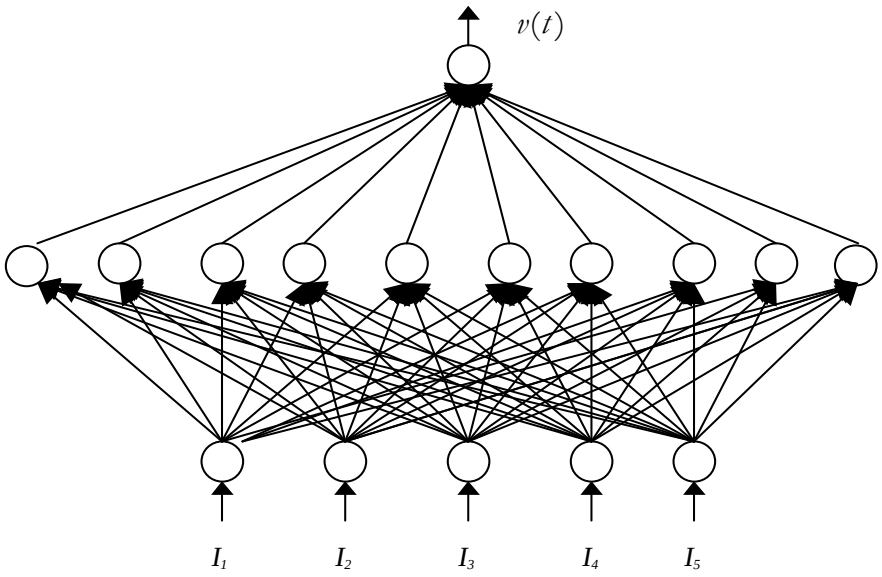


Figure 5.20. The feed-forward network with external inputs used at cases F.1, A and B.

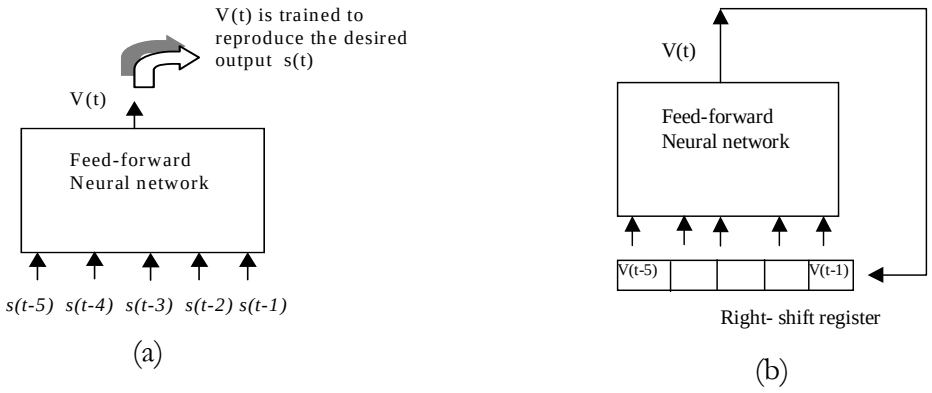


Figure 5.21. A feed-forward predictor: (a) Training process: *s(t)* is the training signal; *v(t)* is the actual output of the network. (b) point-by-point prediction

5.3.1. Case F.1.A: Prediction of a sine function using a feed-forward network

First, this architecture was tested to predict a sine function. After 10,000 epochs, the MSE was 2.0E-3. Figure 5.22 shows the first 99 predicted points when using the training signal to feed the network; Figure 5.23 shows a long-time prediction of 1,500 points, where feedback of its own outputs was provided to the network. This long-time prediction generated very good results, except for a slight shift in the phase of the predicted signal, which is almost unnoticeable.

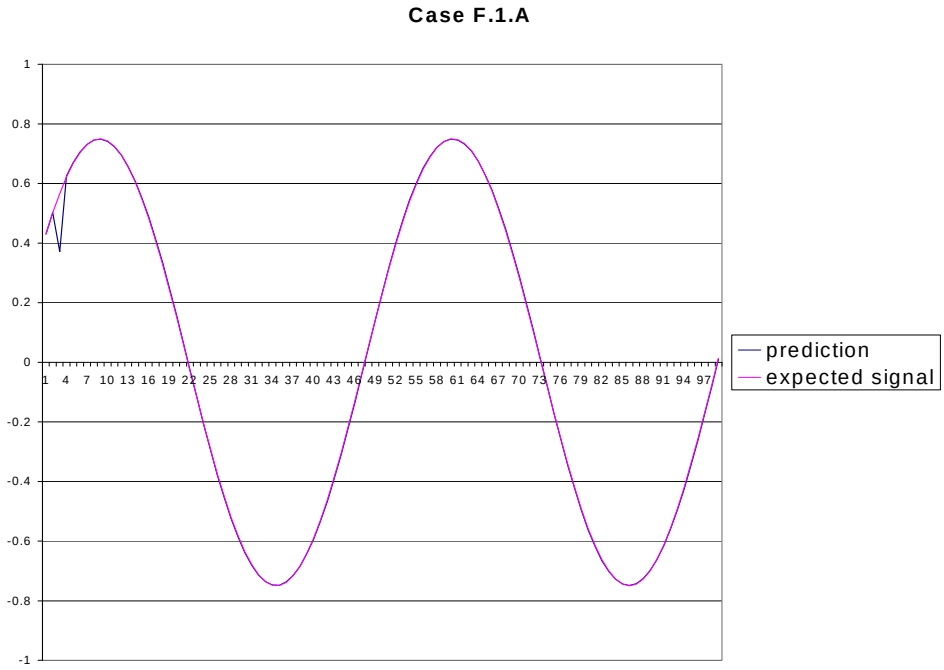


Figure 5.22. First 99 predicted points using original observations as input. Case F.1.A. MSE = 2.0E-3.

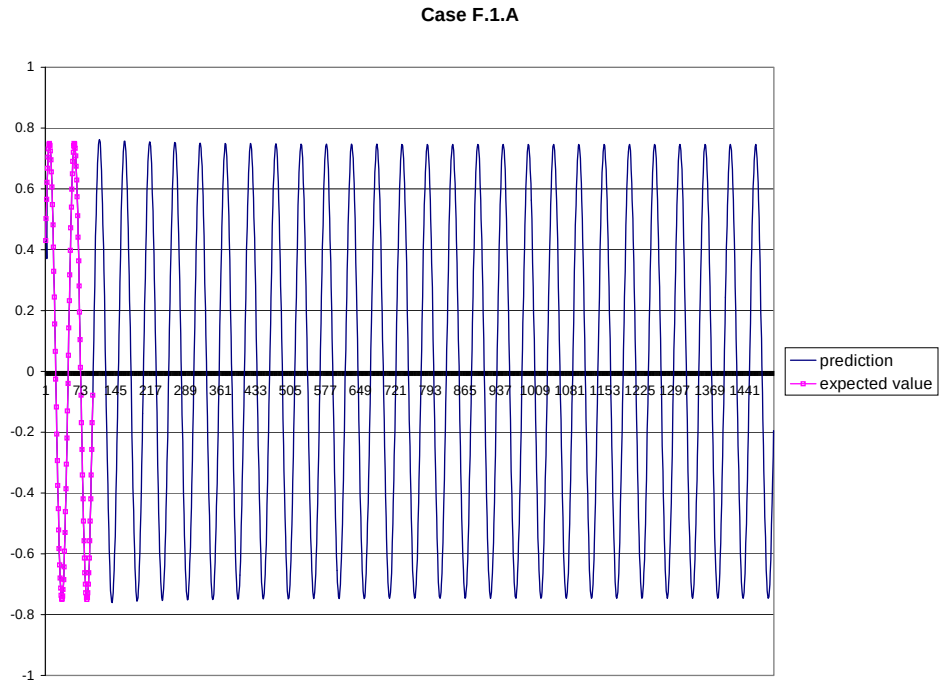


Figure 5.23. Long-time prediction of a sine function using feedback, compared with expected values. Case F.1.A.

5.3.2. Case F.1.B: Prediction of an ECG Using a Feed-forward Network

The same feed-forward net used in case F.1.A was trained in this case to reproduce and predict an ECG. The resulting MSE after 50,000 epochs was 1.4E-3. Figure 5.24 shows the first 470 points predicted when using original observations as external inputs. This next-point prediction works very well. Figure 5.25 shows the absolute error by point obtained during the prediction of this segment. The error in most of the points is small, except during the prediction of the R-peaks.

Figure 5.26 shows a long-term prediction from points 460 to 660. The first 10 points in this plot were calculated using inputs coming from observations; from point 471 ahead the network received feedback of its own calculated values. The prediction resulted in a periodical signal oscillating approximately to a frequency of 142 Hz., without any resemblance to the ECG. Notice that the amplitude at some points reached values greater than 10, while the expected highest values for this ECG signal is 0.75.

Figure 5.27 shows the absolute error-by-point obtained when predicting points 471 to 479, which required to feed the network with one or more of its own outputs; after the 6^{th} point, the error started growing fast

Therefore, this network was unable to realize any long-term prediction successfully, regardless of its excellent capability for next-point prediction.

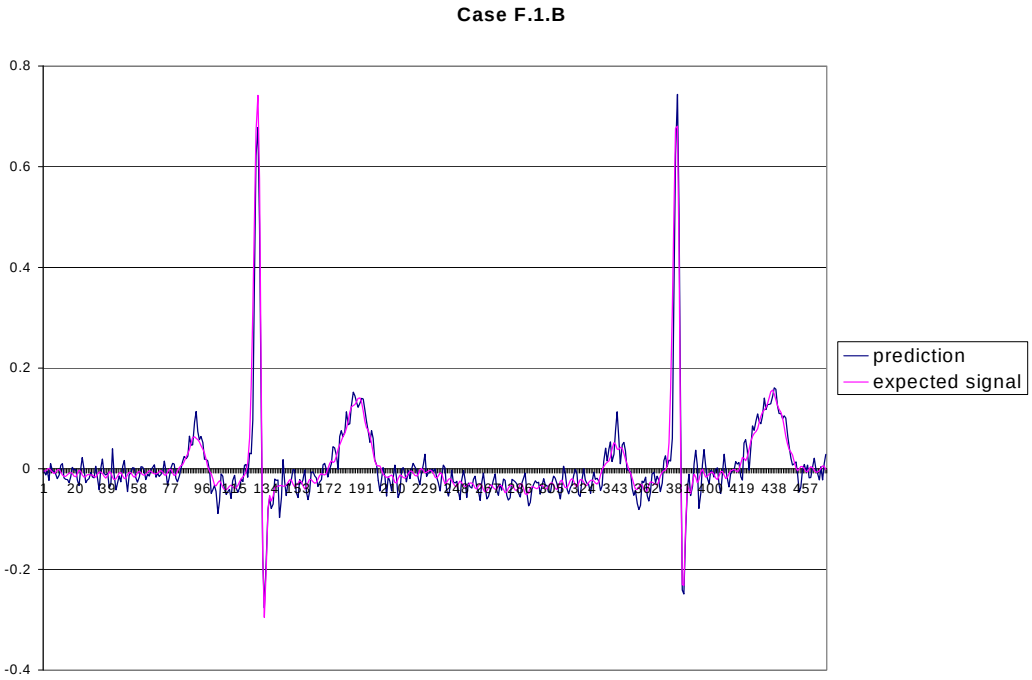


Figure 5.24. First 470 predicted points of an ECG. Case F.1.B
MSE = 1.4E-3

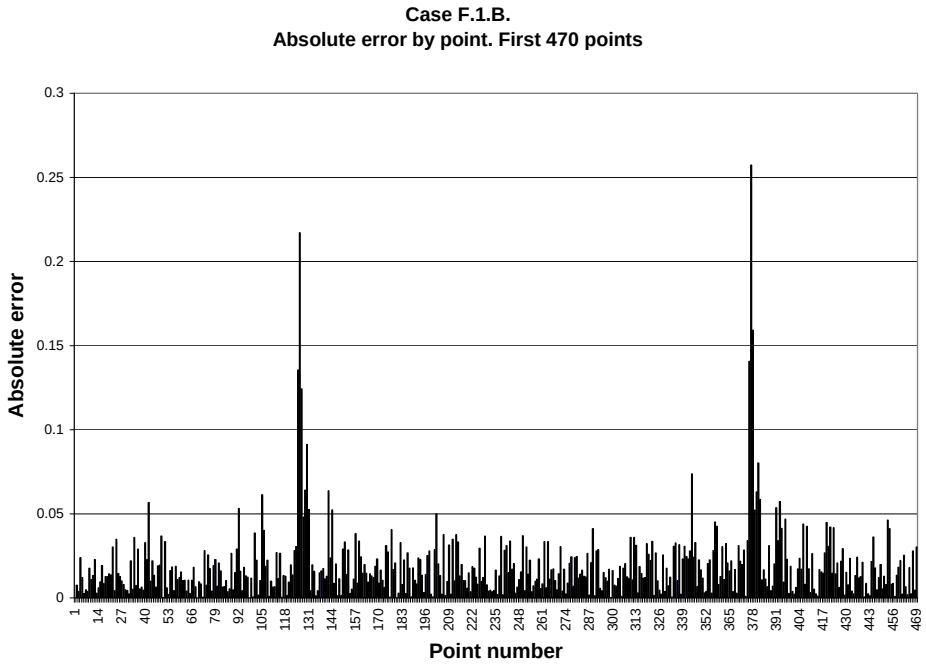


Figure 5.25. Absolute error by point for the first 470 predicted points.
Case F.1.B

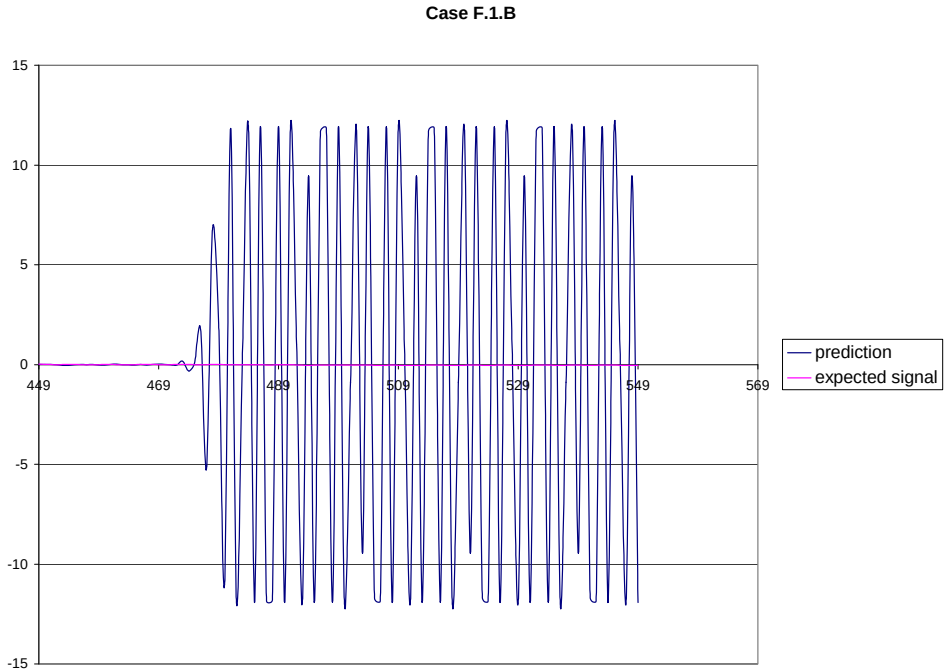


Figure 5.26. Predicted points 460 to 660. Case F.1.B.

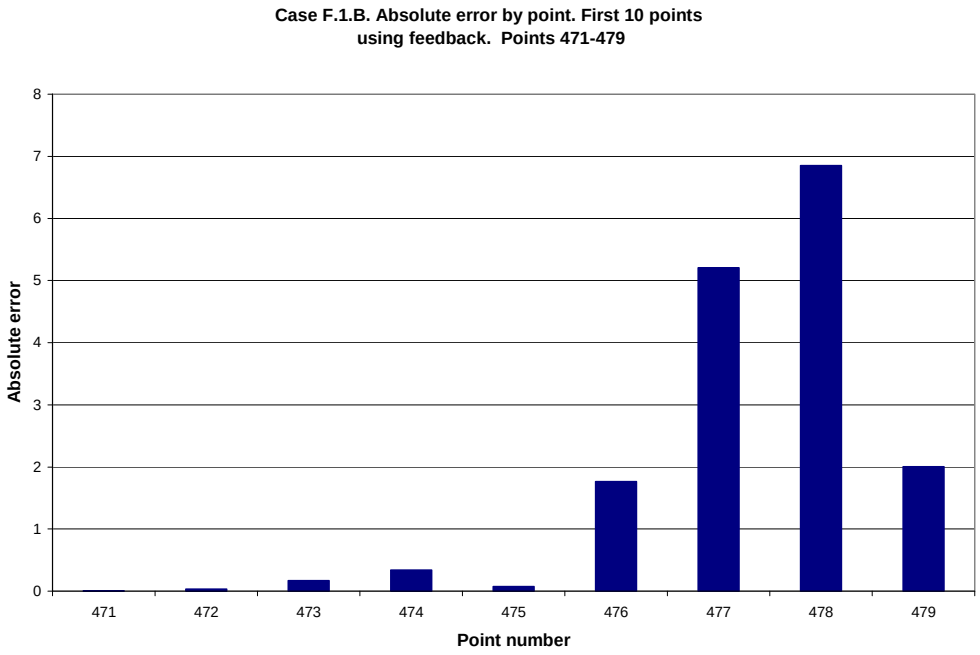


Figure 5.27. Absolute error by point for points 471-479. Case F.1.B

5.3.3. Case F.2. Addition of Recurrent Connections to a Feed-forward Network

It is obvious from the results of case F.1.B that a feed-forward net, by itself, is not able to capture the dynamics of the system generating the training signal, when several frequency components are involved. In an attempt to fix this problem, recurrent connections were added to that network, in order to give it the ability to represent the temporal information imbedded in the signal.

First, a feed-forward net was fully trained and the value of the minimum reached error was recorded. Next, the network was trained again using the same initial weights, but this time the training was stopped when the error reached approximately ½(maximum error + minimum error). This was done in order to induce in the weights of the network some information about the signal, and consequently its Lyapunov exponents, but leaving some learning possibilities to the recurrent network. After that, the feed-forward network was converted to fully-recurrent and trained until it reached a local minimum.

The feed-forward network used here is the same as at case F.1: 5 input nodes, 10 hidden nodes and 1 output node (Figure 5.20). Training was stopped after 92 epochs when MSE was 0.0029. Next, recurrent connections were added to make it a fully-connected recurrent network with 5 input nodes. The training continued 100,000 more epochs; at this point the MSE was 8.34E-4. Figure 5.28 shows the first 470 points predicted when original observations were given as inputs; Figure 5.29 shows the absolute error-by-point generated during the prediction of this segment. This next-point prediction performed excellent. Figure 5.30 shows the error-by-point generated when predicting points 471 to 479, which were calculated using one or more outputs of the network as external inputs. After the 8th predicted point, the error starts growing; from that point ahead, the outputs of the network grew exponentially with time.

Comparing these results with the ones obtained by case F.1, it is clear that the recurrent connections gave to the network some ability to predict better in a short time fashion, but the network failed to realize any meaningful long-term prediction.

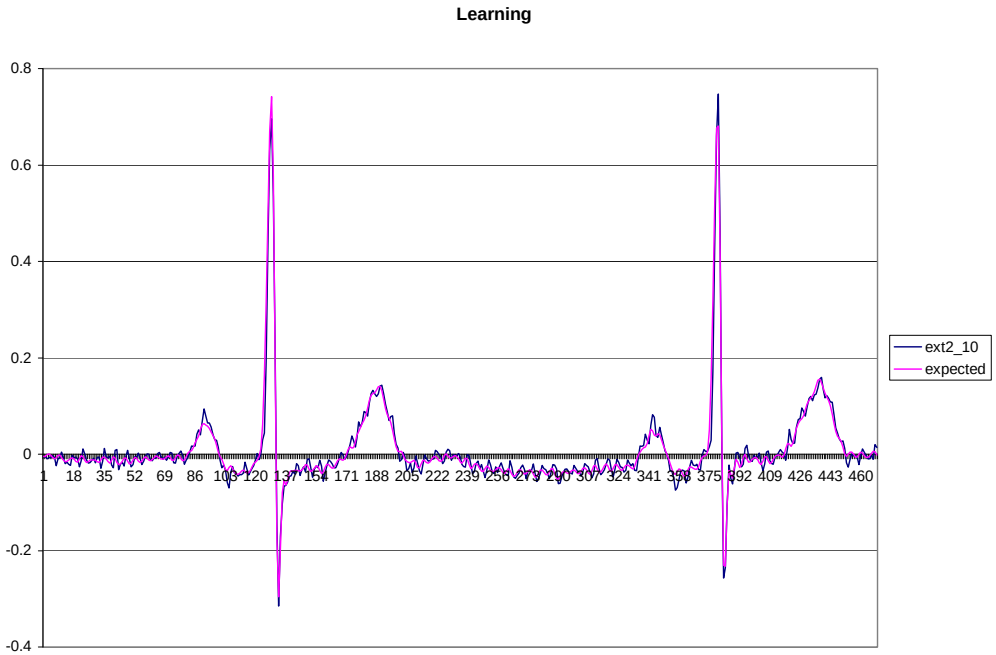


Figure 5.28. First 470 points predicted in case F.2.
MSE = 8.4E-4

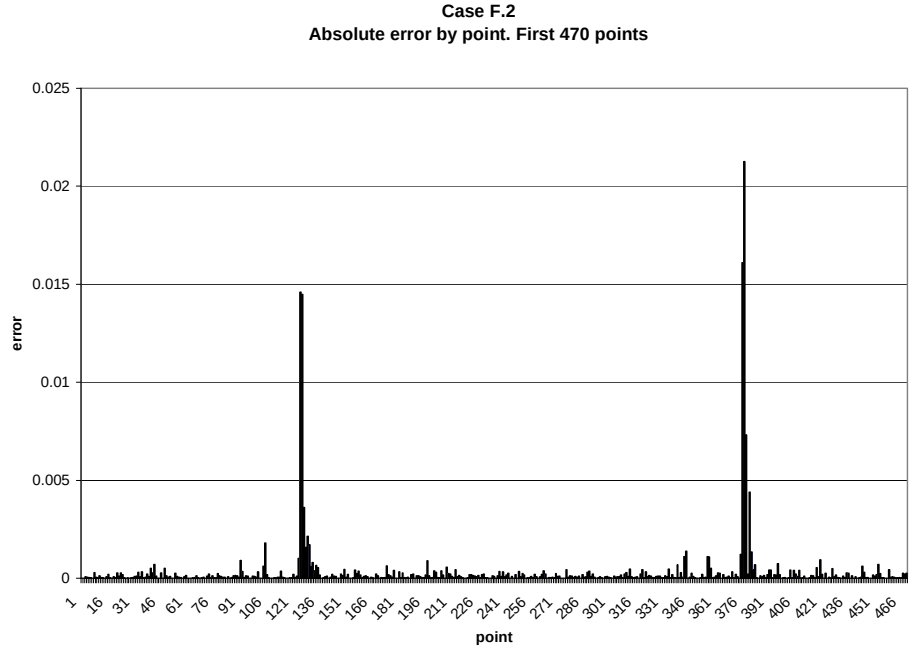


Figure 5.29. Absolute error by point for first segment predicted in case F.2

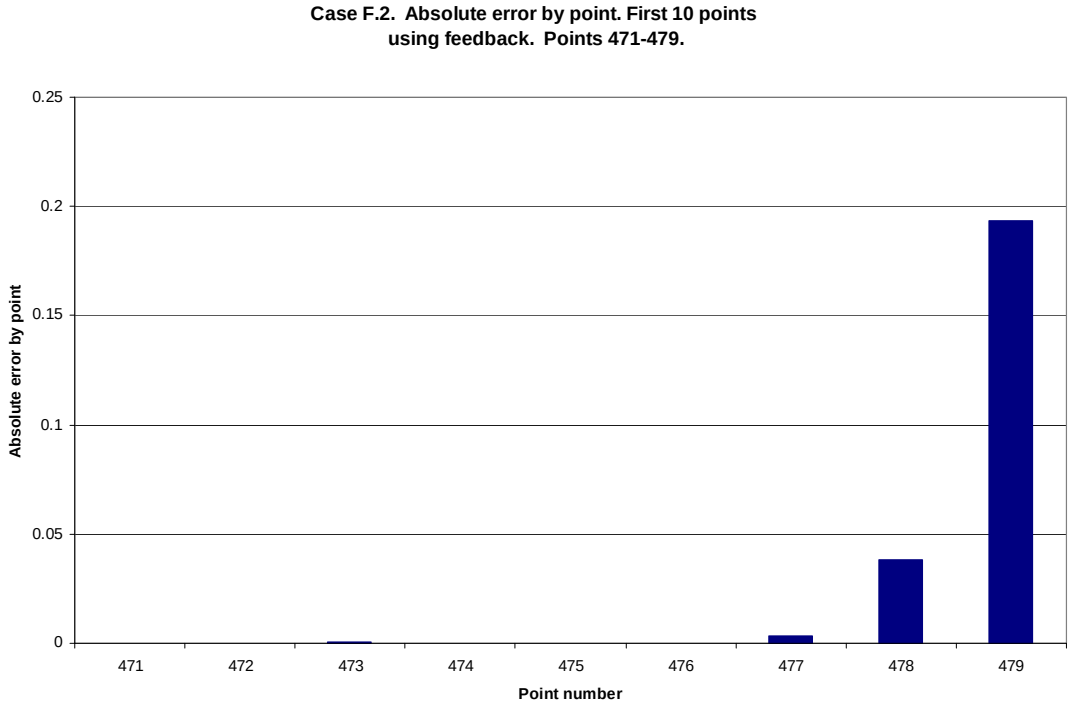


Figure 5.30. Absolute error by point for points 471-479. Case F.2

5.4 Case K: Predictors Based in Hybrid Networks Using Harmonic Generators and External Inputs

From past experiments it was observed that feed-forward networks are able to approximate functions very well, provided that exact information about the past of the signal is given. However, they are not well suitable for long-term prediction, probably because of the error accumulated due to the feedback of non-exact, predicted values used as external inputs. Feed-forward networks contain no memory, they can go only to fixed points. It was also shown that harmonic generators made with a 3-node fully-recurrent neural network are able to keep oscillations for long periods of time and, therefore, they allow accurate long-term prediction without need of external inputs. The complex network model showed that such generators can be combined to produce signals with many frequency components. However, the signals generated by the whole recurrent network do not reproduce the peaks of the signals with enough accuracy to be useful for long-term prediction.

Based on this observations, two kinds of predictors were built combining the ideas of harmonic generators and external inputs. One of them did not include any hidden nodes; the other include a hidden layer to allow for a better internal representation, but with a corresponding cost in learning. In both experiments, the network had 5 external inputs and 7 harmonic generators.

These predictors were trained in a way similar to the one used in case F.2: first an "almost feed-forward" network was partially trained, in order to induce in the system information about the Lyapunov exponents of the signals; then the network was converted to a fully recurrent network and trained until it reached a local minimum. Also, similar to case F.2, feedback of the values calculated by the network is required during the long-term prediction process, because a time is reached when original observations are not available anymore to feed the network (see figure 5.21). The obtained results are described next.

5.4.1. Case K.1: A Predictor with no Hidden Nodes

Figure 5.31 shows the network used in this case. For simplicity not all connections are drawn in the figure but section C.3 fully describes its topology.

In the first phase of prediction, the network trained during 10,000 epochs. At this point the MSE was 3.67E-3. Next all recurrent connections were added and the network trained 20,000 more epochs, reaching an MSE of 3.10E-3. Figure 5.32 shows the first 507 points (one segment) predicted using original observations as inputs. Figure 5.33 (a) shows a long term prediction of 4 segments, compared with the training signal.

In figure 5.33 (b) the same prediction is plot in a grid with vertical lines separated the same distance as the two first R-peaks of the predicted signal. The period of the signal keeps almost the same for the 4 cycles; but the amplitudes of the peaks R and T are not as expected.

The maximum LE of the first predicted segment was 4.47 ± 0.33; the one corresponding to four segments was 5.09 ± 1.14; the maximum LE of the training signal is 3.23 ± 0.27. These results are the best of all experiments executed in this research with respect to accuracy in the Lyapunov exponents.

Figure 5.34 shows the return map of 4 predicted segments. Even though its shape is not as the expected, some regularity in the geometrical shape is noticed in the attractor.

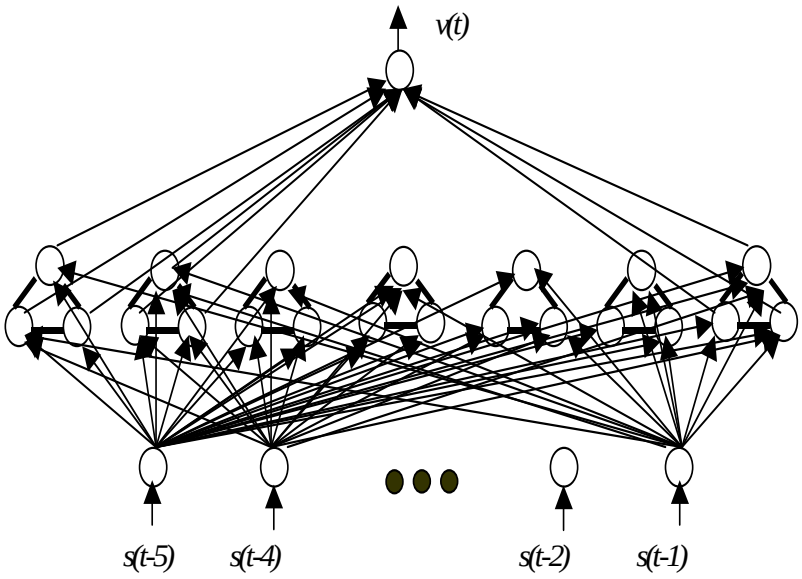


Figure 5.31. A neural network with harmonic generators, external inputs and no hidden layers used at case K.1

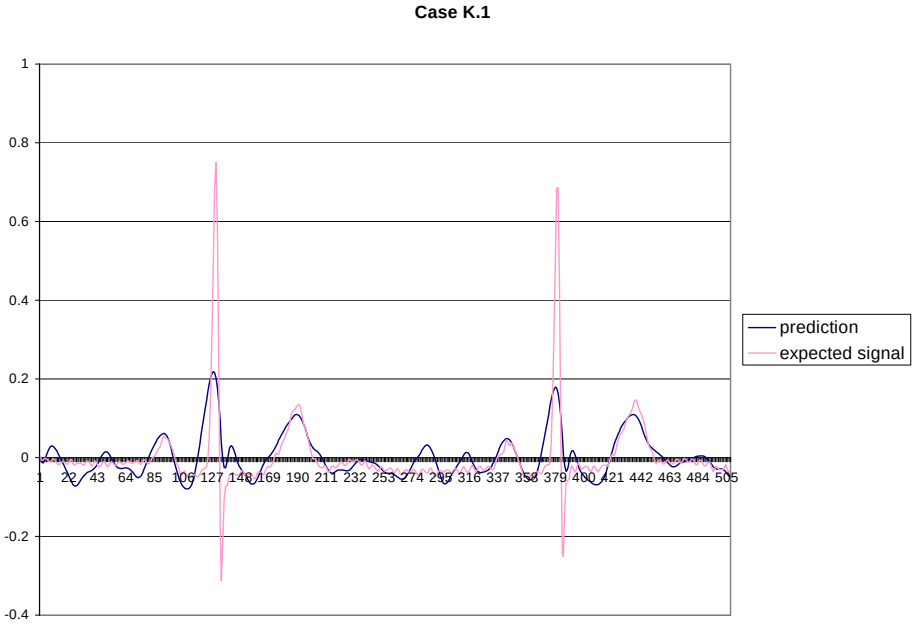


Figure 5.32. First 507 predicted points obtained at case K.1
MSE = 3.10E-3

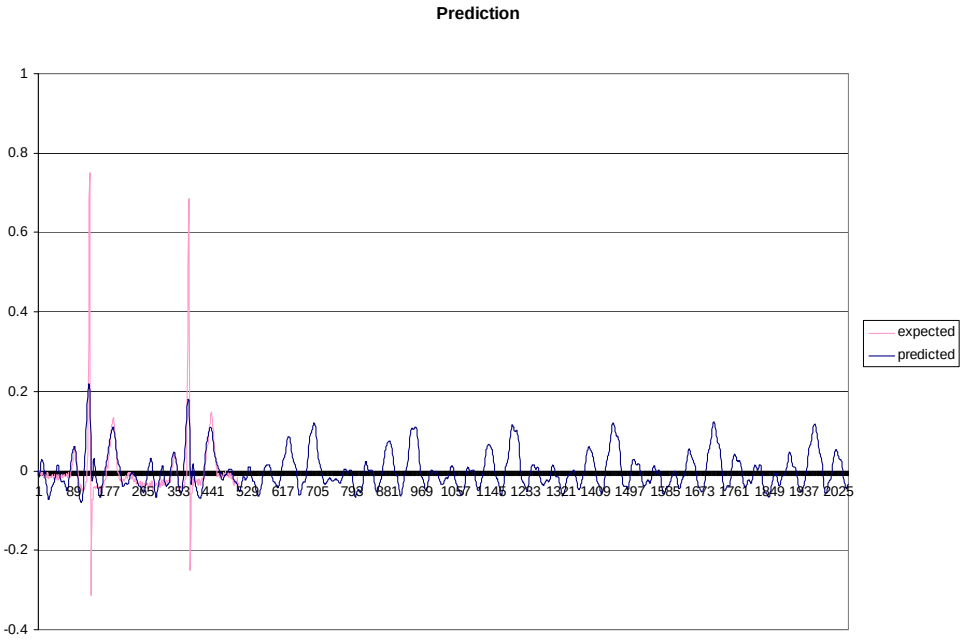


Figure 5.33. Results of case K.1. (a) Four segments predicted by case K.1.

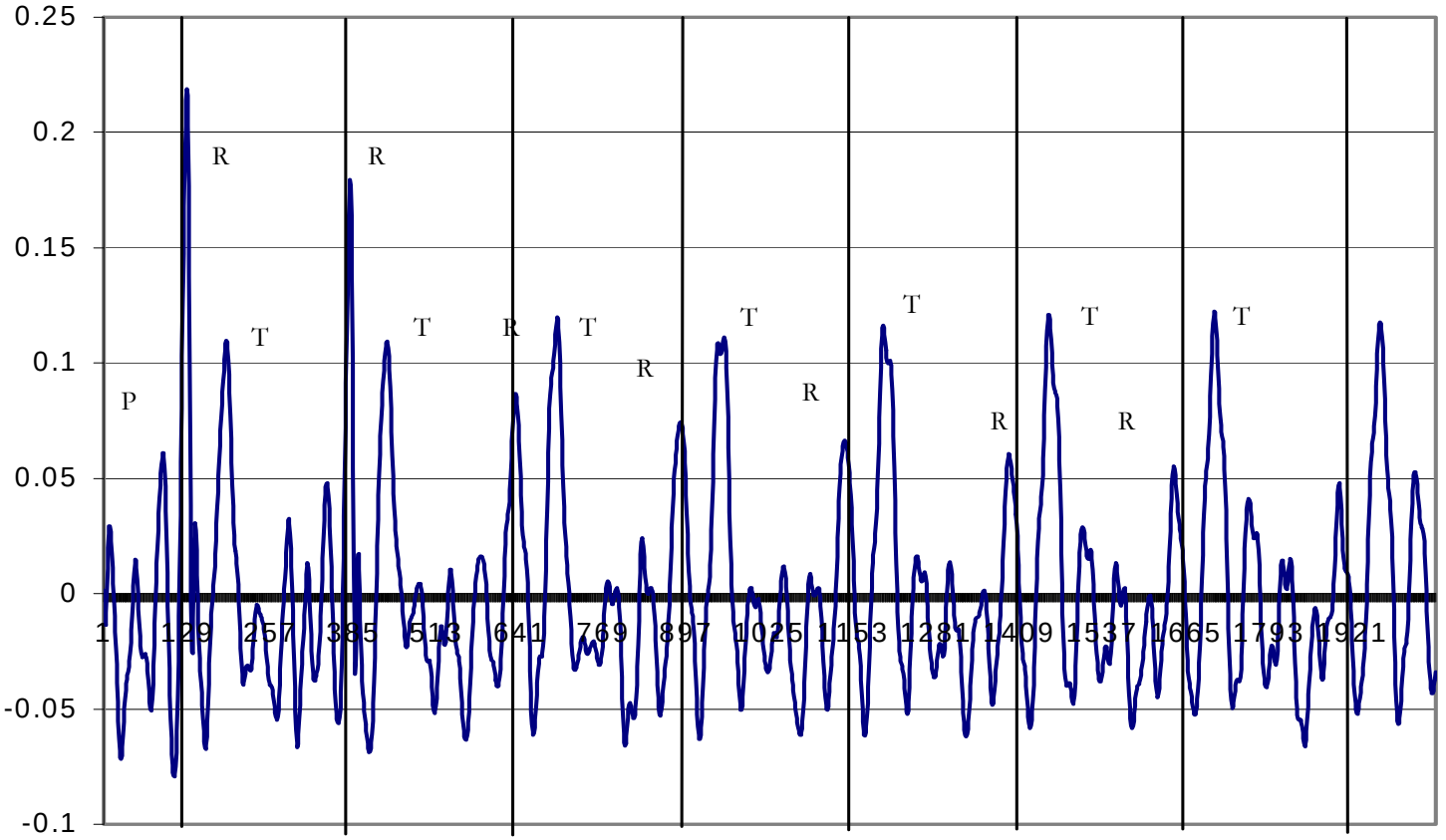


Figure 5.33 (continuation). (b) Time periods in the prediction obtained at case K.1

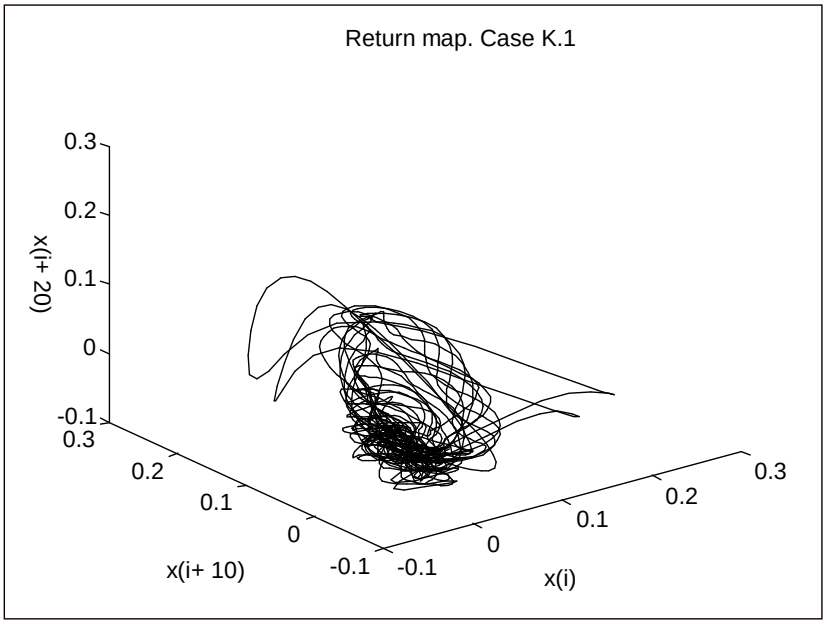


Figure 5.34. Return map of 4 predicted segments. Case K.1

5.4.2. Case K.2.A A predictor with hidden layers for ECG

In this case a hidden layer of 5 nodes was included in the network described before, and trained to model an ECG signal. Figure 5.35 shows this network; its corresponding connection matrix is given at section C.4.

This network was trained for 10,000 epochs, obtaining a MSE of 3.14E-3. Next, it was converted to a fully-recurrent network and trained 20,000 more epochs. The MSE reached a value of 2.35E-3. Figure 5.36 shows the first 507 points (one segment) predicted when using original observations as external inputs; Figure 5.37 (a) shows a long-term prediction of 4 segments. Figure 5.37 (b) shows the prediction with vertical lines separated to the same distance as the first R-peaks. Notice the shift of the signal after the second segment.

Figures 5.38(a) to (d) show the Fourier transform of each consecutive segment predicted by the network. The first one (a), shows some resemblance to the FFT of the training signal (see figure D.3). However, the rest of the FFT do not contain several of the frequency components presented at figure D.3.

Figure 5.39 shows the return map of the prediction of 4 segments; Figures 5.40 (a) to (d) show the return maps of each one of these 4 segments. Notice that only the return map corresponding to the first segment resembles in some way to the geometry expected in a return map of an ECG.

Comparing these results with case C.3.A, it is noticed that, during the prediction of the first 507 points in case K.2.A, the amplitude of their peaks is larger than in case C.3.A, which is an advantage of K.2.A over C.3.A. However, after the first predicted segment, in K.2.A the shape of an ECG is lost, probably due to the inaccuracy of the output values calculated by the network and given as feedback to be used as external inputs when original observations are not available anymore.

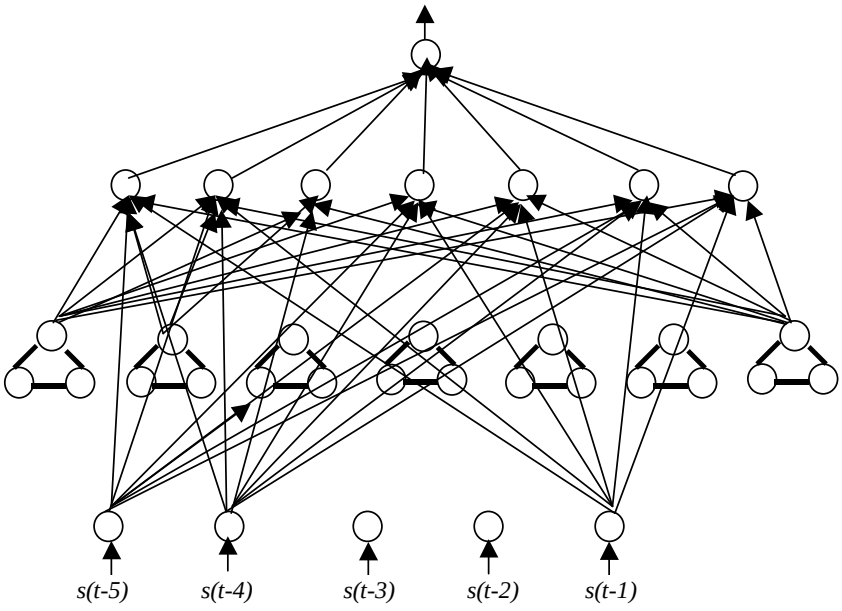


Figure 5.35. A network with harmonic generators, external inputs and a hidden layer. Topology D (section C.4)

Figure 5.36. First 507 points predicted at case K.2.A.
MSE = 2.35E-3.

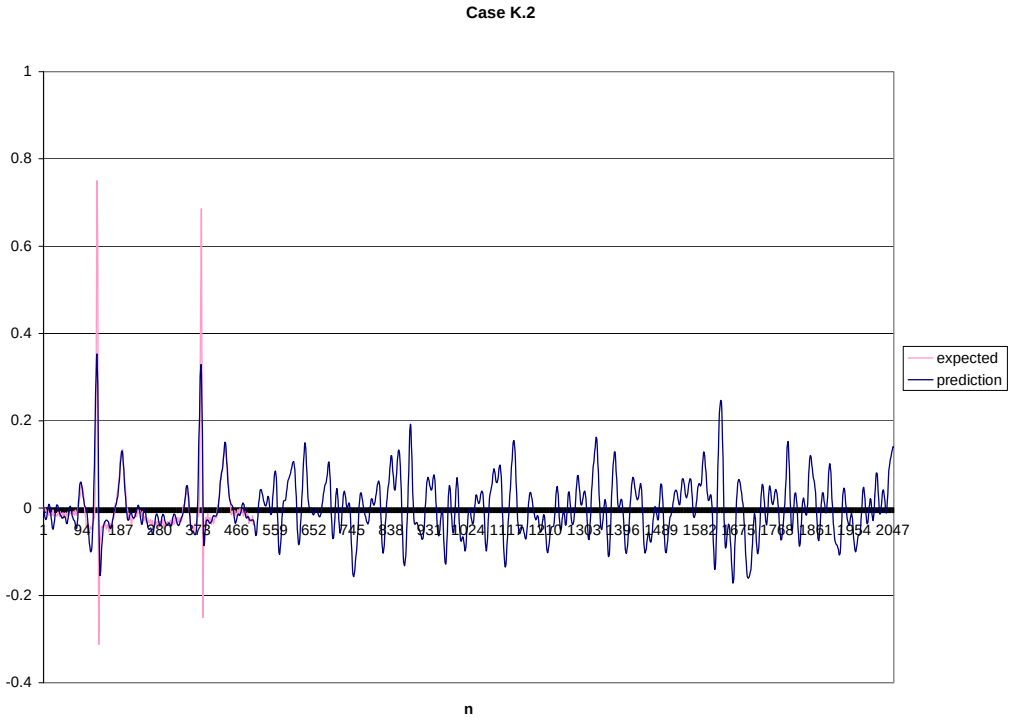


Figure 5.37. Results of Case K.2.A. (a) A prediction of 2,048 points. Case K.2.A

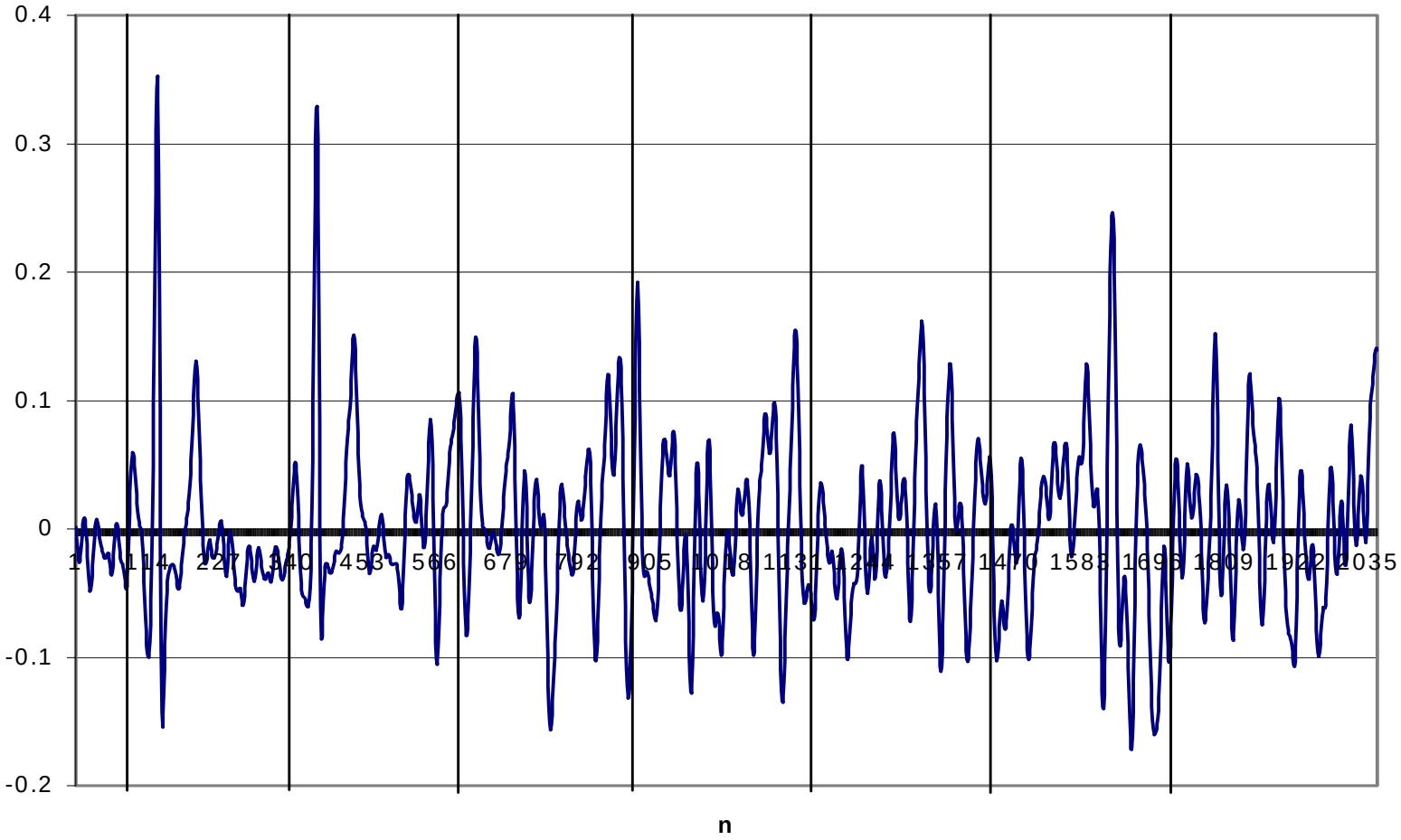
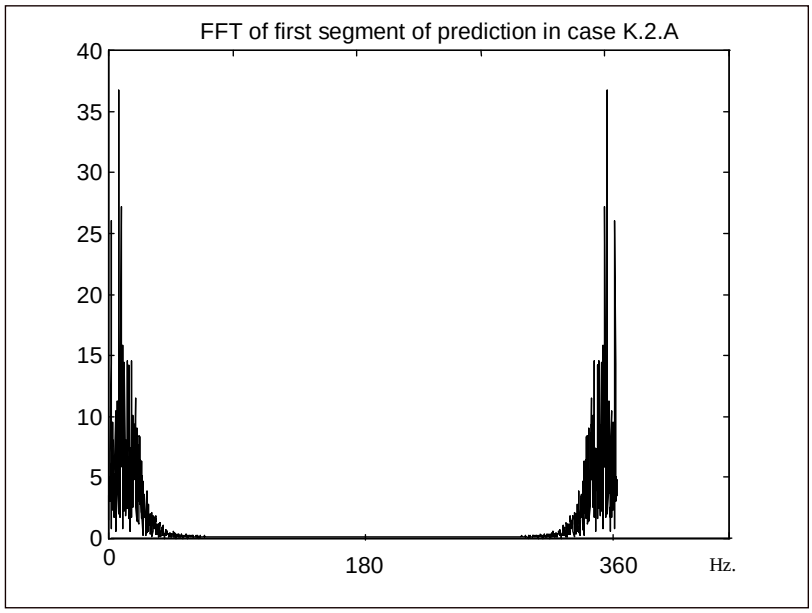
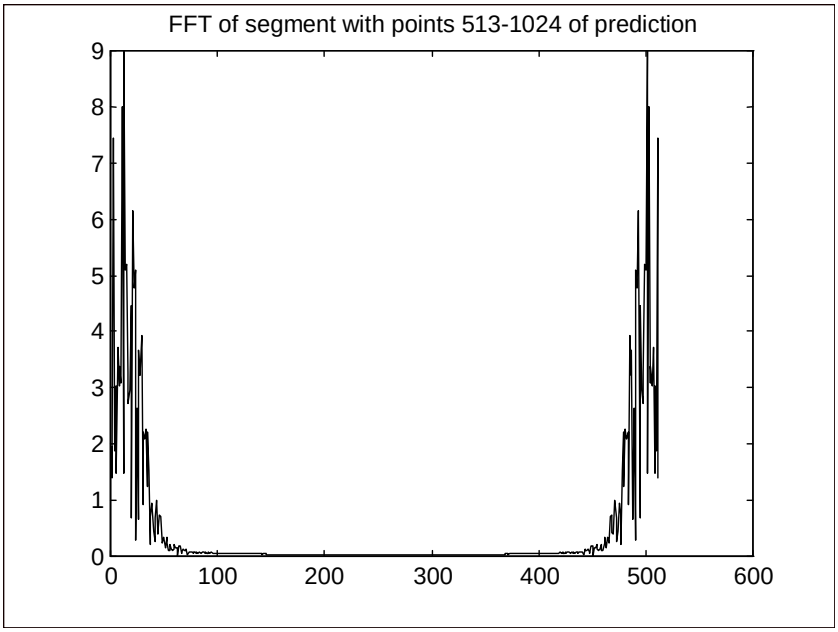


Figure 5.37 (continuation). (b) Time periods in the prediction obtained at case K.2.A

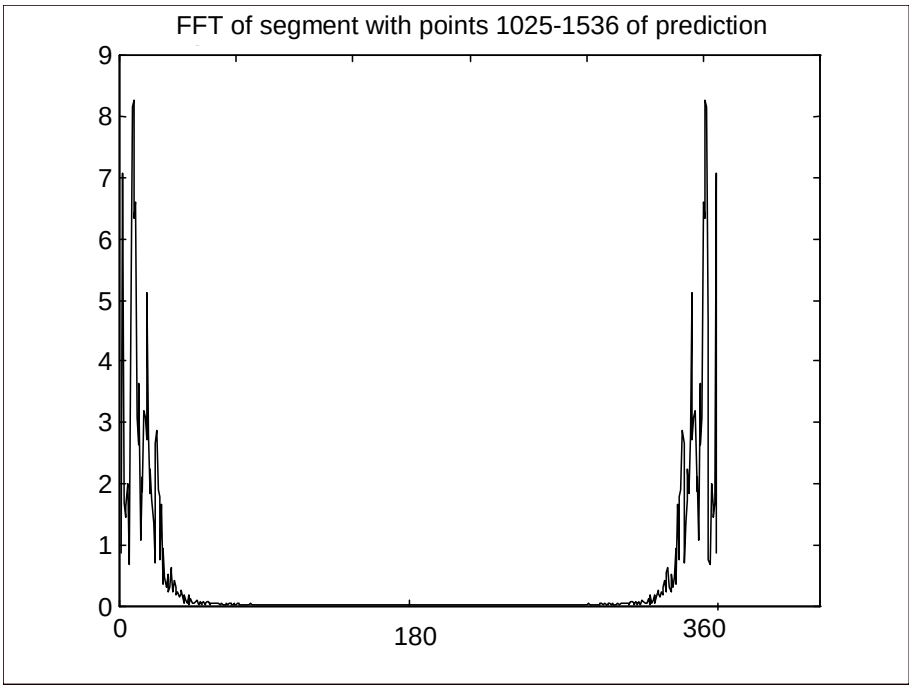


(a)

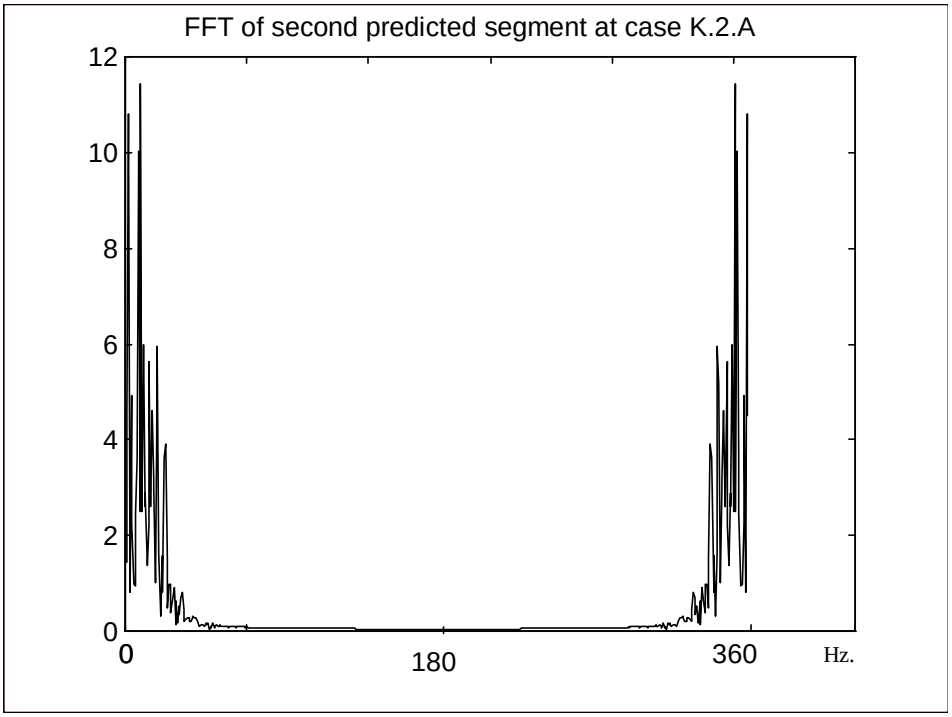


(b)

Figure 5.38. FFT of four consecutive segments in prediction K.2
(a) points 1-512, (b) points 513-1,024



(c)



(d)

Figure 5.38 (cont.). FFT of four consecutive segments in prediction K.2.A (c) points 1,025-1,536; (d) points 1,537-2,048

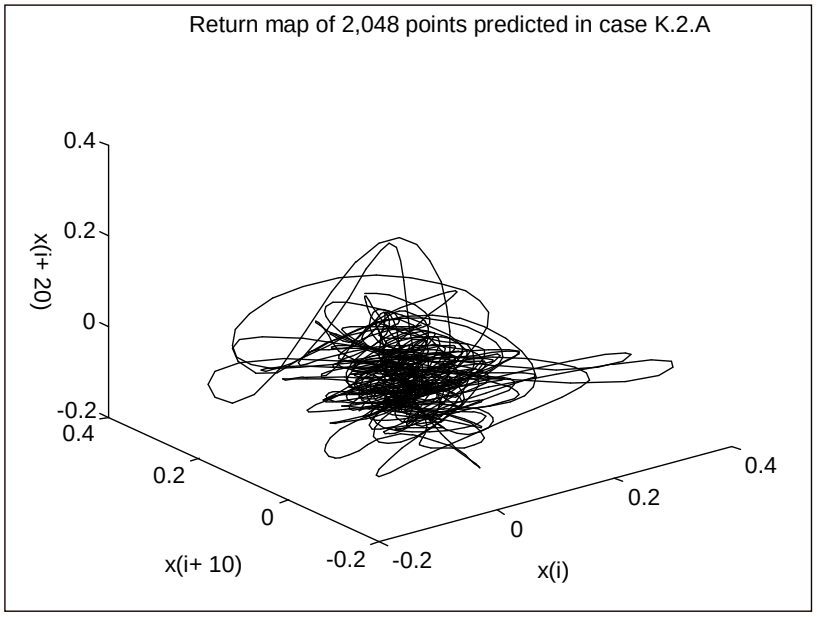
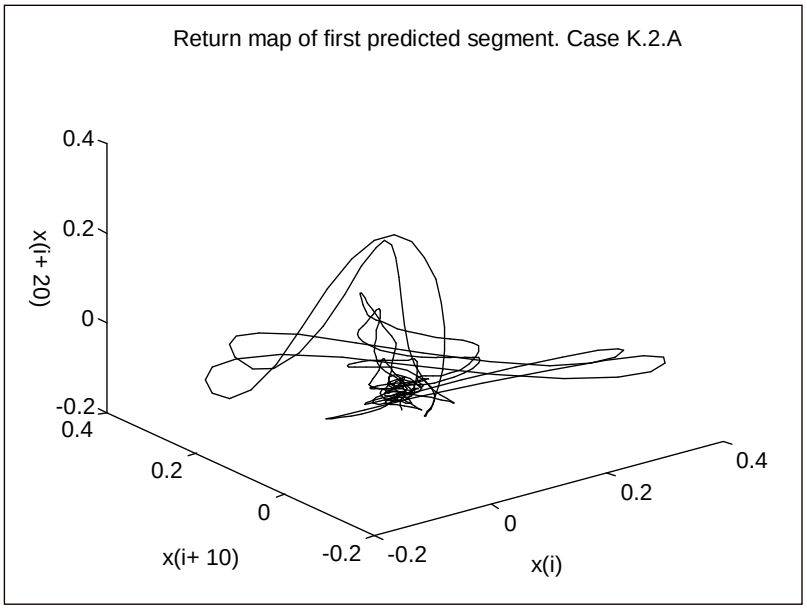
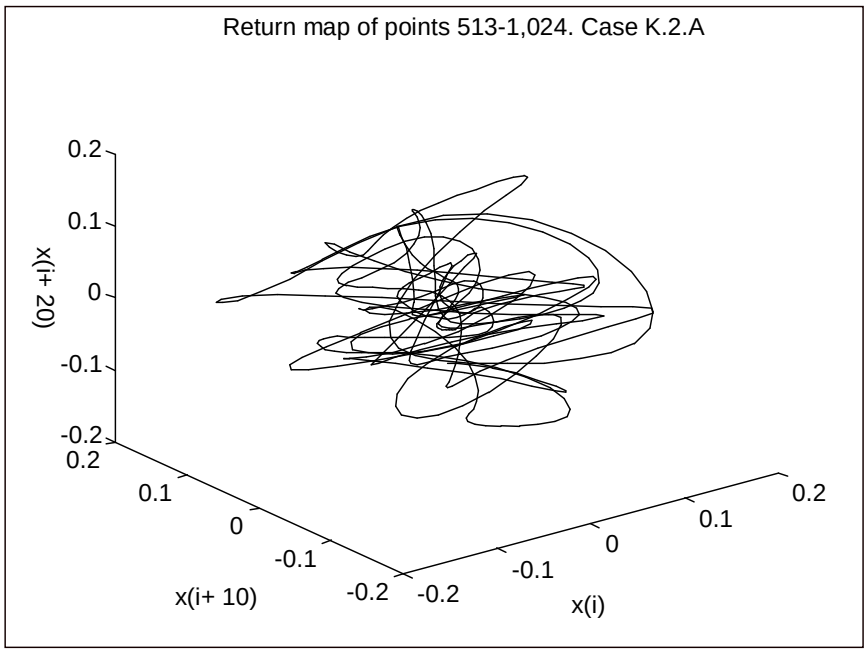


Figure 5.39. Return map of four segments predicted at case K.2.A.

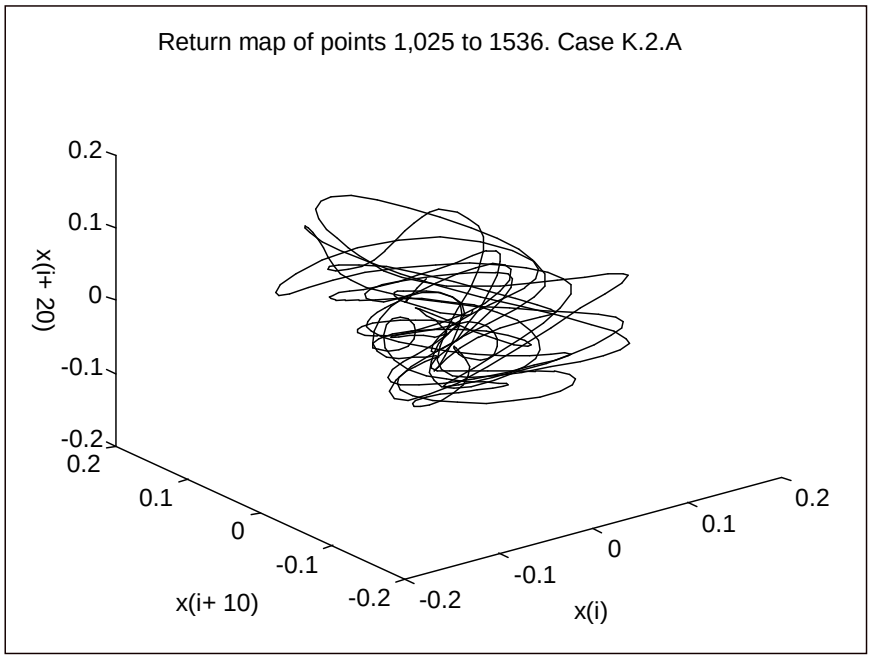


(a)

Figure 5.40. Return maps of segments of 512 points each, predicted at case K.2.A. (a) points 1-512

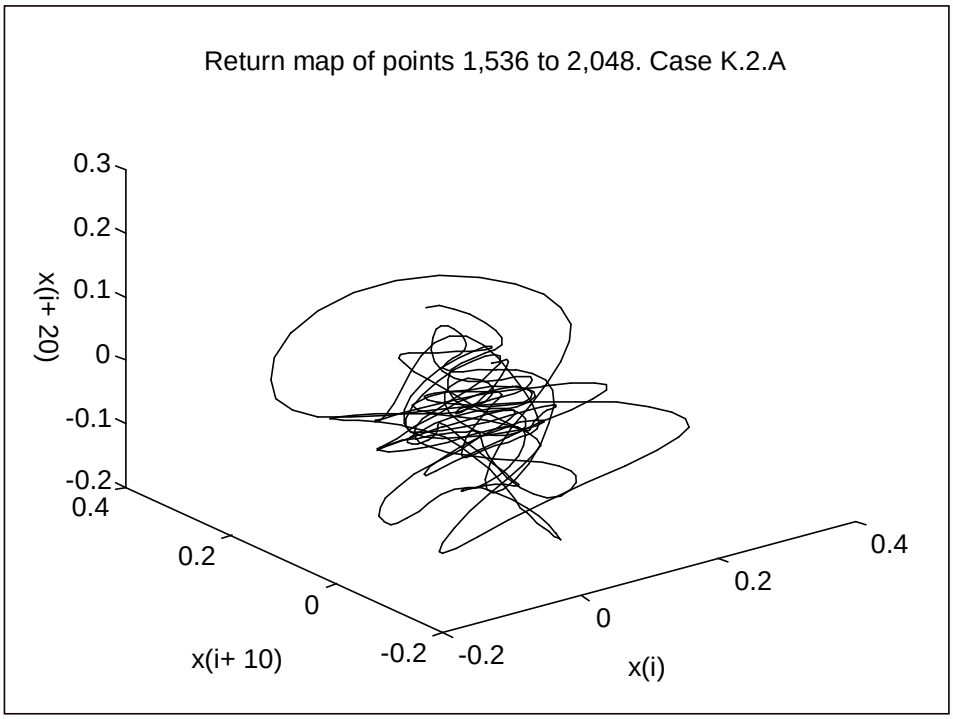


(b)



(c)

Figure 5.40 (cont.). (b) points 513 to 1,024; (c) points 1025 to 1536



(d)

Figure 5.40 (cont.). (d) points 1,537 to 2,048

5.4.3. Case K.2.B A predictor with hidden layers for Mackey-Glass Data

The same network described in past section was used to predict data generated by the Mackey-Glass equation (equation 2.8), which is chaotic., but simpler than an electrocardiogram. In this example, 210 points of the data were used for training. After 31,000 epochs, an MSE of 8.8E-1 was reached and the network got stuck in a local minimum. Figure 5.41 shows the first segment of prediction. Figure 5.42 shows the prediction of 4 segments, together with the expected signal for the same number of segments. The prediction resembles the expected signal, even though it is shifted in time. However, this is expected because the signal is chaotic.

The maximum LE of first predicted segment was 0.0374±0.006; the one corresponding to four segments was 0.0845±0.005; the maximum LE of the training signal is

0.0334±0.003. These results show an excellent representation of the dynamics of the system during the first predicted segment, and a good representation during the prediction of four predicted segments.

Figure 5.43 shows the return map of this prediction. The return map of the predicted signal resembles the shape of the attractor expected for this kind of data (see Figure 2.8). Therefore, this predictor seems to be acquiring the dynamic embedded in this chaotic data set, and probably, as in case K.2.A., the inaccuracy of its outputs is due to the feedback of inexact values to feed the network.

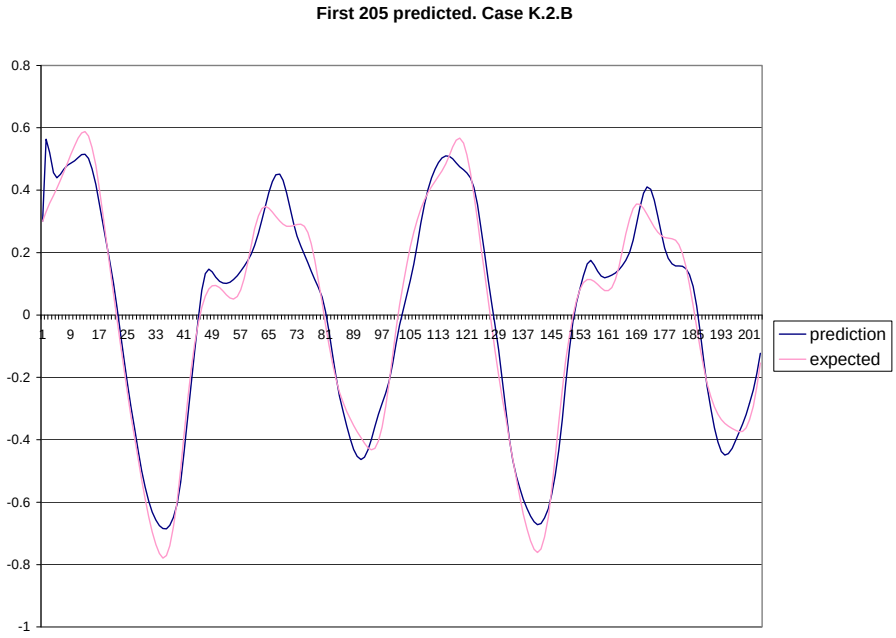


Figure 5.41. First segment of prediction for case K.2.B.

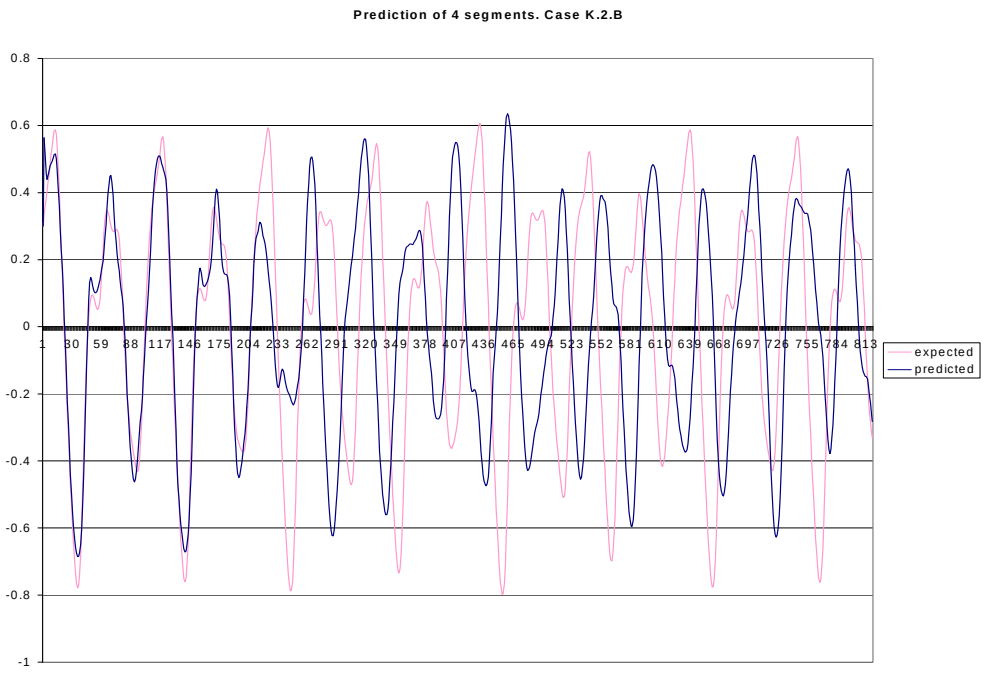


Figure 5.42. Four segments of prediction for case K.2.B.

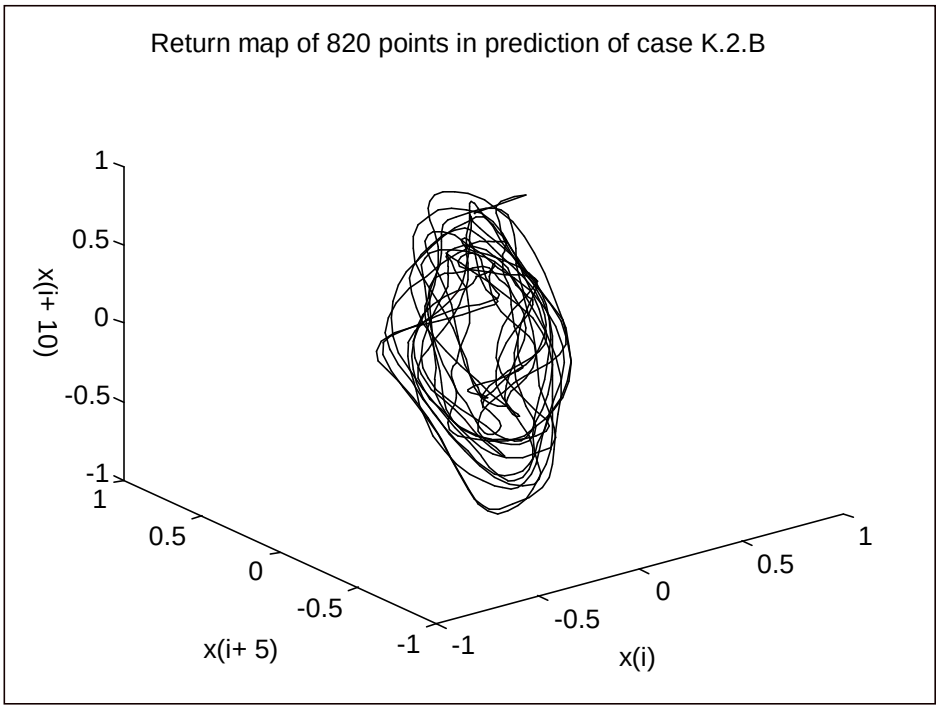


Figure 5.43. Return map of four predicted segment in case K.2.B.

CHAPTER VI
CONCLUSIONS

The main objective of this research was to include information about the Lyapunov exponents of a chaotic time series in a complex network (a recurrent neural network built with harmonic generators), as a way to improve the long-term prediction capabilities of the network when trained using electrocardiograms. Harmonic generators are 3-node fully-connected recurrent neural networks previously trained to generate sine functions with specific frequencies; Lyapunov exponents are invariant measures of the exponential divergence of several trajectories of a dynamical system. Electrocardiograms are time series with positive maximum Lyapunov exponents which make them chaotic if not at least very complex and hence, a real challenge for prediction.

It is known that a function approximated by a feed-forward network can contain the same Lyapunov exponents as the unknown dynamical system that generated the observations used to train such a network (Gencay and Dechert 1992). Using this idea, the concept of Lyapunov exponent was implanted in a complex predictor, combining the topology of feed-forward networks with harmonic generators.

The main conclusion of this research is that the information embedded by the Lyapunov exponents when implanted in a complex network using this fashion is not enough to allow this network to completely learn the dynamics of the system. However, it was also found that such information is useful in some way, as it is explained below.

A feed-forward network approximating a function requires that values of that function at past times be fed to it. For this reason, feedback is necessary during long-term prediction in this case. For the cases reported in this work, it was found that, even though the harmonic generators controlled the oscillation of the generated time series, the error produced by the feedback of outputs accumulated fast, making the prediction diverge very soon. Case K.2.A (section 5.4.2), showed that when this predictor is fed with original observations, that is, with accurate data, it is able to generate very well the attractor defining the system (Figure 5.40 (a)). However, when receiving feedback, it got lost very soon (Figure 5.40 (b) to (d)).

A predictor without external inputs, as the one defined at case C.3.A (section 5.1.3), is able to keep longer a shape in its attractor (Figure 5.10 (a) to (d)) with some resemblance to the expected for an ECG, when compared to a predictor with external inputs, as in case K.2. This is also noticed when comparing Figures 5.37 (b), corresponding to case K.2 with external inputs, with Figure 5.13 (b), corresponding to C.3.A without external inputs.

An important result of this research was the inclusion of time-constant weights in the complex model. Time constants are parameters able to control the amount of modifications in the values of a neuron from time t to time $t+1$. In the original complex model, this value was kept constant for all nodes during the training stage. The definition of one adaptive value for each node, (Pearlmutter 1990), increased the performance of the predictors tested here. This improvement is clear comparing cases C.0 (without time constant weights, section 3.3.1.4) with C.3.A. (using time constant weights, section 5.1.3). The MSE of C.3.A is greater that the one obtained by C.0; however, the long term prediction at Case C.3.A obtained a maximum LE (3.92±1.11) nearer to the expected value (3.23±0.27) when compared to the LE obtained by case C.0 (1.63±0.75). Besides, the return map of case C.3.A presents an attractor with a geometrical shape more uniform that the one obtained by case C.0. The improvement due to time constant weights was noticed only for recurrent networks; for the cases F.1 (sections 5.3.1 and 5.3.2) and F.2 (section 5.3.3), where only feed-forward connections are involved, the time constants made impossible to train the network using back-propagation through time, due to instability in calculation of the value of z (equation B.10).

The harmonic generators were found to be powerful tools for driving and keeping under control the oscillations of the networks in long-term predictions. When a predictor was constructed without using these sub-networks, but just using feedback if a feed-forward network, the results were completely inadequate. This is demonstrated at case F.1.B (section 5.3.2). This network was very good approximating the ECG signal when fed with original observations (Figures 5.24 and 5.25); the MSE was 1.4E-3. The LE of this segment was 19.34±5.29, which is very near to the LE of the training signal (19.7±5.12). However, the same predictor was not able to generate any long-term prediction (Figure 5.26). It was found that the differences between the predicted and expected values were unacceptable after the sixth point predicted when using feedback (Figure 5.27). This predictor worked

very well when generating a function with only one frequency component. Figure 5.23 shows that this predictor, when trained with 99 points of a sine function, was able to reproduce such a function quite accurately for 1,500 points.

The use of recurrent connections in the complex model and in its derived models played an important role in the performance of long-term predictors. In case H.1 (section 5.2), it was found that when the recurrence was eliminated from all nodes except the ones belonging to harmonic generators, the performance of the predictor decreased; it is discovered when comparing the long-term predictions and return maps of case C.3.A (fully recurrent network, section 5.1.3) with case H.1. The results in case H.1 look periodical with peaks of similar amplitudes among them, which does not resemble an ECG (figure 5.18). It resulted interesting that the maximum LE of the long term prediction of H.1 (2.96 ± 0.52) is similar to the LE of the training signal (3.23 ± 0.27). This shows that the fact of embedding the LE invariant in the neural network is not enough to predict the dynamical system.

The down-sampling of the training series did not improve in the performance of the predictors tested in this work. Actually, it seemed to affect the performance in a negative way. This can be corroborated when comparing cases C.0 (section 3.3.1.4) which uses the original training signal, with case C.1 (section 5.1.1) which uses a down-sampled signal; both using the same original complex network; the MSE of cases C.0 (7.1E-3) is much smaller than the MSE obtained in cases C.1 (1.47E-2). Also some decline in performance is noticed when comparing Figure 5.3, the return map generated by case C.0, with Figure 3.7, the return map generated by case C.1. The same changes in performance are noticed between cases C.3.A and C.3.B which both work with the same network but using different training signals, no down-sampled and down-sampled, respectively.

The training algorithm back-propagation through time proved to be a versatile tool in this research. Due to its characteristics, it was used to train feed-forward, hybrid and recurrent neural networks without any modifications. This attribute allowed us to dynamically convert topologies during the training of the hybrid networks. However, a few drawbacks were found with the implementation suggested by Pearlmutter (1990). The main disadvantage was found in the use of the parameter Δt, which drives the time step in the numerical integration of the differential equations used in the algorithm. In some of the

cases reported here, this parameter could not be normalized to one, then several trails had to be done before finding its right value.

The idea of teacher forcing has proved to be useful in some applications, but it did not help to improve the performance of the predictors analyzed in this work. This conclusion is reached when comparing case C.1 (no teacher forcing, section 5.1.1) with case C.2 (teacher forcing, section 5.1.2); both the MSE (1.47E-2 and 1.51E-2) and return maps (Figures 5.3 and 5.6) are the worst obtained in this work.

As in any research, several ideas may be recommended as a continuation of this work. First we suggest to try other implementations of the algorithm back-propagation through time. The algorithms proposed at (Werbos 1994) and (Haykin 1994) could be a good starting point. Another important modifications could be done in the equation defining the dynamics of the neuron (equation B.1). Werbos (1990) proposed the use of the outputs of neurons in previous times *t-1 , t-2* an so on, during the calculation of the output of a neuron at time *t*. Weights are associated to this previous outputs, controlling its effect on the dynamics. However this modification may compromise the learning speed of the network. Other interesting modification could be to train the network defined for cases K.1 and K.2 using sometimes its own output values as external inputs, and some other times inputs coming from the training signal. This should be done after the network has trained for a while using the values coming from training.

As it was hundreds of years ago, the accurate prediction of the future continues to be a not-yet solved but fascinating problem. The recent advances in non-linear dynamic theory, artificial neural network and parallel systems, as well as the fast increase in the power of computers, may allow us to find useful solutions to these kinds of problems in the near future.

BIBLIOGRAPHY

Abarbanel, Henry D. I., Reggie Brown and James Kadtke. "Prediction in Chaotic Nonlinear Systems: Methods for Time Series with Broad Band Fourier Spectra." Physical Review A, Vol. 41, pp. 1782-1807, 15 February 1990.

Abarbanel, Henry D. I., Reggie Brown, John J. Sidorowich and Lev Sh. Tsimring. "The Analysis of Observed Chaotic Data in Physical Systems," Reviews of Modern Physics, Vol. 65, No. 4, pp. 1331-1392, October 1993.

Albert, David E. "Chaos and ECG: Fact and Fiction." Journal of Electro-cardiology, Vol. 24 supplement, pp. 102-106, 1990.

Babloyantz, A. and D. Destexhe. "Is the Normal Heart a Periodic Oscillator?," Biological Cybernetics, Vol. 58, pp. 203-211, 1988.

Banbrook, M., G. Ushaw and S. McLaughlin. "How to extract Lyapunov Exponents from Short and Noisy Time Series," IEEE Transactions on Signal Processing, Vol. 45, No. 5, pp. 1378-1382, 1997.

Barna, György and Ichiro Tsuda. "A new Method for Computing Lyapunov Exponents," Physics Letters A, Vol. 175, pp. 421-427, 1993.

Brockwell, Peter J. and Richard A. Davis. Time Series Theory and Methods. Second Edition. Springer Editors, New York, 1991.

Brown, Reggie and Paul Bryant. "Computing the Lyapunov Spectrum of a Dynamical System From an Observed Time Series," Physical Review A, Vol. 43, No. 6, pp. 2,787-2,806, March 15, 1991.

Burrs, C. Sidney, J. H. McClellan, Alan V. Oppenheim, Thomas W. Parks, Ronald Oppenheim, Alan V and Ronald W. Schafer. Discrete Time Signal Processing, Prentice Hall.

Burrs, C. Sidney, James H. McClellan, Alan V. Oppenheim, ThomasW. Parks, Ronald W. Schafer and Hans W. Schuessler. Computer-based Exercises for Signal Processing using MATLAB, Curriculum Series, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.

Casaleggio, A. S Braiotta and A. Corana. "Study of the Lyapunov Exponents of ECG Signals from MIT-BIH Database," Computers in Cardiology, IEEE Press, pp. 697-700, 1995.

Casaleggio, A., S. Cerutti and M.G. Signorini. "Study of the Lyapunov Exponents in Heart Rate Variability Signals," Methods of Information in Medicine, Vol. 36, No. 4-5, 1997.

Casdagli, M. "Nonlinear Prediction of Chaotic Time Series," Physica D, Vol. 35, pp. 335-356, 1989.

Christiasen, B. T. ECG Time Series Prediction with Neural Networks. Master's thesis, Department of Computer Science, Texas Tech University, Lubbock, TX, August 1995.

Corwin, Edward M. Chaos and Learning in Recurrent Neural Networks. Ph.D. dissertation in Computer Science, Texas Tech University, Lubbock, TX, 1995.

Denton, T. A. (a). "Fascinating Rhythm: A Primer on Chaos Theory and its Application to Cardiology," Journal of Electrocardiology, Vol. 24, Supplement, pp. 84-90, December 1990.

Denton, Timothy A., George A. Diamond, Steven S. Khan, and Hrayr Karagueuzian (b). "Can the Techniques of Nonlinear Dynamics Detect Chaotic Behavior in Electrocardiograph Signals?" Journal of Electrocardiology, Vol. 24, Supplement, pp. 84-90, 1990.

Epstein, Joshua M. Nonlinear dynamics, Mathematical Biology and Social Science, Lecture Notes, Vol. 4, Santa Fe Institute, Addison-Wesley Publishing, Reading, MA, 1997.

Garfinkel, Alan; Mark L. Spano, William L. Ditto, James N. Weiss. "Controlling Cardiac Chaos," Science, Vol. 257, pp. 1230-1235, 28 August 1992.

Gencay, Ramazan and W. Davis Dechert. "An algorithm for the n Lyapunov exponents of an n-dimensional unknown dynamical system," Physica D. Vol. 59, pp. 142-157, 1992.

Glass, Leon. "Complex Cardiac Rhythms," Nature, Vol. 330, No. 24/31, pp. 695-696, December 1987.

Glass, L. and M. C. Mackey. From Clocks to Chaos, Princeton University Press, Princeton, New Jersey, 1988.

Glass, Leon; Peter Hunter, Andrew McCulloch, editors. Theory of Heart, Springer-Verlag, New York, 1991.

Gómez-Gil, Pilar and W.J.B. Oldham. "Recurrent Neural Networks as a Tool for Modeling and Prediction of Electrocardiograms," Proceedings of the World multi-conference on Systemics, Cybernetics and Informatics (SCI'98), Orlando, USA, 1998.

González-F., Jesús J., Ismael Espinosa-E. and Alberto Fuentes-M. "Lyapunov Exponents from Chua's Circuit Time Series Using Artificial Neural Networks," Second International Workshop on Harmonic Oscillators, edited by D. Han and K. B. Wolf, NASA Conference Publications 3286, Scientific and Technical Information Branch, 1995.

Gourieroux, Christian and Alian Monfort. Time Series and Dynamic Models, Translated and edited by Giampiero Gallo, University Press, Cambridge, 1997

Greenberg, Michael D. Foundations of Applied Mathematics, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

Harvard-MIT Division of Health Sciences and Technology. The MIT BIH Arrhythmia Database CD-ROM. Second Edition, Biomedical Engineering Center, Cambridge, MA, August 1992.

Hayashi, Yukio. "Oscillatory Neural Network and Learning of Continuously Transformed Patterns," Neural Networks, Vol. 7, No. 2, pp. 219-231, 1994.

Haykin, Simon. Neural Networks. A Comprehensive Foundation, Macmillan College Publishing Co., New York, 1994.

Hornik, K., M. Stinchcombe and H..White. "Universal Approximation of an Unknown Mapping and its Derivatives using Multi-layer Feed-forward Networks," Neural Networks, Vol. 3, pp. 535-549, 1990.

Kaashoek, Johan F. and Herman K. van Dijk. "A Neural Network Applied to the Calculation of Lyapunov Exponents," Econometric Reviews, Vol. 13, No. 1, pp. 123-137, 1994.

Kaashoek, Johan F. and Herman K. van Dijk. "A neural Network applied to the calculation of Lyapunov Exponents," Econometric Reviews, Vol. 13, No. 1, pp. 123-137, 1994.

Kaplan, Daniel T. and Richard J. Cohen. "Is Fibrillation Chaos?" Circulation Research, Vol. 67, No. 4, October 1990.

Karanam, Rajaiah. Prediction of the Behavior Of The Human Heart Using Methods of Non-Linear Dynamics, Master's thesis in Computer Science, Texas Tech University, Lubbock, TX, May 1996.

Lao, Xueying. Time Series Prediction on Electrocardiogram Data by Radial Basis Function Neural Networks, Master's thesis in Computer Science, Texas Tech University, Lubbock, TX, December 1994.

Lapades, R. and R. Farber. "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling," Technical Report LA-UR87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico, 1987.

Logar, Antonette M. Recurrent Neural Networks and Time Series prediction, Ph.D. Dissertation in Computer Science, Texas Tech University, Lubbock, TX, December 1992.

Oldham, W. and Pilar Gómez-Gil. "Modeling and Prediction of Time Series Using Recurrent Neural Networks: an Application to ECG," Proceedings of the "Second Joint Mexico-US International Workshop on Neural Networks and Neurocontrol Sian Ka'an '97," Quintana Roo, México, August 1997.

Oldham, W. and Pilar Gómez-Gil. "Application of Recurrent Neural Networks for the Prediction on the Behavior of Biological Oscillators," Proceedings of the VII International Conference on Electronics, Communications and Computers. CONIELECOMP 97," Puebla, México, February 1997.

Oppenheim, A..V. and R..W. Schafer. Discrete Time Signal Processing, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

Parker, T. S and L.O. Chua. Practical Numerical Algorithms for Chaotic Systems, Springer-Verlag, New York, 1989.

Pearlmutter, B. A. "Learning State Space Trajectories in Recurrent Neural Networks," Neural Computation, Vol.1, pp. 263-269, 1989.

Pearlmutter, B. A. "Dynamic Recurrent Neural Networks," Technical Report CMU-CS-90-196, School of Computer Science, Carnegie Mellon University, Pittsburgh, December 1990.

Press, William H., Saul A.. Teukolsky , William T. Vetterling and Brian P. Flannery. Numerical Recipes in C. The Art of Scientific Computing. Second Edition, Cambridge University Press, 1992.

Príncipe, Jose C. and Jyh-Ming Kuo. "Dynamic Modeling of Chaotic Time Series with Neural Networks," Advances in Neural Information Processing Systems 6, edited by Cowan, Tesauro and Alspector, Morgan Koufmann, pp. 311-318, 1994.

Príncipe, Jose C., Ludong Wang and Jyh-Ming Kuo. "Non-linear Dynamic Modeling with Neural Networks," Proceedings of the First European Conference on Signal Analysis and Prediction, 1997.

Proakis, John G. and Dimitris G. Manolakis. Digital Signal Processing. Principles, Algorithms and Applications, third Edition. Prentice Hall, Englewood Cliffs, New Jersey, 1996

Rosenstein, Michael T., James J. Collins and Carlo J. De Luca. "A practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets," Physica D, Vol. 65, pp. 117-134, 1993.

Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning internal representations by Error Propagation," In Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. I, Bradford Books, Cambridge, MA, 1986.

Sattin, Fabio. "Lyap: A Fortran 90 Program to Compute the Lyapunov Exponents of a Dynamical System from a Time Series," Computer Physics Communications, Vol. 107, No. 1-3, pp. 253-257, 1997.

Stein, Kenneth M., Neal Lippman and Paul Kligfield. "Fractal Rhythms of the Heart," Journal of Electrocardiology, Vol. 24, supplement, pp. 72-76, 1990.

Takens, F. "Detecting Strange Attractors in Turbulence," Dynamical Systems and Turbulence, edited by D. A. Rand and L. S. Young, Springer-Verlag, Berlin, 1981.

Tong, Howell. Non Linear Time Series. A Dynamical System Approach, Oxford Science Publications, Clarendon Press, Oxford, 1993.

Wang, Lipo and Daniel L. Alkon. Artificial Neural Networks: Oscillations, Chaos and Sequence Processing, IEEE Computer Society Press, Washington DC, 1993.

Werbos, Paul J. The Roots of Backpropagation. From Ordered Derivatives To Neural Networks And Political Forecasting, John Wiley and Sons, New York, 1994.

Werbos, Paul J. "Backpropagation Through Time: What it Does and How to Do it," Proceedings of the IEEE, Vol. 74, No. 10, pp. 1550-1560, October 1990.

Williams, Ronald J. and David Zipser. "Experimental Analysis of Real-time Recurrent Learning Algorithm," Connection Sciences, Vol. 1, No. 1, 1989.

Wolf, Alan, Jack B. Swift, Harry L. Swinney and John A. Vastano. "Determining Lyapunov Exponents from a Time Series," Physica 16D, pp. 285-317, 1985.

APPENDIX A
CALCULATION OF THE MAXIMUM LYAPUNOV EXPONENT OF A TIME SERIES

Following is a structured version of the algorithm defined at (Wolf et al. 1995) to calculate the largest non-negative Lyapunov Exponent from a time series. For more details about Lyapunov exponents and this algorithm see section 2.7.

1. Read input parameters and time series
2. Construct an attractor
3. Set *ind* = 1. Consider 1st. point as fidutial point.
4. Find nearest point to fidutial point. Let *d1* ≡ its distance to fidutial point.
   ind2 ≡ its index.

sum = 0 *; its* = 0

5. Repeat
   - 5.1 *pt1* = *z*[*ind* + *evolv*] ; *pt2* = *z*[*ind2* + *evolv*]
   - 5.2 Let *df* ≡ distance between *pt1* and *pt2*
   - 5.3 *its* = *its* + 1 ;
   - 5.4 $sum = sum + \frac{\log(\frac{df}{di})}{evolv * dt * \log(2.0)}$
   - 5.5 $zlyp = \frac{sum}{its}$
   - 5.6 Print *zlyp, evolv*its, di, df*
   - 5.7 Look for a replacement of pt2. While looking for replacement:
     - 5.7.1 Among all points, select a point that:
       - a) is not too far or too close to *pt1*. (*dnew*, its distance to *pt1* should: *dnew* <= *zmult*scalmx* and *dnew* >= *scalmn*)
       - b) it is the one with smallest angle between *pt1* and *pt2* and itself. Name this angle as *thmin,* its distance to *pt1* as *dii* and its index as *ind2*

5.7.2 if *thmin* < *anglmx*

5.7.2.1 This is a good point, don't look anymore

else

5.7.2.2 A look at longer distances is needed:

zmult = *zmult* + *1*

5.7.2.3 if *zmult* > *5* the requirements need to be relaxed:

5.7.2.3.1 *anglmx* = *2* anglmx ; zmult* = *1*

5.7.2.3.2 if *anglmx* > *3.14*

then there is no possible point as replacement,

continue with this one..

ind2 = *ind2* + *evolv*

dii = *df*

else

look again with new zmult and anglmx...

endif

endif

end if

end of while

5.8 *ind* = *ind* + *evolv*

5.9 if *ind* < *npt di* = *dii*

until (*ind*>*npt* or *ind2* > *npt*) (until there is no more data to use)

6. End.

APPENDIX B
THE TRAINING ALGORITHM BACKPROPAGATION THROUGH TIME

Backpropagation through time (BPTT) is an algorithm that attempts to minimize the error obtained over a period of time between the output of a neuron and the desired value of such output. It was originally proposed by Werbos (1990). Some other neurons besides the output will be required in order to represent the dynamics of the system. The total error in an output neuron is represented by:

$$E = \int_{t_0}^{t_1} (y(t) - d(t))dt\,, \tag{B.1}$$

where *y(t)* is the "real output" obtained by the output neuron and *d(t)* is the desired one. BPTT looks for a minimization of the square root of such value E.

In a fully-connected recurrent neural network, the dynamics of a neuron y_i can be described by the equation (Pearlmutter 1989):

$$\frac{dy_i}{dt} = -y_i + \sigma(x_i) + I_i\,, \tag{B.2}$$

which, using recurrent equations is:

$$yi(t + \Delta t) = yi(t) + \Delta t[-yi(t) + \sigma(xi) + Ii]\,, \tag{B.3}$$

where

$$x_i = \sum_j w_{ji} y_j\,. \tag{B.4}$$

x_i represents the total input to the i-th neuron coming from other neurons, I_i is an external input to neuron *i,* w_{ij} is the weighted connection from neuron *i* to neuron *j*, and $\sigma(x)$ is an arbitrary differentiable function, normally a sigmoid, for example:

$$\sigma(x) = (1 + \exp(-x))^{-1}. \tag{B.5}$$

Pearlmutter found that the modification to the weights (learning) can be described by the equation:

$$\Delta w_{ij} = -\eta \sum_t y_i \sigma'(x_j) z_j \Delta t\,, \tag{B.6}$$

where η is a learning coefficient. Using a discrete notation, the value of z_i is given by:

$$z_i(t) = z_i(t + \Delta t) - \Delta t \frac{dz_i}{dt}, \tag{B.7}$$

where:

$$\frac{dz_i}{dt} = z_i - e_i - \sum_j w_{ij}\sigma'(x_j)z_j, \tag{B.8}$$

and

$$e_i(t) = y_i(t) - desired_i(t), \tag{B.9}$$

in discrete notation z is:

$$\begin{aligned} z_i(t) = z_i(t + \Delta t) - \Delta t[z_i(t + \Delta t) - e_i(t + \Delta t) \\ - \sum_j w_{ij}\sigma'(x_j(t + \Delta t))\, z_j(t + \Delta t)], \end{aligned} \tag{B.10}$$

$desired_i(t)$ is the desired value for node *i* at time *t*. Notice that the integration of z_i is calculated backwards. Following is a description of the algorithm that implements this method in a complex network.

B.1 Algorithm to train a Recurrent Neural Network using BPTT

Given a discrete signal *d(nΔt)* over a period of time [0, *FT*], the Pearlmutter's implementation of BPTT to train one output node of a complex network to learn *d(t)* results in the following algorithm:

1. Get parameters describing the architecture of the network, where *n* = total number of nodes.
2. Get the desired values *d[i][t]* for each i-th output node" in the network at each time *t* in the trajectory, assuming that such values are separated by a time period Δt. The size of such trajectory will be refereed from now as *FinalTime*
3. Get or calculate the initial conditions for the outputs of all nodes, that is *y[i][0]*
   for all nodes $i = 0 \ldots n\text{-}1$
   - 3.1 If this network is learning by first time then
     - 3.1.1. Initialize weights of with small random numbers and/or
     - 3.1.2 Set any previously calculated weights corresponding to any sub-networks

   else

3.1.3 Read weights of network corresponding to past runnings

4. Repeat until *TotalError* is small enough or until desired number of epochs (each step in this loop is called an *epoch*)

4.1 For t=1 to *FinalTime* (and each time t in the trajectory)

4.1.1. For i = 0 to n-1 (each node in network),

calculate output of i-th node at time t as:

$$y[i][t] = [1 - \Delta t]\, y[t - \Delta t] + \Delta t \sigma(x[i][t]) + \Delta t I[i][t],$$

where $x[i][t] = \sum_j w_{ji} y[j][t-1]$

4.2 For i=0 to n-1

4.2.1 for t=1 to *FinalTime-1*

calculate the error-by-node as:

$$e[i][t] = \begin{cases} y[i][t] - d[i][t] & \text{if } i = \text{ output node} \\ 0 & \text{if i is not an output node} \end{cases}$$

4.3 Calculate the Total error at this epoch as:

$$TotalError = \frac{1}{2} sqrt\left(\sum_t \sum_i e[i][t] \right)$$

4.4 For t = *FinalTime*-1 to 1 (propagation of error backwards)

4.4.1 For i=0 to n-1 (for each node)

$$z[i][t] = \Delta t e[i][t + \Delta t] + [1 - \Delta t] * z[i][t + \Delta t] + \Delta t \sum_j w_{ij} \sigma'(x[j][t + \Delta t]) z[j][t + \Delta t]$$

4.5 Update the *w's*

4.5.1 For i=0 to n-1

45.1.1 For j=0 to n-1

$$w[i][j] = w[i][j] - \eta \Delta t \sum_t y[i][t] * \sigma'(x[j][t]) * z[i][t]$$

5. End.

B.2 Algorithm to Predict the trajectory of a time series

Once that the complex network is trained, or at any time during training, the output of this predictor for a trajectory or arbitrary length can be calculated given only the initial conditions of the net (value of each node at time t=0) and the value of weights. Following is the appropriate algorithm:

1. Read the weights of complex network
2. Read the initial conditions of each node, *y[i][0] , i =1.. n-1*
3. Read the size of the prediction, called as *PredictionTime*
4. For *t*=1 to *PredictionTime*

    4.1 For *i*=0 to n-1 (each node in network),

        4.1.1. Calculate output of i-th node at time t as:

$$y[i][t] = [1 - \Delta t]\, y[t - \Delta t] + \Delta t \sigma(x[i][t]) + \Delta t I[i][t],$$

$$\text{where } x[i][t] = \sum_j w_{ji} y[j][t-1]$$

        4.1.2 If i == n-1 (last output node) then

            4.1.2.1 display *y[i][t]*

5. End.

APPENDIX C
TOPOLOGIES OF THE HYBRID NETWORKS

This appendix contains the connection matrices describing the hybrid networks used in this work. A *hybrid network* is one that may have both recurrent and no recurrent connections between its nodes. It is also possible that some connections may no be present in a hybrid network. A *connection matrix* **C** for a network with *n* nodes is defined as:

$$c[i, j] = \begin{Bmatrix} 1 & \text{if there is a weight from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{Bmatrix},$$

for $i, j = 1, 2 \ldots n$.

Following are the connection matrices for the hybrid networks used at cases H.1, F.1 A and B, F.2, K.1 and K.2. The first row and first column of each table identify the number of node. Bold lines separate "layer" in the network.

C.1 Connection Matrix A

This section describes the hybrid network used for case H.1 (see Figure 5.16). The first layer contains nodes 1 to 21 making the 7 harmonic generators (fully 3-node RNN). Nodes 22 to 28 are hidden nodes, located at the second layer. The last layer contains nodes 29 to 32, which are pseudo-output nodes, and node 33, which is the output node.

Table C.1. Connection Matrix A.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
4	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
5	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
6	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
7	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
8	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
9	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C.2 Connection Matrix B

This section describes the network used at cases F.1 A and B (Figure 5.20). It is a pure feed-forward network with 5 nodes in the input layer, 10 nodes in the hidden layer and one node in the output layer.

Table C.2. Connection Matrix B.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
3	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
4	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
5	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C.3 Connection Matrix C

This section describes the hybrid network used at case K.1 (Figure 5.31). It has 5 input nodes at the first layer; seven harmonic generators in the hidden layer that result in 21 nodes and one output node.

Table C.3. Connection Matrix C.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
3	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
4	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
5	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
6	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C.4 Connection matrix D

This section describes the hybrid network for case K.2. (Figure 5.35). It has 2 dimensions in the first layer: one with 5 input nodes and other with 7 harmonic generators (21 nodes). Both dimensions connect to 7 hidden nodes in the second layer. There is only one node in the output layer.

Table C.4. Connection Matrix D.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
6	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
9	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
12	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	1	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APPENDIX D
SUMMARY OF TRAINING FILES

All training data used in this work came from the database produced by the Harvard-MIT Division of Health Science and Technology Biomedical Engineering Center (Harvard 1992). Only short segments of such files were use for training the networks. Such segments were selected and then filtered, normalized and or down-sampled, as described at chapter IV. Following is a summary of the characteristic of each one:

D.1 File ecg4.fil

This file is a modified version of file a0310_5a.dat, described by Christiasen (1995). The data was originally sampled to a frequency of 360 Hz. First, this signal was converted to mean equal zero. Second, the signal was filtered by a FIR filter of order 40 with cutoff frequencies lying at 0.5 and105 Hz. Last, the magnitude of the signal was normalized to the interval [-0.3122,0.75].

Size: 512 points

Lyapunov exponent: 3.23 ± 0.27

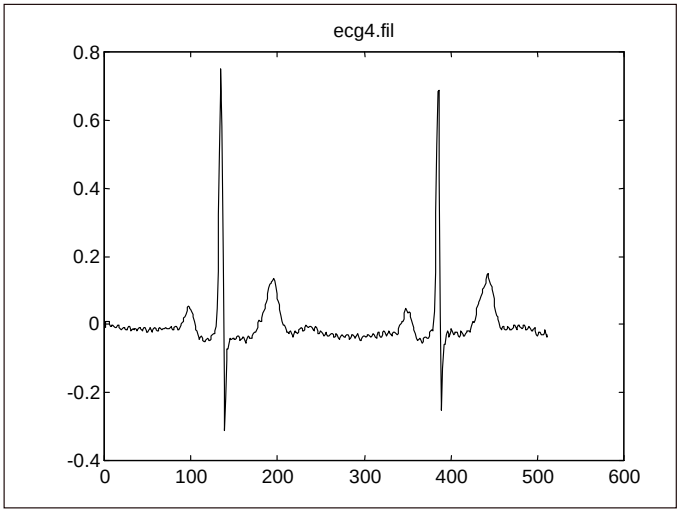


Figure D.1 File ecg4.fil

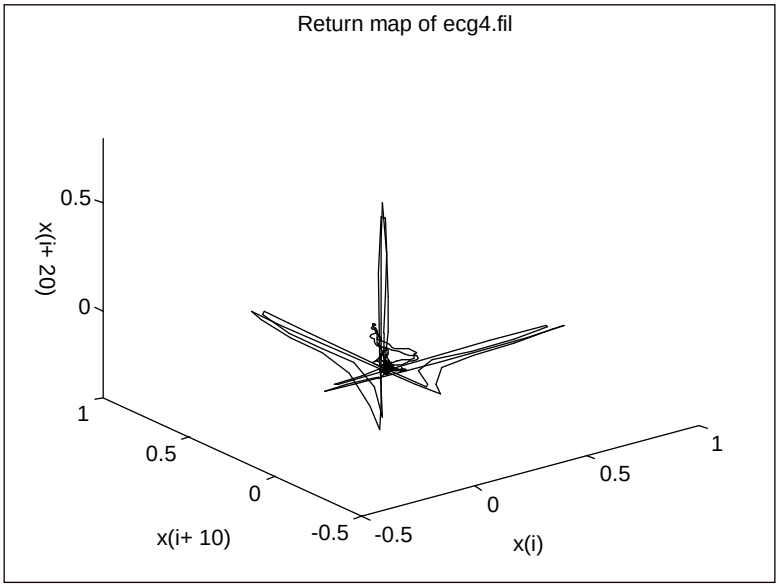


Figure D.2. Return map of ecg4.fil

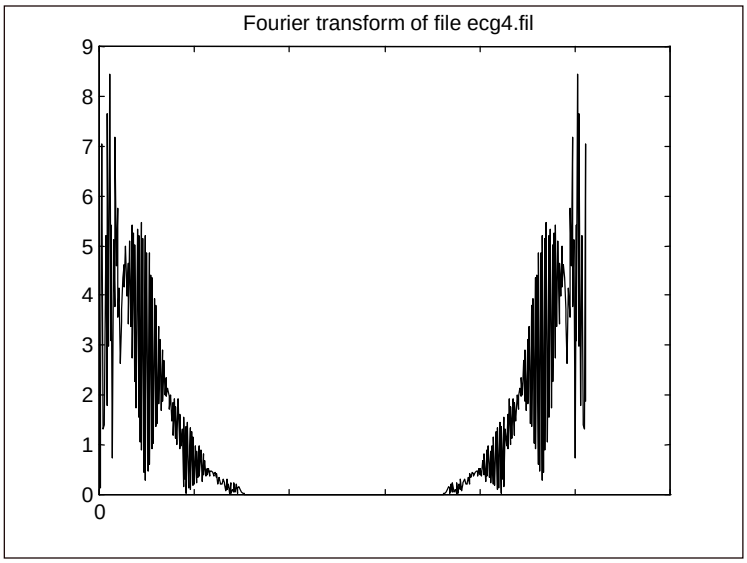


Figure D.3. Fourier Transform of ecg4.fil

D.2 File ecg6n.su4

Similar to ecg4.fil, this is a modified version of a portion of a0310_5a.dat. This signal was converted to mean zero, filtered by a band pass from 0.5 to 45 Hz., and decimated by a factor of 4. After that, the magnitudes of the signal were normalized to values in the range [-0.2175, 0.75].

Size: 128 points

Lyapunov exponent: 10.77 ± 2.94

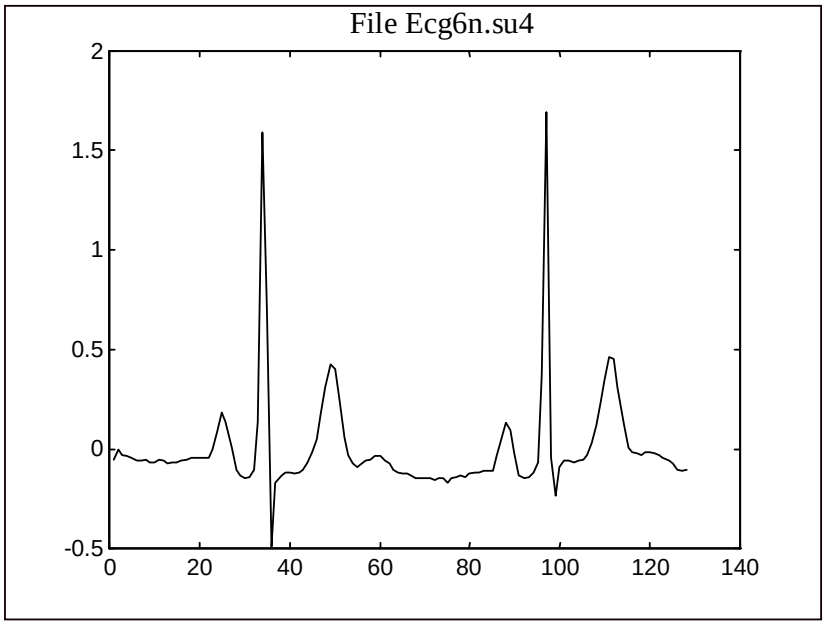


Figure D.4. File Ecg6n.su4

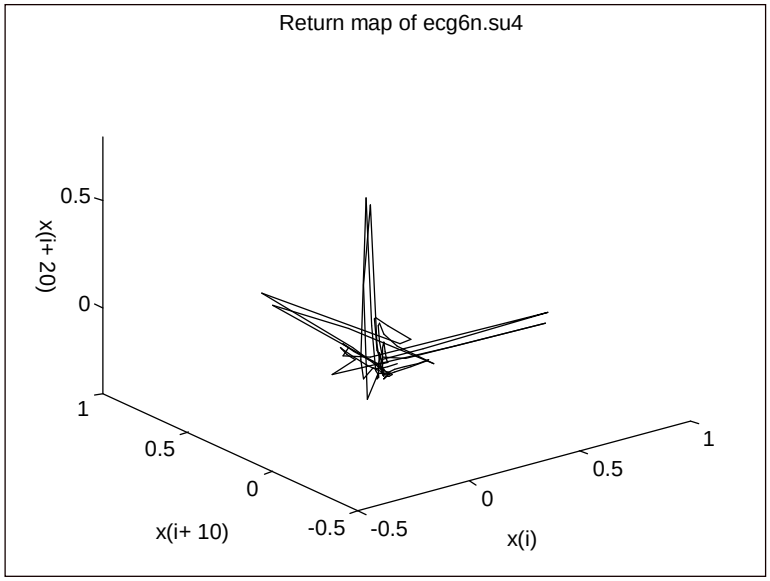


Figure D.5. A Return map of signal ecg6n.su4

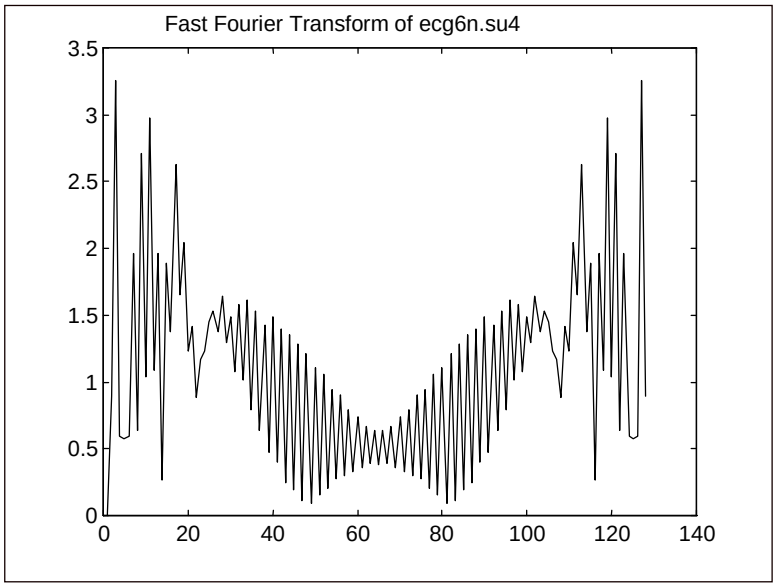


Figure D.6. Fourier Transform of ecg6n.su4

D.3 File A0310z.fil

Size: 475 points

Lyapunov exponent: 19.34 ± 5.25

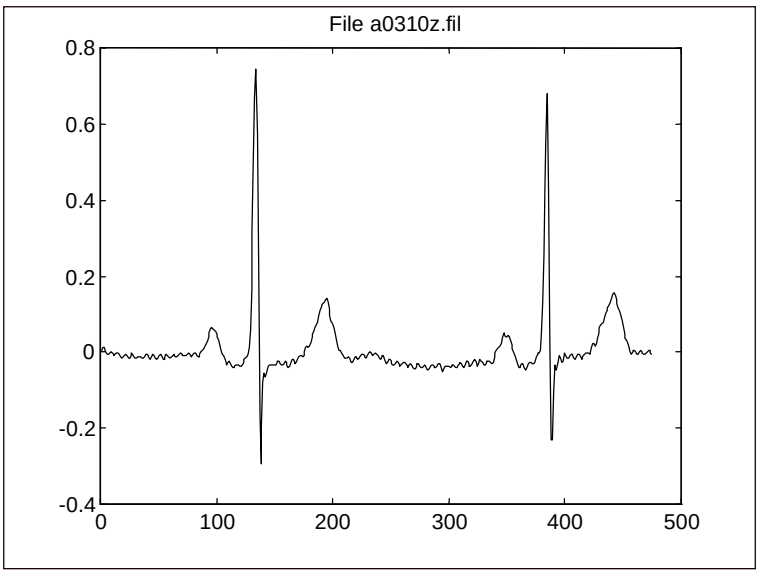


Figure D.7. The 475 points of file a0310z.fil

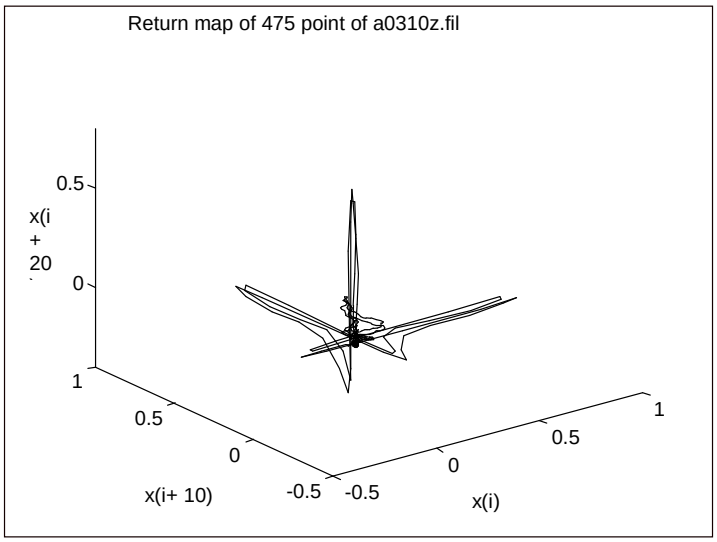


Figure D.8. Return map of 475 points of a0310z.fil

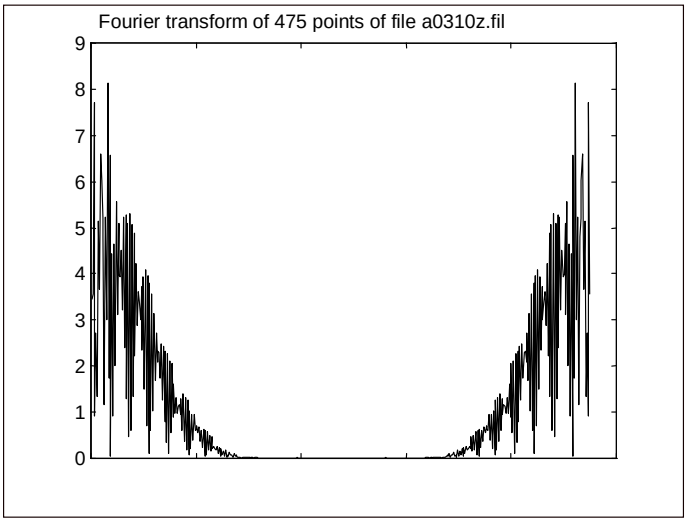


Figure D.9 Fourier transform of 475 points of a0310z.fil

D.4 File good8.dat (Mackey-Glass Data)

Size: 820 points

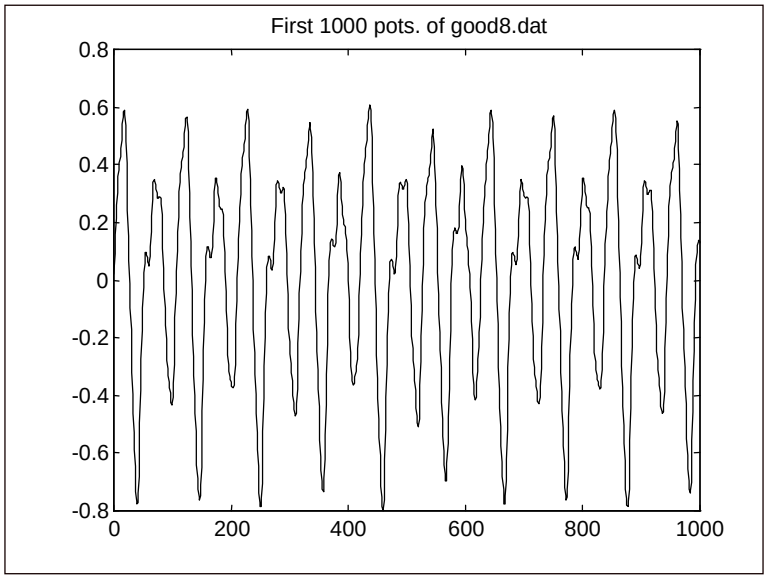


Figure D.10. 1,000 points of Mackey-Glass data

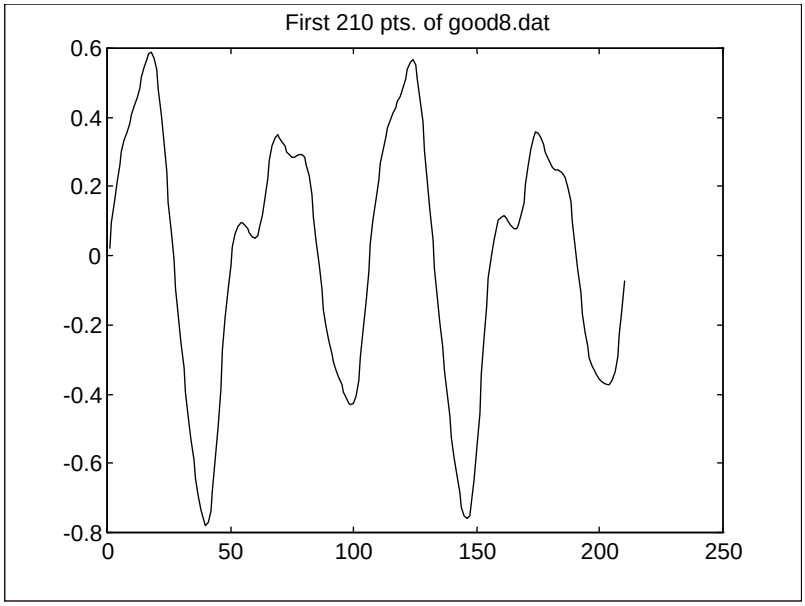


Figure D.11. Training file for Mackey-Glass data

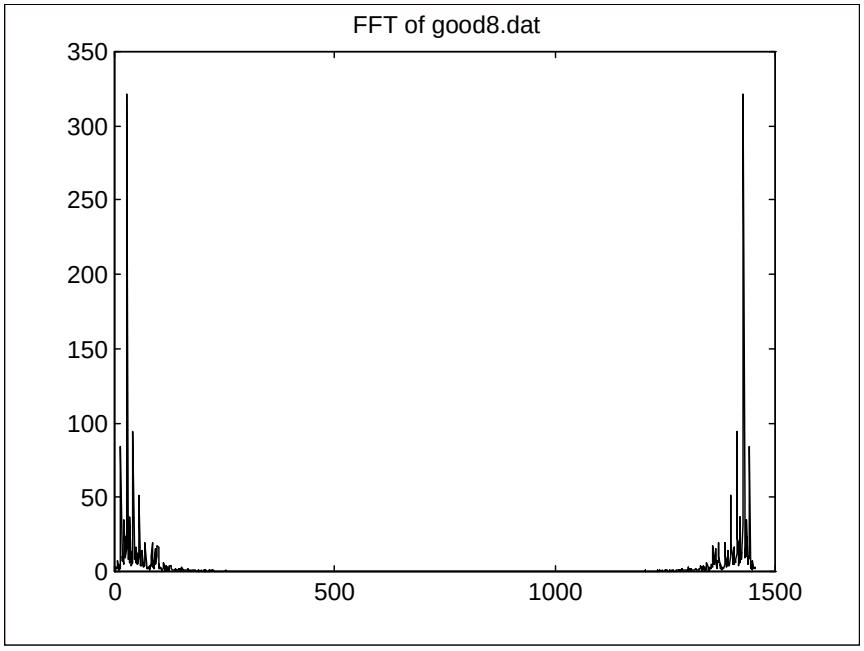


Figure D.12. Fourier transform of 210 points of good8.dat

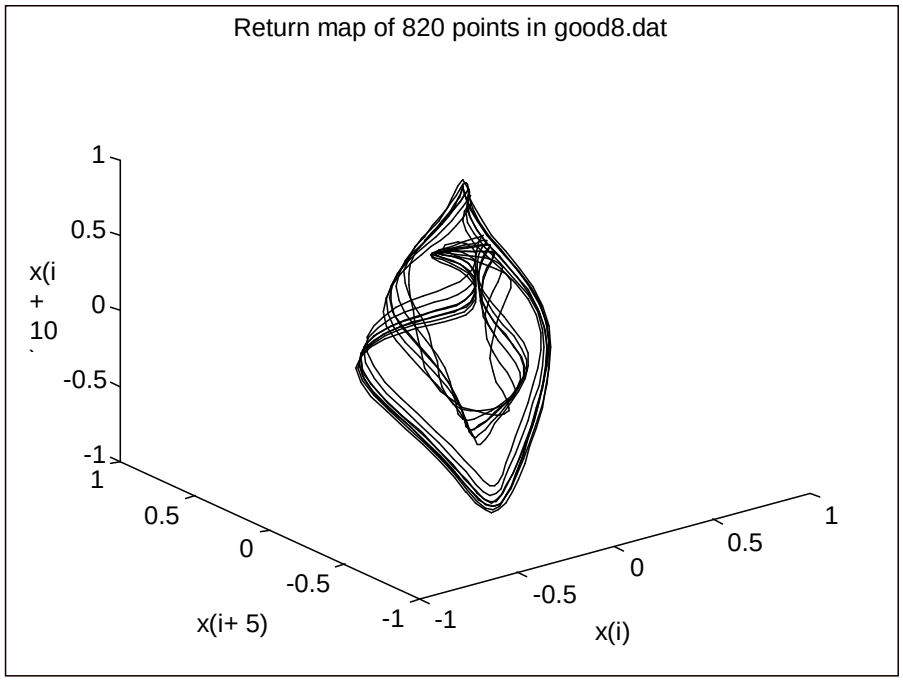


Figure D.13. Return map of 820 points in good8.dat