# SYMBIOTIC ADAPTIVE NEURO-EVOLUTION APPLIED TO CONTROL THE LORENZ SYSTEM FROM TIME-SERIES

**Danniel Kenneth Borges Costa, dannielunb@gmail.com**
**Tito Dias Jr, titodiasjr@gmail.com**
Departamento de Engenharia Mecânica, Universidade de Brasília, Campus Darcy Ribeiro, Asa Norte, Brasília – DF, 70910-900

***Abstract:*** *The main subject of this paper is the chaos control. This can be achieved through applying small disturbances to the system parameters in order to stabilize unstable periodic orbits (UPO). As final result a software has been developed, using Matlab programming language, based on artificial neural networks (ANN) and genetic algorithms (GA) theories, capable of controlling chaotic systems upon desired periodic orbits. The Lorenz dynamical system was used as chaotic model to study the effectiveness of the proposed method.*

***Keywords:*** *Artificial neural networks, Genetic algorithms, Unstable chaotic orbits, Lorenz's atractor.*

## 1. INTRODUCTION

A deterministic system is called chaotic when its time evolution depends on your initial condition sensitively. It implies that two trajectories emerging from different initial conditions diverge exponentially on time. A chaotic deterministic system must be non-linear and, at least, tridimensional.

Chaos makes possible to produce infinity dynamical behaviors, periodic or not, through the same chaotic system, by applying appropriated small disturbances to the system parameters. These disturbances are chosen after a learning time when the relation between the dynamic and some external controls is tested. The chaos controlling idea consists on disturbing a controlling parameter when the natural trajectory lies on a small neighborhood from the desired point (this is warranted by ergodicity), in order to put the trajectory upon a stable part of the system, healing all the divergences and converging the natural time evolution (except non-linearities and noise) to an specific behavior.

This project's main objective was to develop a software capable of controlling chaotic systems upon periodic orbits by choosing its period, using for this negative feedback on a system available variable, based on artificial neural networks and genetic algorithms theories.

## 2. CHAOS ADAPTATIVE CONTROL THROUGH ARTIFICIAL NEURAL NETWORKS AND GENETIC ALGORITHMS

### 2.1. Artificial Neural Networks (ANNs)

Artificial neural networks are composed by systems that reproduce, in a certain way, a human's brain structure. Composed by units (nodes) organized in unidirectional connected layers and associated to weights that storage the information of the model (SUYKENS; VANDEWALLE; DE MOOR, 1996). This information is responsible to weigh input of each network's node (BRAGA; LUDEMIR; CARVALHO, 2000).

The solving problem procedure to ANNs is composed for a learning phase, when several examples are presented to the network in order to catch sufficient characteristics to represent the received information. The ANNs are capable of prescind non explicit information and describe multivariable functions with computational cost growing linearly with the number or variables (DIEDERICH, 1990).

Inspired on scientific knowledgement of the biological neuron, McCulloch and Pitts proposed MCP model. This model is formed by "$n$" input terminals "$x_1, x_2, ..., x_n$", which represent dendrites, and na output terminal "$y$", which represent the axon.

The inputs' weights "$w_1, w_2, ..., w_n$", can be either positive as negative (excitatory or inhibitory contribution, respectively) and its function is measure how important one specific connection contribution is to an specific neuron (WEEKS; BURGESS, 1997), emulating the biological synapses behavior (HOLLAND, 1992). The figure 1 shows a representative diagram of a MCP neuron, its inputs, weights and outputs.

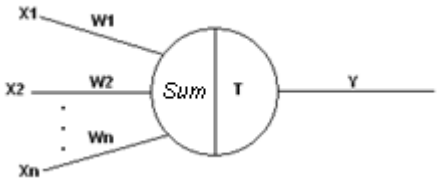


Figure 1. MCP Neuron, where T is the ativation trigger value.

### 2.2. Genetic Algorithms (GAs)

GAs are searching algorithms based on Darwinian's evolution mechanisms: natural selection, genetic recombination (crossover) and mutation (KOZA; KEANE; STREETER, 2003). Using well structured arrays (strings), the idea of most adapted one's survival is applied. These strings exchange pieces of information through mathematical operations (CUNHA, 2006).

At each generation is created a new set of strings (individuals) using part of the most adapted ones from last generation. Mutation processes may happen occasionally in order to generate new parts (with no connection to the existing data) to test solutions out of the original searching region (PRESS, 1986). Genetic algorithms tends to, through generations, obtain the best results exploiting all the available information from the system to chose new analyses points where we hope to get results better than current ones (GOLDBERG, 1989).

GAs are a robust way of searching in unknown spaces (MITCHELL, 1996). Just like a black box where we provide input data and, as result, it returns us an output value without, however, know what happens inside the box (LINDEN, 2006). The figure 2 shows a representative crossover process diagram.

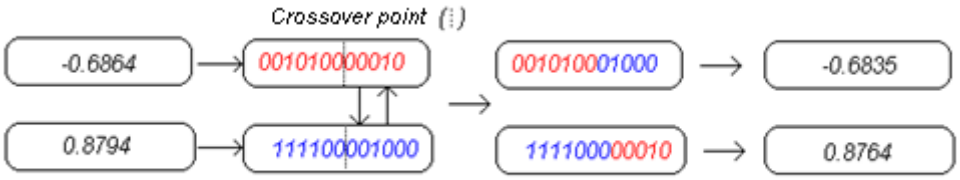


Figure 2. A crossover process.

Figure 3 shows a mutation process diagram.

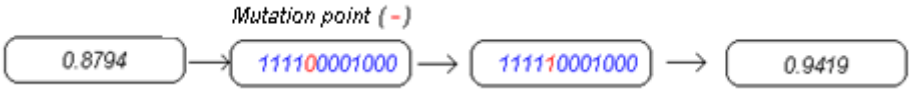


Figure 3: A mutation process diagram.

Here Gas will be used in selection of neurons from ANN between execution loops of the main program. It gives a greater adaptability to the software. We expect that ANNs and GAs, working together, produce a robust and reliable problem's solution system.

## 3. CLOSED LOOP METHOD OF CONTROLING LORENZ'S ATRACTOR.

A chaotic set has imerged in itself a great number of small period periodic orbits. Furthermore, considering the ergodicity, a trajectory visits the neighborhood of each one of these periodic orbits (OTT; GREBOGI; YORKE, 1990). Some of those orbits may produce a specific performance which is desired to the system according to a particular criterion. Chaos, as sensitive dependence to small alterations on current state of the system and causing unpredictability of the same state at large time periods (OGATA, 2003), implies that the system behavior can be manipulated by tiny perturbations on specific available parameters.

The greater advantage of chaos controlling idea is its applicability to experimental systems, where earlier knowledge is not available. Using delayed coordinates may be very useful. Generally a single variable time series' data is sufficient to represent the system state (FRANKLIN; POWELL; WORKMAN, 1997).

### 3.1. Lorenz's Atractor

Lorenz's atractor is an autonomic system that has a strange atractor on it. It is composed by 3 (three) ordinary differential equations (ODEs) of first order:

$$\dot{X} = -\sigma(X - Y), \quad \text{Eq.(1)}$$

$$\dot{Y} = rX - Y - XZ, \quad \text{Eq.(2)}$$

$$\dot{Z} = XY - bZ, (X, Y, Z)\epsilon\mathbb{R}^3, (\sigma, r, b) > 0. \quad \text{Eq.(3)}$$

The divergence found by Lorenz was considered as an evidence of evolutional unpredictability of turbulent fluid flows, caused by the lack of precision on initial conditions determination (FIEDLER-FERRARA; PRADO, 1994).

To understand Lorenz's system is necessary refer to Rayleigh-Bernárd instability studies. It researches a case of a fluid between two horizontal plates, where the superior one has a lower temperature compared with the inferior one. This temperature difference, $\Delta\Theta$, determines the heat transference behavior between the plates. For small differences the heat transfer is by conduction process and for greater differences the heat transfer happens by convection process which produces convection rolls. This is known as Rayleigh-Bernárd instability.

Coefficients $X(t), Y(t)\ and\ Z(t)$ have well defined physical means: $X(t)$ is proportional to the convection intensity; $Y(t)$ is proportional to temperature difference between ascendant and descendent fluid flow; and $Z(t)$ is proportional to form distortion of vertical temperature, related to a linear form. Figure 4 shows these variables behavior.

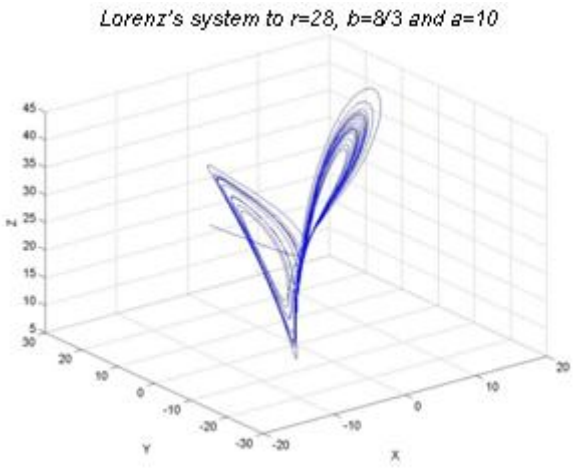


Figure 4: Lorenz's System for $r = 28, b = {}^{8}/_{3}, a = 10$.

### 3.2. Construction and processing of ANN

The neural network used in this work is as simple as possible (HAGAN; DEMUTH, 1999). It is formed for four layers, fig.6. The first one is the input layer, formed for four nodes, where are stored four values of X. The second one is called Hidden; composed for twenty (20) nodes (this number is proportional to training capability of the ANN). Each one of these twenty nodes is connected with all nodes of input layer, considering the weight value $w$ of each node through this process. The third layer is called Net. It is formed for a single node that is generated by Hidden layer nodes' interaction (working together as a team). Fourth and last layer is the network supervisor. It knows the desired behavior of the orbit we are pretending to reach, and its frequency. The supervisor receives values from Net layer and examines it with the difference of $X_*$ values (X delayed of a period of time) and the current value of X. With these information Supervisor is capable of modify Net's output. Figure 5 shows a diagram of the ANN used in this work.

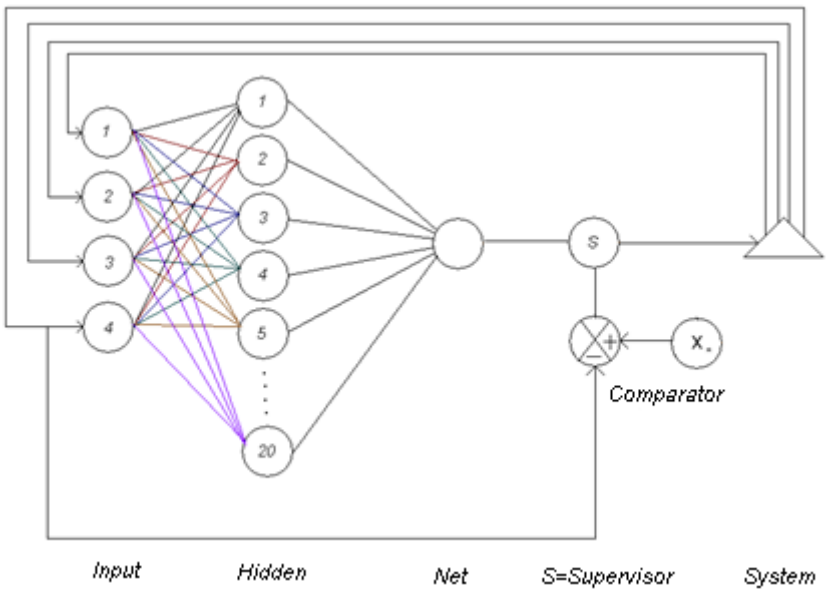


Figure 5: ANN diagram

### 3.3. Training and Simulation of Control System

At first the program generate, randomly, an initial population with one hundred (100) individuals. This population is put in "selection basin". Each individual has six (6) information slot ( $X_{ij}$ ), where the first four correspond to weights $w_{ij}$ with which the system entries will be weigh, establishing connection between input values' and Hidden layer.

The fifth slot is kept to save the weight $w_{ij}$ used to weigh Hidden layer nodes' in order to be examined before being disregarded or has its values processed and transferred to Net layer. To first population these weights are randomly generated between two values (previously chosen). In this case the interval [-1,1].

The sixth slot, initially empty, is reserved for save the fitness value of the node. This value measures how appropriate the node is to cause the system stability. By the end of each generation (program execution cycle) a fitness value is attributed to Hidden layer's individuals (as a team). At this point these individuals return to the "selection basin" and a new Hidden layer is selected.

Each node keeps the fitness obtained at best ANN whose it has participated (best team obtain greater fitness value). Once a initial population has been formed, a random process select a previously defined number of individuals of this population (twenty, in this case), in order to compose the ANN's Hidden layer. In the next step four input values (X values) are generated around one of non-trivial fixed points of Lorenz. These input values will be transferred to Hidden layer for weighting it using $W_{ij}$ and sum the results. This process produces a matriz called Y wich is applied to a sigmoidal function modified to provide outputs inside [-1,1] interval, what produces a column matrix between these limit values. Each element of this matrix is weigh by its correspondent weight $w_{i,5}$ of Hidden layer nodes. The results are summed and the result is also applied to a sigmoidal function (very similar to the one used in the last step) and the result compose the third layer of ANN (Net layer).

Here, the value held by Net layer is a scalar value which is used in a function responsible for the system's control, applying disturbances proportional to this Net value and measuring the new X signal situation. This new situation is presented to ANN's fourth layer, the Supervisor. The parameters used by the supervisor to calculate the disturbance that shall be applied in Lorenz's system are: Is necessary that current X value be inside a neighborhood (with radius determined by the program's user) of one period of time delayed X value to activate the system's control. If this condition is not satisfied, program keeps running without disturb the system until its stocasticity warrants the minimum proximity necessary to the desired orbit for starting the system control. Since this first condition is satisfied, the smaller Net's absolute value be, the smaller will be the control disturbance applied to the system. When stability is reached for a period of time greater than five times the orbit period previously chosen the system will be considered stable.

While these conditions of stability are not satisfied program keeps running cyclically, applying the calculated output values as an ANN feedback (four X values), closing the loop of control. The software trains the ANN continually so possible system's alterations can be solved without loosing of Lorenz's system control.

During this ANN's learning process, the system can leave the desired orbit neighborhood. If it happens the control disturbance is turned off and maintained this way until the system goes over the programmed neighborhood of control activating.

Each Hidden layer selected is submitted to at maximum a hundred thousand iterations (trying to stabilize the system). If this number of iterations do not be sufficient to get system stability, a fitness value inversely proportional to difference between current and delayed (of orbit period of time) X values is attributed to Hidden layer's nodes and this nodes return to "selection basin" becoming possible a new process to begin.

Ninety different Hidden layers are chosen. If though, the stability do not be reached, the genetic operators are applied to "selection basin" population (pop).

First the population is coded, in binary base, inside the [-1,1] interval with twelve slots, being able to assume 212 different values. The binary population is genetically recombined by crossover function. This function organizes the population in decreasing order of fitness. Half of the population (better adapted) will be submitted to genetic recombination's process where parents are randomly chosen, separated into pairs and intersection points are selected in order to produce new individuals (children) different of their parents.

The other half of population (worse adapted) is substituted for better adapted half of population individuals which went submitted to mutation. This process has as input the binary population and a mutation rate (between 0 and 1) arbitrary chosen by program user.

Mutation consists, in this case, in changing the value of one randomly chosen bit position, so if this bit is set as 1 it becomes 0 and if it is set as 0 it becomes to 1.

When crossover and mutation genetic processes are concluded a new population is formed and this binary data is decoded to decimal base again.

This whole process is executed up to ten times. And at each time the population goes through up to thirty different Hidden layers that are submitted up to ten thousand program execution's steps. If the stability is reached the process is interrupted at any time.

### 3.4. Results

For creating and testing the software has been used a computer with: Intel Core 2 Duo processor (1.83GHz, 667 MHz FSB, 2MB cache) and 3 GB of RAM memory. At developed program, the ANN adapts itself as the current X variable situation asks for it (by comparison with the same variable delayed of an orbit period of time), so a appropriated step integration period of time choice is very important to get great results at system control process (HANSELMAN; LITTLEFIELD, 1997).

Figure 6 shows X, Y and Z signals of controlled Lorenz's system upon an orbit with period equal to 650 ms. In this case the stability has been reached after 5 generations, 5 iterations and 530 steps.

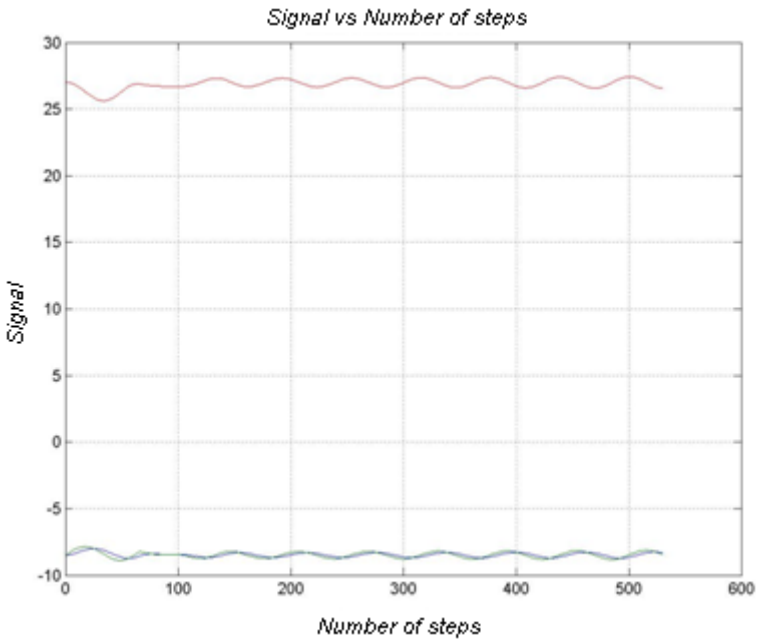


Figure 6. Lorenz's system – stabilized signal

Figure 7 has been added in order to show the perturbation's behavior during the program's iteration when system's stability has been achieved.

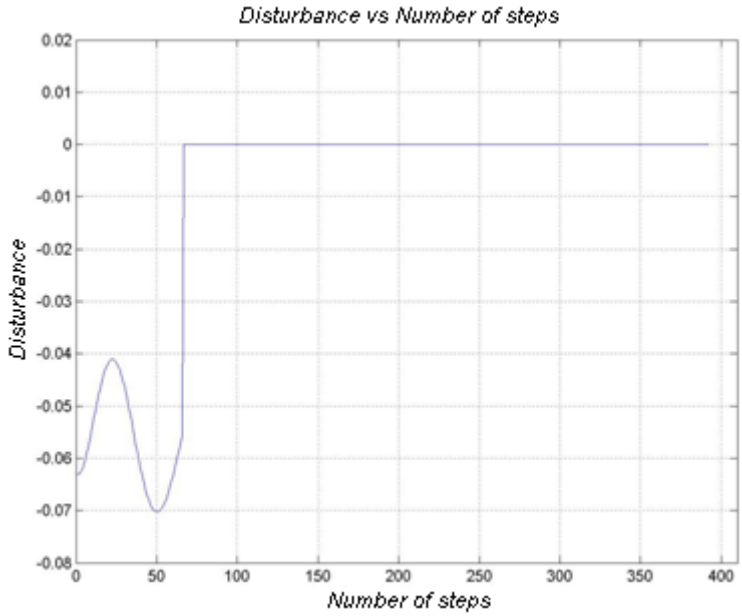


Figure 7. Lorenz's system – Signal perturbation vs Number of steps

We can see that when the stability is achieved the signal perturbation stops. It becomes evident by the horizontal line present in the graphic.

In order to check the loyalty of the results to the stipulated frequency Fig.8 has been plotted showing the signal X Fourier's transformation versus the selected period of time.

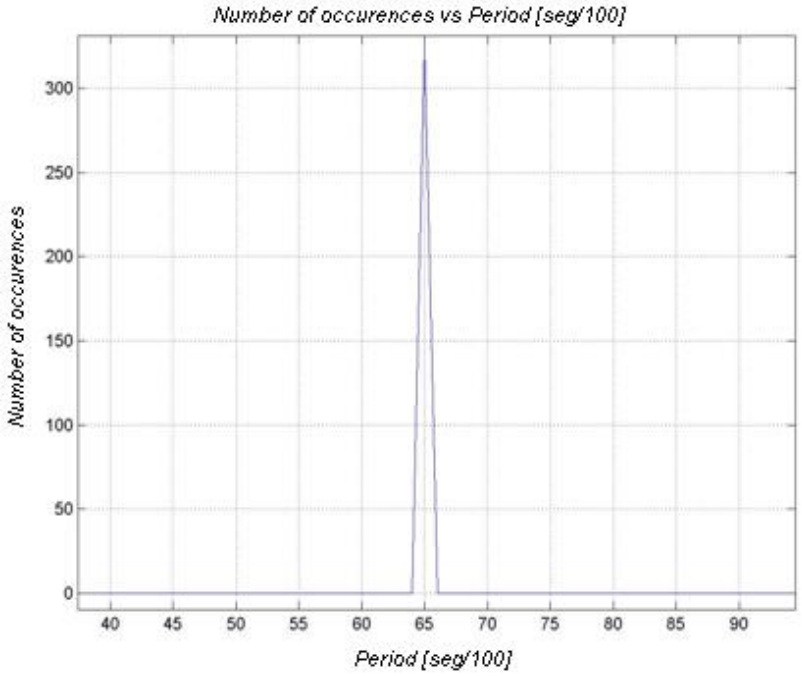


Figure 8. Number of occurrences vs Period of time [sec/100]

More than 80% of the program steps, at this specific iteration (when stability has been achieved), present the stipulated period (65 steps or 650 ms) where each step has 0.01 second of duration.

Since the control network has already been obtained, is possible to test it to stabilize the system with intervals where the control is turned-off. With this purpose has been plotted Fig.9, showing X signal behavior versus number of steps.

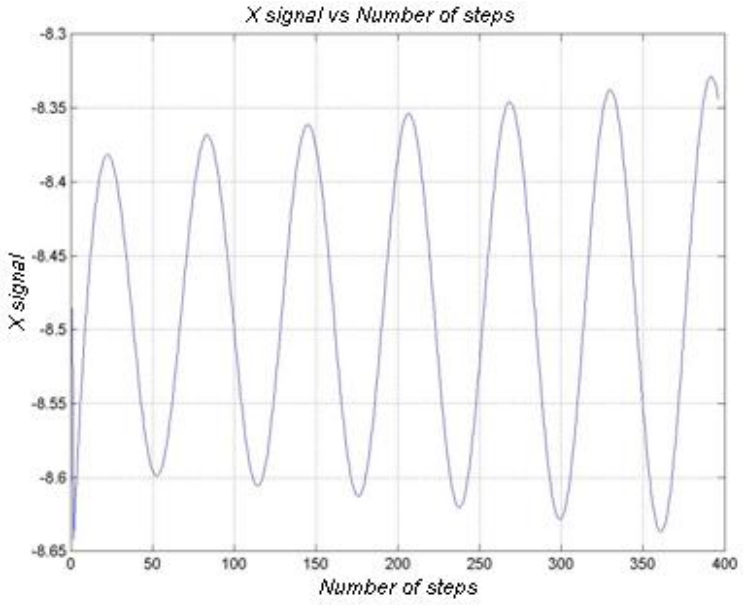


Figure 9. Stabilizing the system with control signal turned-off.

We can see that even with the control signal turned-off for a thousand steps, the stability has been achieved with the same number of steps (391 steps). It happened because, in this case, even without the system perturbation the control network warrants the system stability upon the selected periodic orbit.

Noise is a very common factor at natural systems. At practice it would be hard to find experiments where is not necessary consider the noise in order to reach more reliable results. Because of it noise interference has been applied to the system, so would be possible to verify how the control network reacts when this variable is added to the system.

With noises intensities of 15% of the controlled signal's mean amplitude (calculated as half of the absolute value of the difference between the last peak and valley of controlled signal) fig.10 has been plotted, showing signal X behavior versus number of steps (with control perturbation turned-off for first one thousand steps).

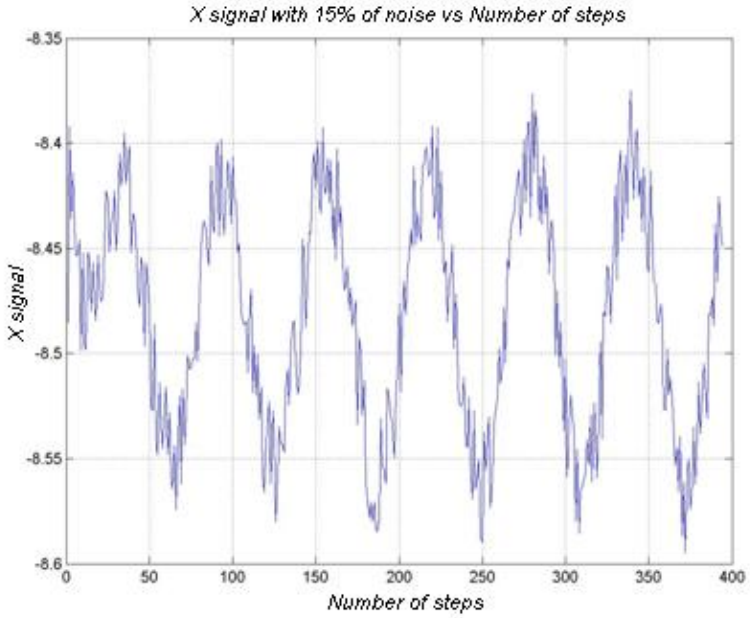


Figure 10. Trained network – X signal with noise intensity of 15%, with control perturbance turned-off.

Noise interference becomes evident in this case, but according with current stability criteria it still was possible to achieve it. It shows the developed program and control network's robustness in stabilizing the Lorenz's system upon periodic orbits.

Critical sesitivity to initial conditions is an intrinsic characteristic of chaotic systems. Seeking to verify this characteristic some tests have been made to our system, using different initial conditions: Upon one of non-trivial fixed points of Lorenz's system $(-\sqrt{72}, -\sqrt{72}, 27)$, 1%, 5% and 20% from that fixed point. It has been done to verify the program's behavior.

To the first situation, the system answer stabilized upon the non-trivial Lorenz's fixed point (did not produce a periodic orbit); To 1% the system answer stabilized upon a periodic orbit with amplitude of about 0.006; to 5% the stabilized orbit amplitude is about 0.03 (more than an order of magnitude higher than the previous measure); and to 20% from the non-trivial Lorenz's fixed point the stabilized orbit's mean amplitude for X signal was about 0.16 (over than an order of magnitude higher than obtained to 5%), as showed at Fig. 11.

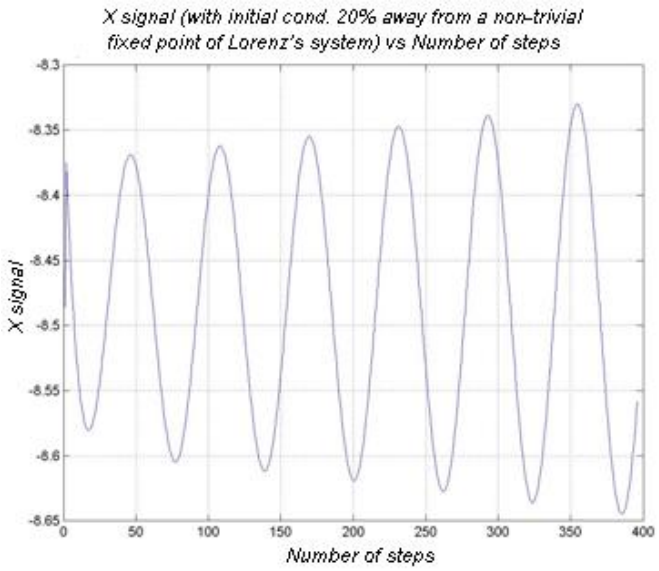


Figura 11. Sinal X com Cond. Inicial 20% Afastada de P. Fixo Não Trivial de Lorenz

Another important point is try to control the network with noise at its signal's measures. It makes harder the duty of finding a satisfactory control network. It has been analyzed cases with noises of 1%, 5% and 10% of mean amplitude of stabilized signal.

To 1% of noise you can see that still at first generation, iteration 11, after 562 steps, system stability has been achieved upon the desired orbit. It shows that even in more demanding environments' situations the program is able to find a satisfactory control network. To 5% noise's interference, program took a little longer to find the appropriate control network; it spent 3 generations, 10 iterations and 453 steps. An interesting observation is that with a noise of 10% of the average amplitude of the stabilized signal, the stability has been achieved still at first generation, after 14 iterations and 402 steps (faster than with 5% of noise).This is due to the random nature of the networks' selection during the program. A more appropriate network (to a specific situation) can be chosen quickly and move the system to stability in a more efficient way, as shown in Fig.12.

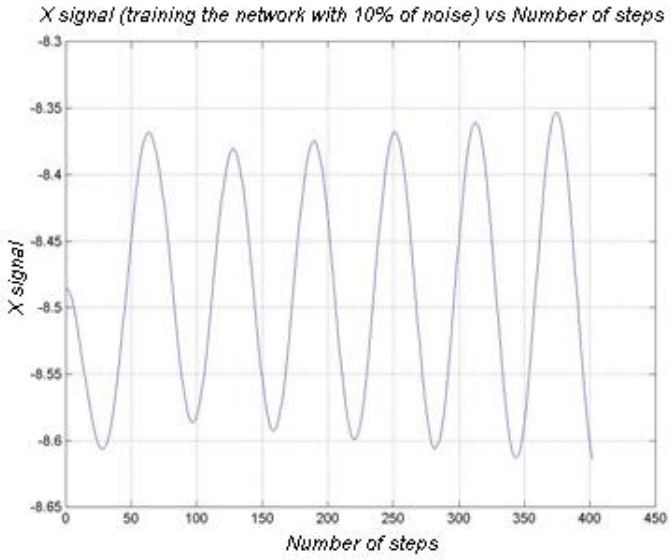


Figure 12. X signal (network trained with 10% of noise interference)

## 4. FINAL CONSIDERATIONS

Has been developed a software whose function is to control unstable systems, linear or not, using as control method artificial neural networks and genetic algorithms (working together).

These control methods showed a great learning and adaptation capability, forming a flexible and efficient control system which does not need any previous information about the plant to be controlled.

A trained network has been used during the software testing and no additional training has been necessary. This way we are able to show how good software answers are in controlling Lorenz's system, in many different situations, like:

- With control interference turned-off during time intervals, producing system unpredictability when control is turned-on.
- With noise's interference at measures after finding an appropriated control network to control system upon the desired periodic behavior. It requires more adaptability from control network in use.
- With noise's interference even before finding a control network capable of produce the specific periodic behavior. It creates uncertain propagation through iterations during program run. Signal's noise insertion simulates white noises and real system hardware's limitation, what can be decisive to mechanical implementation's success of simulated control systems.
- System initial conditions alteration. Although it forces system starts from a point different of non-trivial Lorenz's fixed points, trained ANN is still capable of promote system control upon a selected periodic behavior.

Software presented great answers when applied to unstable systems (chaotic), like Lorenz's attractor.

These results prove developed system's learning capability and show clearly how robust is the control's system project method utilized.

## 5. REFERÊNCIAS

BRAGA, Antônio; LUDERMIR, Teresa; CARVALHO, André. Redes neurais artificiais: teoria e aplicações. Rio de Janeiro: LTC, 2000.

CAVALCANTI, J.H.F; ALSINA, P.J.; FERNEDA, E. Posicionamento de um pêndulo invertido usando algoritmos genéticos. Revista da Sociedade Brasileira de Automática, vol 10/jan., fev., abril, p. 31-38, jan. 1999.

CUNHA, Wiliam Ferreira et al. Fitting potencial energy surface of reactive systems via genetic algorithm. International Journal of Quantum Chemistry, New York, v. 106, p. 2650-2657, 2006.

DIEDERICH, Joachim. Artificial neural networks: Concept learning. Los Alamitos: IEEE, 1990.

FIEDLER-FERRARA, Nelson; PRADO, Carmen P. Cintra do. Caos: uma introdução. São Paulo: Edgard Blücher, 1994.

FRANKLIN, Gene F.; POWELL, J. David; WORKMAN, Michael L. Digital control of dynamic systems. 3rd ed. Massachusetts: Addison Wesley, 1997.

GOLDBERG, David E. **Genetic algorithms in search, optimization and machine Learning**. [USA]: Addison-Wesley, 1989.

HAGAN, M. T.; DEMUTH, H. B. Neural networks for control. In: AMERICAN CONTROL CONFERENCE, San Diego, 1999. v. 3, p. 1642-1656.

HANSELMAN, Duane; LITTLEFIELD, Bruce. MATLAB: versão do estudante: guia do usuário. São Paulo: Makron Books, 1997.

HOLLAND, John H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. Cambridge, Massachusetts: MIT press, 1992.

KOZA, John. R.; KEANE, Martin A.; STREETER, Matthew J. Aperfeiçoando inventos. Scientific American Brasil, São Paulo, v. 2, n. 10, p. 61-67, mar. 2003.

LINDEN, Ricardo. Algoritmos genéticos: uma importante ferramenta da inteligência computacional. Rio de Janeiro: Brasport, 2006.

MITCHELL, Melanie. An introduction to genetic algorithms. Cambridge, Massachusetts: MIT Press, 1996.

OGATA, Katsuhiro. Engenharia de controle moderno. 4. ed. São Paulo: Prentice Hall, 2003.

OTT, E.; GREBOGI, C.; YORKE, J. A. Controlling chaos. **Physical Review Letters**, New York, v. 64, n. 11, p. 1196-1199, 1990.

PRESS, William H. Numerical recipes: the art of scientific computing. Cambridge: Cambridge University Press, 1986.

SUYKENS, J. A. K.; VANDEWALLE, J.; DE MOOR, B. L. .R. **Artificial neural networks for modeling and control of non-linear systems**. Boston: Kluwer Acad, 1996.

WEEKS, E. R.; BURGESS, J. M. Evolving artificial neural networks to control chaotic systems. **Physical Review E**. v. 56, n. 2, p. 1531-1540, Apr. 1997.

## 5. RESPONSABILITY NOTE

The authors are the only responsible for the printed material included to this article.