

A *Matlab* Problem-Solving Environment for Nonlinear Systems Education in Mathematics, Physics and Engineering

Akemi Gálvez Tomida

Department of Applied Mathematics and Comp. Sciences

University of Cantabria, Avda. de los Castros

s/n, E-39005, Santander, Spain

galveza@unican.es

Abstract

Currently, European countries are in the process of rethinking their Higher Education systems due to harmonization efforts initiated by Bologna's declaration. This scenario of reforms demands a completely new approach to the instructional process. A major issue in this context is the development of better, updated educational tools and materials specially adapted to the topics under study. This work reflects author's experience in developing a problem-solving environment designed for a first course on nonlinear systems for undergraduate students of Mathematics, Physics and Engineering. In this paper the architecture of this computer system along with a description of its main functionalities are briefly reported.

1 Introduction

Bologna's declaration - seen today as the well-known synonym for the whole process of reformation in the area of higher education - was signed in 1999 by 29 European countries with the objective to create "*a European space for higher education in order to enhance the employability and mobility of citizens and to increase the international competitiveness of European higher education*" [1]. Its upmost goal is the commitment freely taken by each signatory country to reform its own higher education system in order to create overall convergence at European level. This process encompasses the adoption of a common framework of readable and comparable degrees as well as the introduction of undergraduate and postgraduate levels in all countries along with ECTS (European Credit Transfer System) credit systems to ensure a smooth transition from one country's system to another one, thus enforcing free mobility of students, teachers and administrators among the European countries.

Unquestionably, Bologna's declaration opened the door to a completely new scenario for higher education in Europe. Nowadays, the European countries are in the midst of the process of restructuring their higher education system in order to fulfill the objectives of the declaration. At this time, the developments focus especially on academic aspects,

such as the definition of the new curricula and grading systems. However, the upcoming changes go far beyond these structural changes, as the personal development of students and teachers is also at the root of this new concept of education. For instance, students in this new model are no longer passive actors of the learning process. On the contrary, Bologna's declaration emphasizes the concept of self-learning so that students are getting more and more involved in their own learning.

An important issue in this process is to provide students with a good collection of scholar materials that enable them to accomplish the learning process by themselves. During the last few years, the author has been involved in the development of computer software for a first course on nonlinear systems for undergraduate students of Mathematics, Physics and Engineering. As a result, a new *Matlab* problem-solving environment designed to attain the demands of this new situation has been created from scratch. In this paper the architecture of this computer system along with a description of its main functionalities are briefly reported.

2 Nonlinear (Chaotic) Systems

The analysis of chaotic dynamical systems is one of the most challenging tasks in Computational Science. Because the chaotic systems are essentially nonlinear, their behavior is much more complicated than that of linear systems. In fact, even the simplest chaotic systems exhibit a bulk of different behaviors that can only be fully analyzed with the help of powerful hardware and software resources.

The range of different phenomena associated with the nonlinear systems is extremely varied. Chaos can be found in almost any field, ranging from chemical reactions to electronic circuits and lasers [2, 11], meteorology [16], ecology [18], etc. Nonlinearity appears in both discrete and continuous systems, which are described by iterated functions and differential equations, respectively [15, 19]. That means that the accurate analysis of such chaotic systems requires specialized mathematical tools and techniques, designed to account for the kind of system involved. This challenging issue has motivated an intensive development of programs and packages aimed at analyzing the range of different phenomena associated with the chaotic systems.

Among these programs and packages, those based on computer algebra systems (CAS) are receiving increasing attention during the last few years. Recent examples can be found, for instance, in [3, 4, 5, 6, 8] for *Matlab*, in [7, 9, 10, 12, 13, 14, 21] for *Mathematica* and in [22] for *Maple*, to mention just a few examples. In addition to their outstanding symbolic features, the CAS also include optimized numerical routines, nice graphical capabilities and - in a few cases such as in *Matlab* - the possibility to generate appealing GUIs (Graphical User Interfaces).

In this paper, we describe a problem-solving environment for the analysis of chaotic dynamical systems. The program, an improvement of the system reported in [5, 6] and implemented in the popular CAS *Matlab*, is suitable for both discrete and continuous chaotic systems. To this purpose, specialized symbolic and numerical libraries have been developed. Further, to provide end-users with a nice navigation and intuitive access to the main methods and routines, a powerful graphical user interface (also described in this paper) has been implemented. To show the good performance of this proposal, some illustrative examples are also reported.

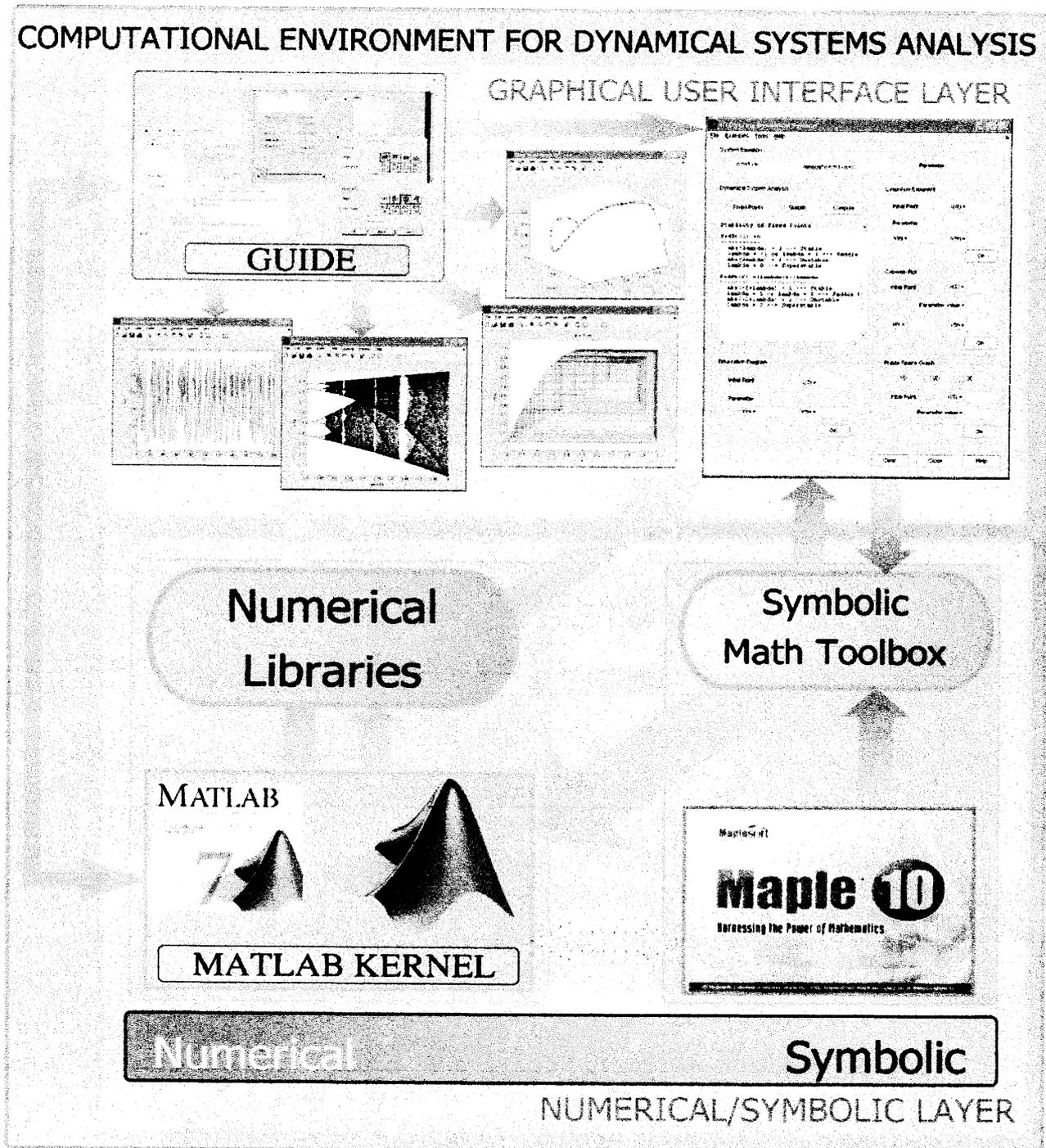


Figure 1: Architecture of the system.

3 Program Architecture and Implementation

3.1 Program Architecture

Figure 1 shows the architecture of the program described in this paper. It consists of two interconnected layers:

1. a *numerical-symbolic layer*: it is basically a collection of numerical and symbolic libraries containing the commands, functions and routines implemented to perform numerical and symbolic tasks.
2. a *graphical user interface (GUI) layer*: this component is responsible for input/output windowing, display of graphical output and smooth interaction with the user.

3.1.1 Numerical-symbolic layer

The numerical-symbolic layer is comprised of three different modules, according to the distinct processes (numerical, symbolic and graphical) to be carried out:

1. a *set of numerical libraries* containing the implementation of the commands, functions and routines for the numerical tasks. They have been implemented in the native *Matlab* programming language. To this aim we take advantage of the large collection of numerical routines available in the *Matlab* kernel such libraries are connected with. These standard *Matlab* routines provide extensive control on different options and are fully optimized to offer the highest level of performance.
2. a *set of symbolic routines and functions*. They have been implemented by using the *Symbolic Math Toolbox* that provides access to several *Maple* routines for symbolic tasks. This symbolic module is more important than it might seem at first sight; for instance, system equations are inputted symbolically so that some functional operators (such as derivatives) can be effectively applied. Further, some additional operators (string manipulation, forward/backward symbolic object-string conversion, symbol replacement and assignment, etc.) have also been used for symbolic purposes. It is worthwhile to mention that the *Symbolic Math Toolbox* is less powerful than the *Maple* kernel system it comes from. Fortunately, it is also possible to connect the kernels of *Matlab* and *Maple* for very specialized symbolic tasks.
3. some *graphical commands*. The powerful *Matlab* graphical capabilities exceed those commonly available in other CAS such as *Mathematica* and *Maple*. Although our current needs do not require applying them at full extent, they avoid the users the tedious and time-consuming task to implement many routines for graphical output by themselves. Some nice viewing features such as 3D rotation, zooming in and out, labeling, scaling, coloring and others are also automatically inherited from the *Matlab* graphical and windowing systems.

3.1.2 Graphical user interface layer

Although the libraries in previous layer are often enough to meet our computational needs, end-users might be challenged for using them properly unless they are really proficient on both *Matlab* syntax and functionalities and our implemented routines. This limitation can be overcome by creating a GUI; a well-designed GUI uses readily recognizable visual cues to help the user navigate efficiently through information. *Matlab* provides a powerful mechanism to generate GUIs by using the so-called **guide** (*GUI Development Environment*). This feature is not commonly available in many other CAS so far. Although its implementation requires - for complex interfaces - a high level of expertise,

it allows end-users to deal with our libraries with a minimal knowledge and input, thus facilitating its efficient use and dissemination.

Based on this discussion, a GUI layer has been implemented. Some examples of typical windows of our GUI are depicted in upper part of Figure 1. Some windows are for user interaction - typically input acquisition, parameter tuning and option selection tasks. Others are windows to display graphical output. All them allow an effective use of powerful interface tools designed according to the type of entities being displayed (e.g., drop-down menu for a choice list, check buttons for Boolean options, text boxes for displaying messages, dialog boxes for input/output user interaction, etc.). Additional functionalities are provided in hidden menus or in separate windows which can be invoked at will if needed so as to keep the main window streamlined and uncluttered. For instance, all graphical output is displayed in separate windows so that the information is better organized and flows in a natural and intuitive way. As a result, the system presents a GUI that is both aesthetic and very functional to the user.

3.2 Implementation issues

Regarding the implementation, this program has been developed by the author in *Matlab* v2007b [17] running on Windows XP operating system by using a PC with Intel Core 2 Duo processor at 2.4 GHz. and 2 GB of RAM. However, the program supports many different platforms, such as PCs (with Windows 9x, 2000, NT, Me, XP and Vista) and UNIX workstations. A version for Apple Macintosh with Mac OS X system is also available provided that Mac X11 (the implementation of the X Window System that makes it possible to run X11-based applications in Mac OS X) is properly installed and configured. Figures in this paper correspond to the PC platform version.

The graphical tasks are performed by using the *Matlab* GUI for the higher-level functions (windowing, menus, or input) while the built-in graphics *Matlab* commands are applied for rendering purposes. The numerical kernel has been implemented in the native *Matlab* programming language, and the symbolic kernel has been created by using the commands of the *Symbolic Math Toolbox*.

4 Discrete Systems

The program described in previous section is well suited for dealing with both discrete and continuous dynamical systems. This section illustrates the use of this software for the case of discrete systems.

4.1 Fixed points and stability

Given an iterated map defined by a function $f(x)$, a *fixed point* x^* of $f(x)$ is a point that is mapped to itself by the function, i.e. $f(x^*) = x^*$. Let's consider for example the logistic map given by $x_{n+1} = \lambda x_n(1 - x_n)$, where $n \in \mathbb{N}$ and λ is a system parameter, usually taking values on the interval $[0,4]$. The logistic map became popular following a seminal paper by the biologist Robert May in 1976 [18], where he introduced the discrete version of a demographic model due to Verhulst. Roughly, $x_n \in [0, 1]$ represents the population at year n , evolving from x_0 , the initial population value. Depending on the λ value, the

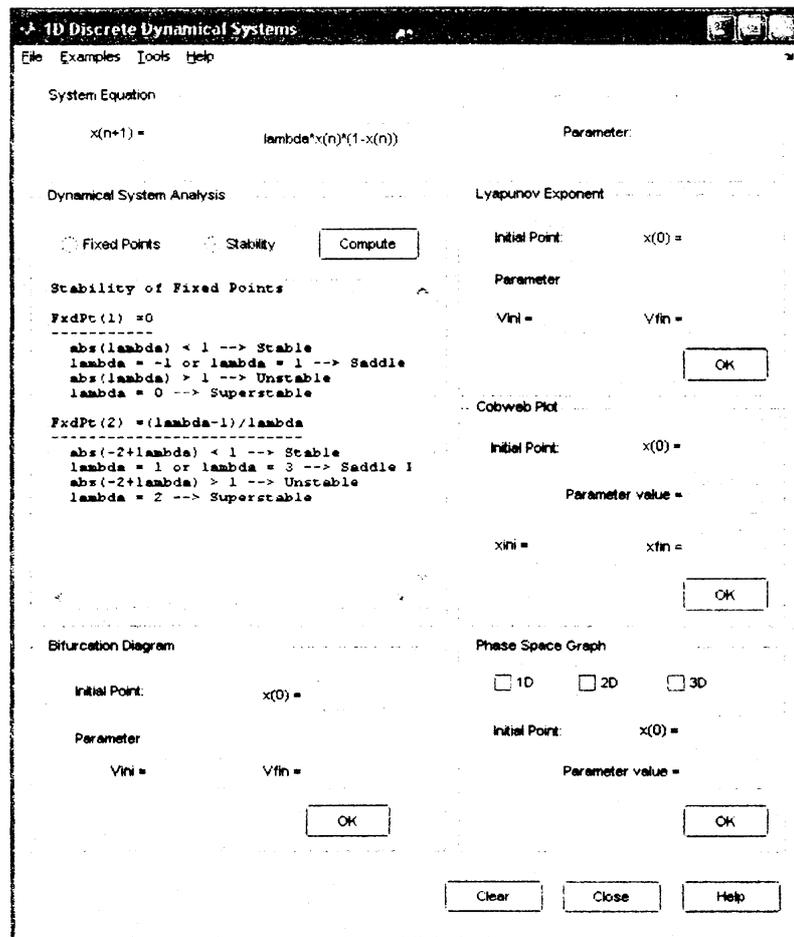


Figure 2: Fixed points and stability of the logistic map.

system evolves among a number of different situations, from the population eventually dying to stabilizing or changing chaotically (see [18] for more details).

The program described in this paper can compute the fixed points of any iterated map. To do so, we must enter the system equation as shown in Figure 2. Note that the equation is given in a mathematical-looking way so that it can be subsequently processed for further symbolic-numerical calculations. For instance, we can also analyze the stability of fixed points, by computing the eigenvalue of the fixed point, given by:

$\phi = \left[\frac{df(x)}{dx} \right]_{x=x^*}$. The fixed point is stable if $|\phi| < 1$, neutral if $|\phi| = 1$, unstable if $|\phi| > 1$, and superstable if $|\phi| = 0$. The fixed points for the logistic map are: $x = 0$ and $x = \frac{\lambda - 1}{\lambda}$. Figure 2 shows the fixed points of the logistic map along with their stability analysis in terms of the λ parameter.

4.2 Bifurcation diagrams

Depending on the λ value, the logistic map evolves among a number of different situations. This behavior is better analyzed by using the *bifurcation diagram*, which shows

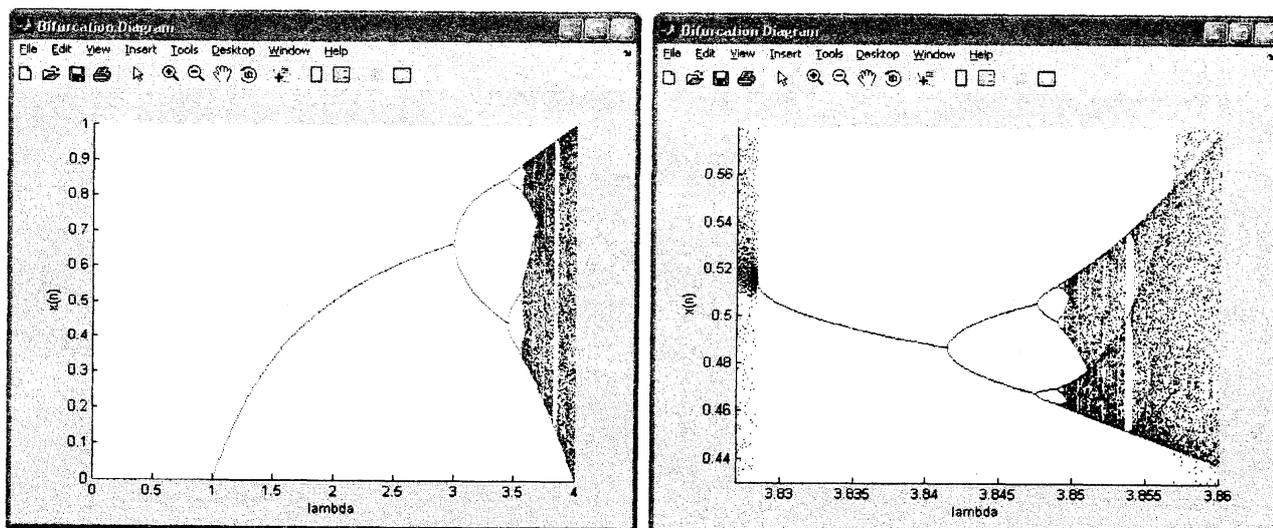


Figure 3: Bifurcation diagram of the logistic map on: (left) interval $[0, 4]$; (right) interval $[3.82, 3.86]$.

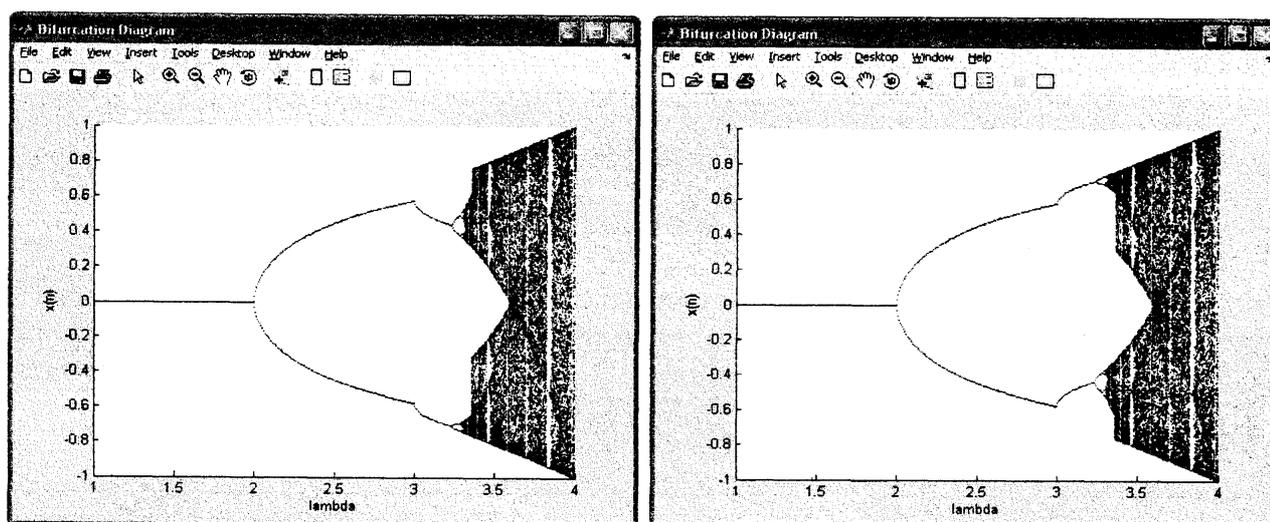


Figure 4: Bifurcation diagram of the cubic map on the interval $[1, 4]$ for the initial conditions: (left) $x_0 = 0.5$; (right) $x_0 = -0.5$.

the possible long-term values (fixed points or periodic orbits) of a system as a function of a system parameter. Figure 3 (left) shows the bifurcation diagram of the logistic map for $\lambda \in [0, 4]$. The initial input consists of the initial point x_0 and the initial and final values of the system parameter. The bifurcation diagram is a fractal: if you zoom in on the value $\lambda = 3.825$ and focus on one branch of the diagram, the situation nearby looks like a shrunk and slightly distorted version of the whole diagram, as shown in Figure 3 (right). This is an example of the deep and ubiquitous connection between chaos and fractals.

An interesting scenario appears when the branches in the bifurcation diagram do

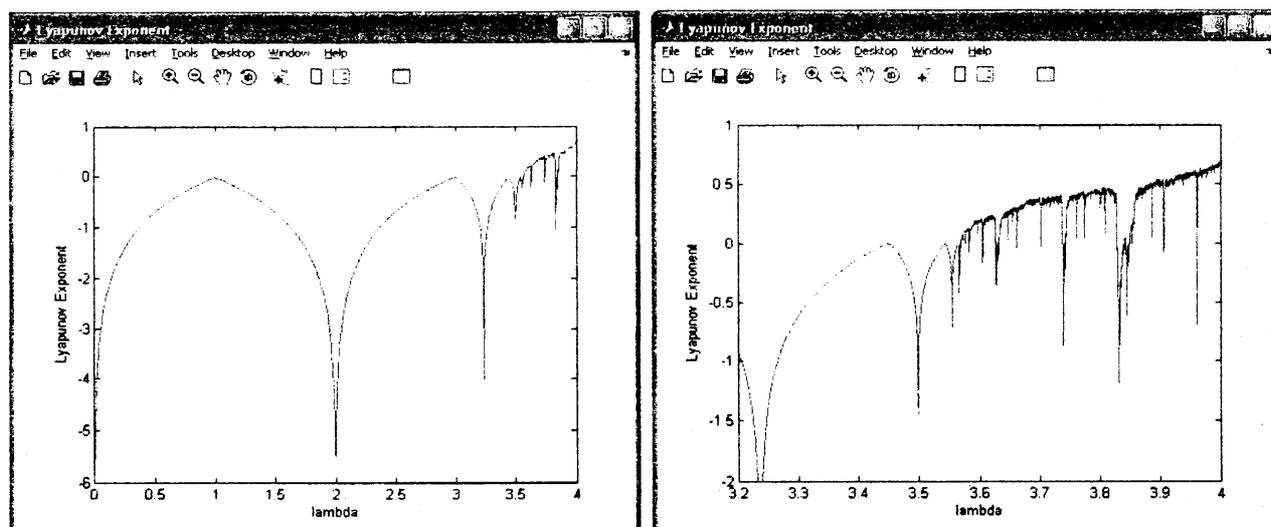


Figure 5: (left) Lyapunov exponent of the logistic map on the interval $[0, 4]$; (right) close up on the interval $[3.2, 4]$.

depend on the initial values of the system variable, x_0 . This happens, for instance, for the cubic map, given by: $x_{n+1} = (1 - \mu)x_n + \mu x_n^3$. This system has two fixed points of the form $x_{1,2} = \pm \sqrt{\frac{\mu - 1}{3\mu}}$, meaning that there is no single value to display the whole bifurcation diagram. Figure 4 displays the two bifurcation diagrams associated with this map, obtained from two different initial conditions for the system variable, namely $x_0 = 0.5$ (left) and $x_0 = -0.5$ (right). As the reader can see, there are two missing branches of the period-4 orbits in each diagram, so both pictures are complementary each other.

4.3 Lyapunov exponents

One indication of chaoticity is the so-called *sensitivity to initial conditions*, meaning that two initially closed arbitrary trajectories diverge exponentially over the time. The *Lyapunov exponent* (LE) of a dynamical system is the number that characterizes the rate of separation of these infinitesimally close trajectories along a given direction. Of course, the rate of separation can be different for different orientations of initial separation vector, leading to as many Lyapunov exponents as the number of dimensions of the phase space. LEs are intensively applied to analyze the behavior of nonlinear systems, since they indicate if small displacements of trajectories are along stable or unstable directions. In short, a negative LE is an indicator of regular (stable) behavior while a positive LE means that the orbit is unstable and chaotic.

Our program allows us to compute the Lyapunov exponents in a very easy way; the initial input is given by the initial point for the system variable and the interval for the system parameter. Figure 5 (left) depicts the Lyapunov exponent of the logistic map for the system parameter on the interval $[0, 4]$. Comparison of this picture with Figure 3 (left) shows that negative values for the LE are an indication of regular behavior, while positive values are associated with chaotic motion. Note also that the LE tends to $-\infty$

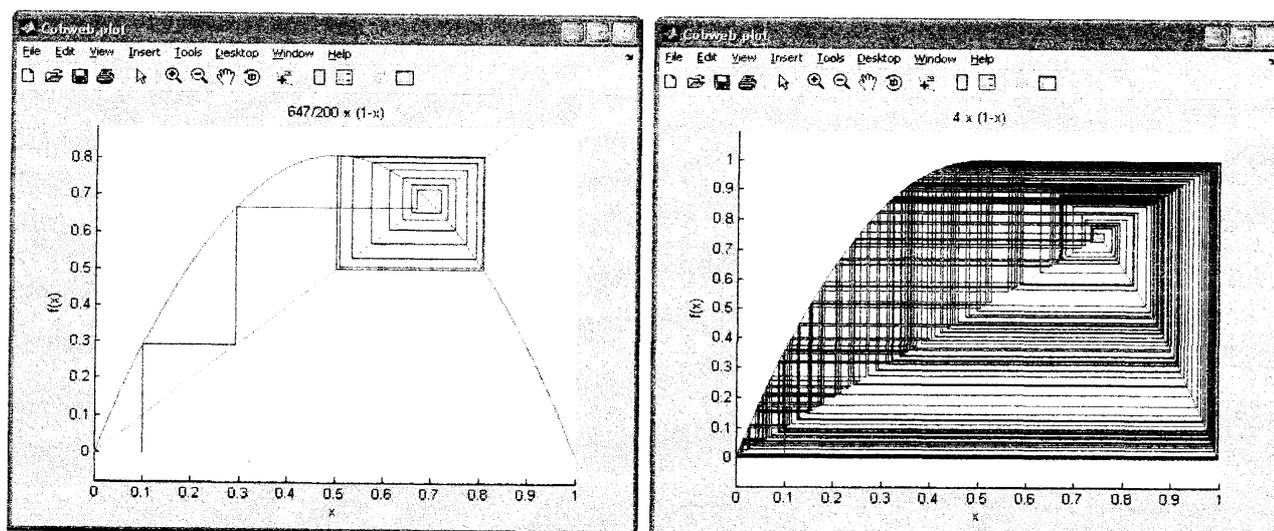


Figure 6: Cobweb plot of the logistic map: (left) $\lambda = 3.235$; (right) $\lambda = 4$.

for $\lambda = 2$, meaning the existence of a superstable fixed point. Figure 5 (right) shows a magnification of the figure on the left for the interval $[3.2, 4]$. We can see that the LE is positive for $\lambda > 3.569\dots$ except by the existence of some periodic windows, thus explaining very well the bifurcation diagram in Figure 3 (left).

4.4 Cobweb plot

A *cobweb plot* is a graphical procedure especially suited to analyze the qualitative behaviour of one-dimensional iterated functions. Cobweb plots are useful because they allow to determine the long-term evolution of an initial condition under repeated application of a map. Figure 6 shows two cobweb plots for the logistic map and the initial conditions $\lambda = 3.235$ (left), and $\lambda = 4$ (right). The first one shows the case of a period-2 orbit (represented by a rectangle) while the second case is a chaotic orbit.

4.5 Phase space graph

A very powerful strategy to analyze chaotic systems is to use the so-called *phase space graph*. By this we mean a collection of pictures associated with the orbit of a given point x_0 and embedded into different n -dimensional spaces. For $n = 1$ we get the signal of the orbit, i.e. the sequence of iterates $\{x_n\}_n$ over the time. Such sequence is usually called the *time series*. For $n = 2$ the graph is obtained by representing the sequence of iterates $\{x_{n+1}\}$ vs. $\{x_n\}$, for $n = 3$, we represent the sequence of iterates $\{x_{n+2}\}$ vs. $\{x_{n+1}\}$ and $\{x_n\}$ and so on.

Figure 7 uses the phase space graph to analyze the chaotic behavior of the logistic map for $\lambda = 4$. These pictures illustrate perfectly how the chaotic behavior looks like. On the left, the signal of the orbit is displayed. As discussed above, a characteristic of chaos is that chaotic systems exhibit a great sensitivity to initial conditions. A common source of such sensitivity to initial conditions is that the map represents a repeated folding and stretching of the space on which it is defined. The $n = 2$ phase space graph for the

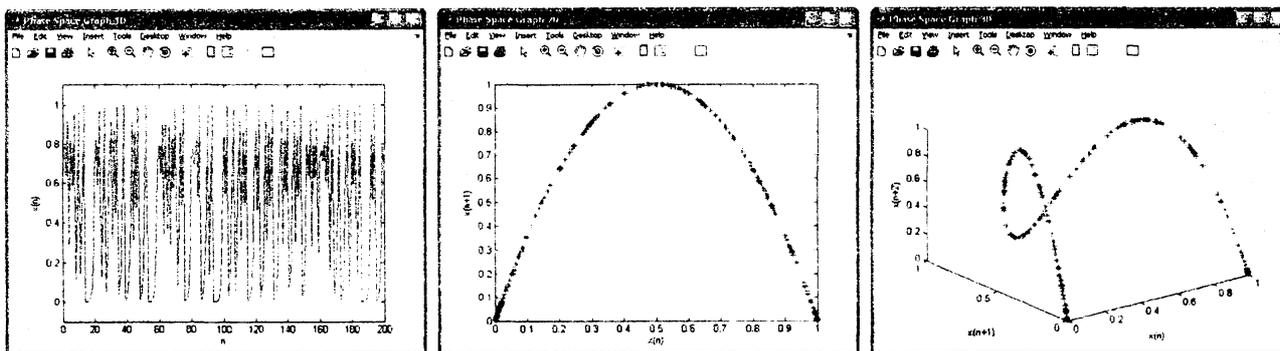


Figure 7: (left to right) 1D, 2D and 3D phase space graph for the logistic map.

logistic map, represented in Fig. 7 (middle), gives a two-dimensional phase diagram of the logistic map showing the quadratic curve of its iterated equation. We can also embed the same sequence in a 3D phase space, in order to investigate a deeper structure of the map. Figure 7 (right) shows how initially nearby points begin to diverge, particularly in those regions corresponding to the steeper sections of the plot.

5 Continuous Systems

In this section we show some applications of the program through illustrative examples for the case of continuous systems. In this work we restrict ourselves to the case of finite-dimensional flows, which are mathematically described by systems of ordinary differential equations.

5.1 Symbolic-numerical analysis

Figures 8-11 show screenshots of a typical session for analyzing 3D continuous systems. The session workflow is as follows: firstly, the user inputs the system equations expressed symbolically. For instance, in Figure 8 we consider the famous Lorenz system [16], given by: $(x', y', z') = (\sigma(y - x), (R - z)x - y, xy - bz)$ where σ , R and b are the system parameters. The program includes a module for the computation of the Jacobian matrix and the equilibrium points of any finite-dimensional flow. The *Jacobian matrix* is a square matrix whose entries are the partial derivatives of the system equations with respect to the system variables. If no value for the system parameters is provided, the computation is performed symbolically and the corresponding output depends on those system parameters. The equilibrium points and the eigenvalues and eigenvectors of the system can also be computed in a similar way.

Figure 8 shows the symbolic Jacobian matrix for the Lorenz system, which depends not only on the system parameters but also on the system variables. Once some parameter values are given ($\sigma = 10$, $R = 60$ and $b = 8/3$ in this example), the Lyapunov exponents (LE) of the system can be numerically computed. To this purpose, a numerical integration method is applied [20]. The corresponding options and parameter values are shown in Figure 9. The numerical values of these LE are 1.4, 0.0012 and -15 respectively. Their graphical representation over the time is depicted in Figure 10.

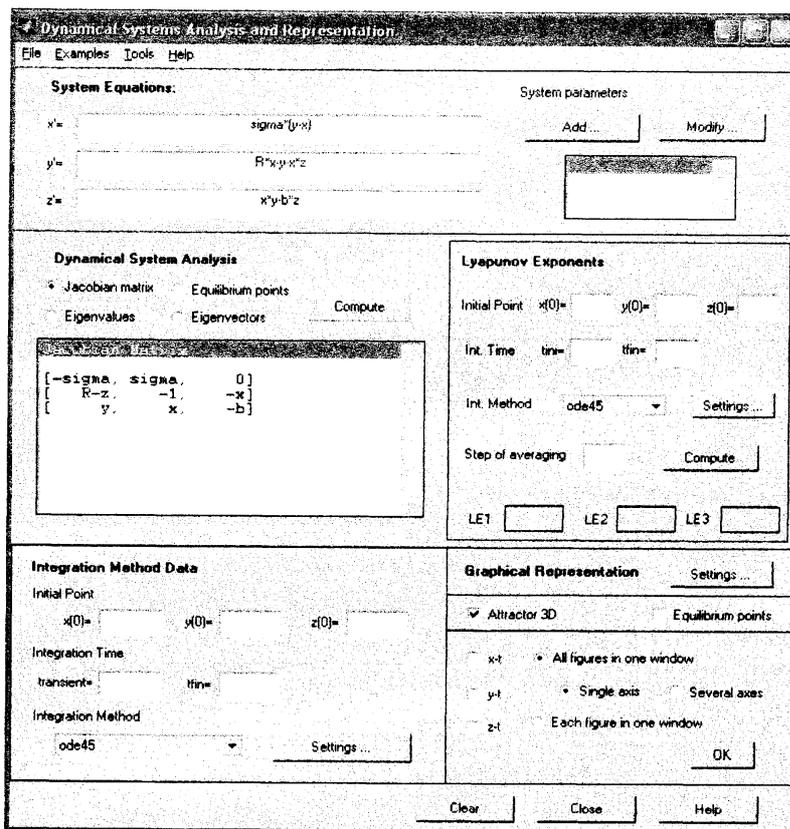


Figure 8: Symbolic Jacobian matrix for the Lorenz system.

Roughly speaking, LEs are a generalization of the eigenvalues for nonlinear flows. In particular, a negative LE indicates that the trajectory evolves along the stable direction for this variable (and hence, regular behavior for that variable is obtained) while a positive value indicates a chaotic behavior.

5.2 Visualization of chaotic attractors

Since in our example we find positive LE, the system exhibits a chaotic behavior. This fact is evidenced in Figure 11 (left) where the corresponding attractor and the equilibrium points of the Lorenz system for our choice of the system parameters are displayed. Their corresponding numerical values are shown in the main window of Figure 9. Finally, Figure 11 (right) shows the evolution of the system variables over the time from $t = 0$ to $t = 50$.

In order to display the attractor and/or the evolution of the system variables over the time (like in Figure 11), some kind of numerical integration is required. The program in this paper allows end-users to choose different numerical integration methods [20], including the classical Euler and 2nd- and 4th-order Runge-Kutta methods (implemented by the author) along with some more sophisticated methods from the *Matlab* kernel such as `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t` and `ode23tb` (see [17] for details). Some input required for the numerical integration (such as the initial point and the integration time) is also given at this stage. By pressing the “Numerical Integration settings” button, some additional options (such as the absolute and relative error tolerance, the initial and

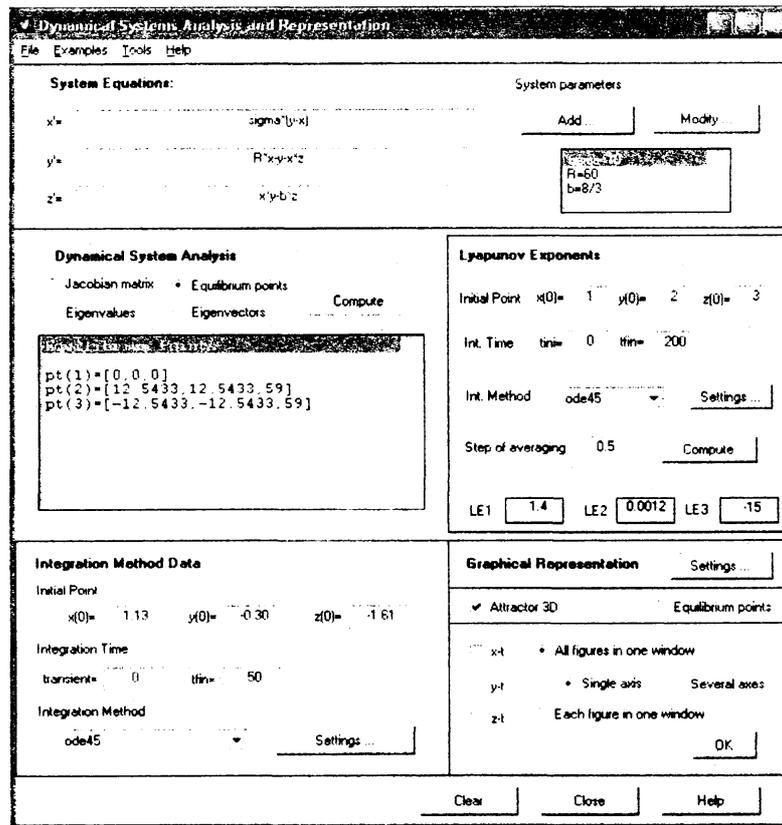


Figure 9: Equilibrium points and Lyapunov exponents for the Lorenz system.

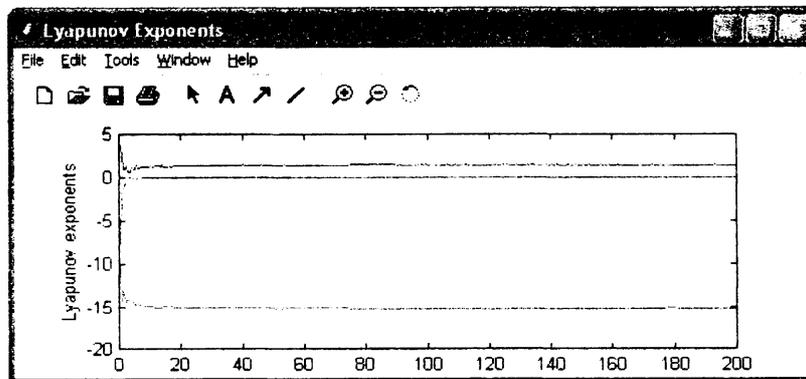


Figure 10: Temporal evolution of the Lyapunov exponents for the Lorenz system.

maximum stepsize and refinement, the computation speed and others) can be set up in a separate window. Then the user proceeds with the graphical representation stage, where he/she can display the attractor of the dynamical system and/or the evolution of any of the system variables over the time. Such variables can be depicted on the same or on different axes and windows. The “Graphical Representation settings” button opens a new window where different graphical options such as the line width and style, markers for the equilibrium points, and some coloring options can be defined. The result is the graphical

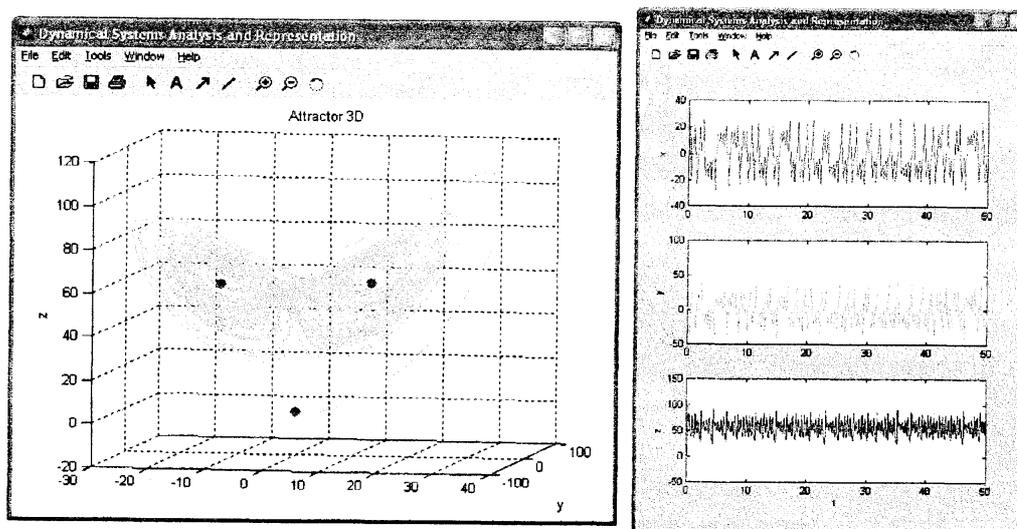


Figure 11: Lorenz system: (left) chaotic attractor and equilibrium points; (right) temporal evolution of system variables.

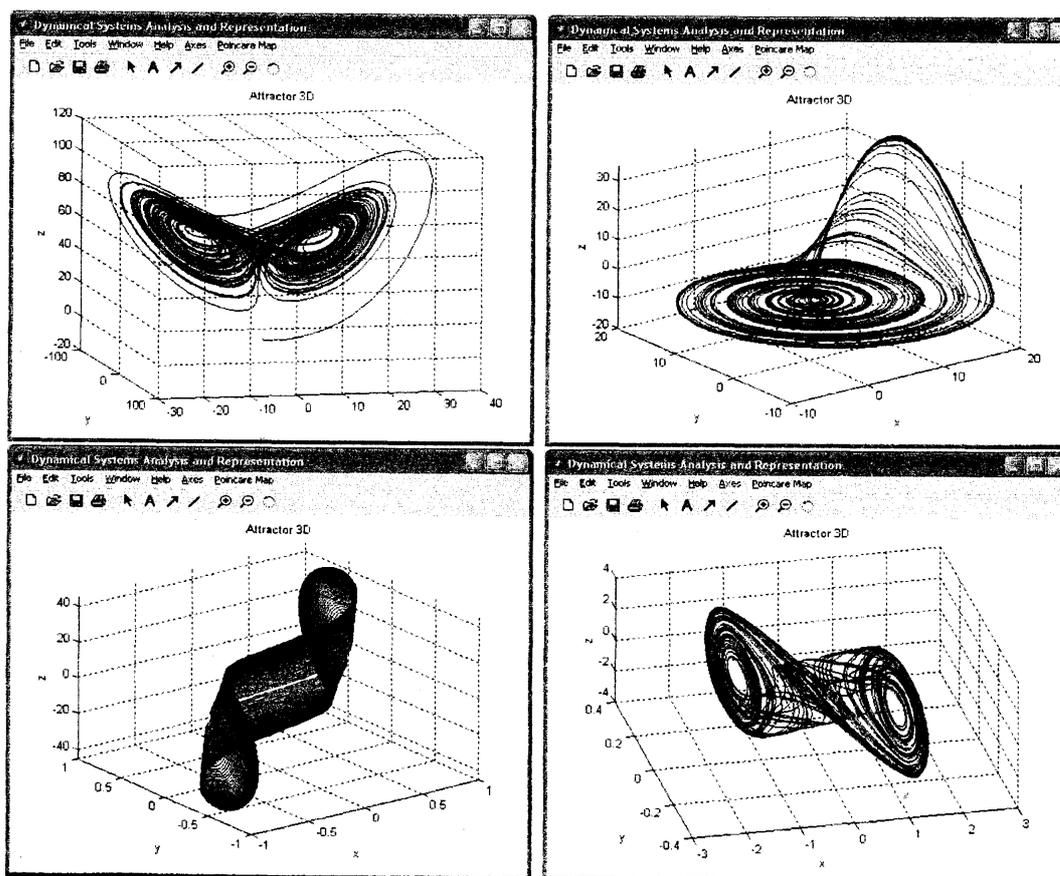


Figure 12: Chaotic attractors of 3D flows: (top-left) Lorenz system; (top-right) Rössler system; (bottom-left) Van der Pol-Duffing oscillator; (bottom-right) Chua's circuit.

output shown in Figure 11 where the chaotic attractor and the temporal evolution of the system variables are displayed.

Figure 12 shows the chaotic attractors of four distinct nonlinear flows: Lorenz system for weather forecasting, Rössler chemical reaction, Van-der-Pol Duffing oscillator and Chua's electronic circuit (see [19] for further information about these systems).

Acknowledgments

The author would like to express her sincere acknowledgment and appreciation to Prof. Setsuo Takato for his kind invitation to participate in this RIMS workshop and visit the lovely city of Kyoto, and for creating such a friendly atmosphere during all my stay in Japan. I really hope we can meet again very soon.

This research has been supported by the Computer Science National Program of the Spanish Ministry of Education and Science, Project Ref. #TIN2006-13615 and the University of Cantabria.

References

- [1] The Bologna Declaration on the European space for higher education: an explanation. Association of European Universities & EU Rectors' Conference (1999) pp. 4 (available at: <http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf>).
- [2] Chua, L.O., Komuro, M., Matsumoto, T.: The double-scroll family. *IEEE Transactions on Circuits and Systems*, **33**, (1986) 1073-1118.
- [3] Dhooze, A., Govaerts, W., Kuznetsov, Y.A.: Matcont: A Matlab package for numerical bifurcation analysis of ODEs. *ACM Transactions on Mathematical Software*, **29**(2) (2003) 141-164
- [4] Dhooze, A., Govaerts, W., Kuznetsov, Y.A.: Numerical continuation of fold bifurcations of limit cycles in MATCONT. *Lecture Notes in Computer Science*, **2657** (2003) 701-710
- [5] Gálvez, A.: Numerical-symbolic Matlab program for the analysis of three-dimensional chaotic systems. *Lecture Notes in Computer Science*, **4488** (2007) 211-218
- [6] Gálvez, A.: Matlab toolbox and GUI for analyzing one-dimensional chaotic maps. *International Conference on Computational Science and Applications, ICCSA2008*, IEEE Computer Society Press, Los Alamitos, CA, (2008) 211-218.
- [7] Gálvez, A., Iglesias, A.: Symbolic/numeric analysis of chaotic synchronization with a CAS. *Future Generation Computer Systems* **25**(5) (2007) 727-733
- [8] Govaerts, W., Sautois, B.: Phase response curves, delays and synchronization in Matlab. *Lecture Notes in Computer Science*, **3992** (2006) 391-398
- [9] Gutiérrez, J.M., Iglesias, A., Guémez, J., Matías, M.A.: Suppression of chaos through changes in the system variables through Poincaré and Lorenz return maps. *International Journal of Bifurcation and Chaos*, **6** (1996) 1351-1362

- [10] Gutiérrez, J.M., Iglesias, A.: A Mathematica package for the analysis and control of chaos in nonlinear systems. *Computers in Physics*, **12**(6) (1998) 608-619
- [11] Iglesias, A.: A new scheme based on semiconductor lasers with phase-conjugate feedback for cryptographic communications. *Lectures Notes in Computer Science*, **2510** (2002) 135-144
- [12] Iglesias, A., Gálvez, A.: Analyzing the synchronization of chaotic dynamical systems with Mathematica: Part I. *Lectures Notes in Computer Science*, **3482** (2005) 472-481
- [13] Iglesias, A., Gálvez, A.: Analyzing the synchronization of chaotic dynamical systems with Mathematica: Part II. *Lectures Notes in Computer Science*, **3482** (2005) 482-491
- [14] Iglesias, A., Gálvez, A.: Revisiting some control schemes for chaotic synchronization with Mathematica. *Lectures Notes in Computer Science*, **3516** (2005) 651-658
- [15] Lauwerier, H.A.: One-dimensional iterative maps. In: *Chaos*. Holden, A.V. (ed.). Manchester University Press, Manchester (1986)
- [16] Lorenz, E.N., Deterministic nonperiodic flow, *Journal of Atmospheric Sciences*, **20** (1963) 130-141.
- [17] The Mathworks Inc: *Using Matlab*. Natick, MA (1999)
- [18] May, R.M.: Simple mathematical models with very complicated dynamics. *Nature* **261** (1976) 459-467
- [19] Ott, E.: *Chaos in Dynamical Systems*. Cambridge University Press, Cambridge (1993)
- [20] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: *Numerical Recipes* (2nd edition), Cambridge University Press, Cambridge, 1992.
- [21] Sarafian, H.: A closed form solution of the run-time of a sliding bead along a freely hanging slinky. *Lecture Notes in Computer Science*, **3039** (2004) 319-326
- [22] Zhou, W., Jeffrey, D.J. Reid, G.J.: An algebraic method for analyzing open-loop dynamic systems. *Lecture Notes in Computer Science*, **3516** (2005) 586-593