Numerical-Symbolic *Matlab* Program for the Analysis of Three-Dimensional Chaotic Systems

Akemi Gálvez

Department of Applied Mathematics and Computational Sciences,
University of Cantabria, Avda. de los Castros,
s/n, E-39005, Santander, Spain
akemi.galvez@postgrado.unican.es

Abstract. In this paper, a new numerical-symbolic *Matlab* program for the analysis of three-dimensional chaotic systems is introduced. The program provides the users with a GUI (Graphical User Interface) that allows us to analyze any continuous three-dimensional system with a minimal input (the symbolic ordinary differential equations of the system along with some relevant parameters). Such an analysis can be performed either numerically (for instance, the computation of the Lyapunov exponents, the graphical representation of the attractor or the evolution of the system variables) or symbolically (for instance, the Jacobian matrix of the system or its equilibrium points). Some examples of the application of the program to analyze several chaotic systems are also given.

1 Introduction

The analysis of chaotic dynamical systems is one of the most challenging tasks in computational science. Because these 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. 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 [2,3,5] for *Matlab*, in [4,7,9,10,11,12,16] for *Mathematica* and in [17] 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, the abovementioned features have been successfully applied to generate a new numerical-symbolic *Matlab* program for the analysis of three-dimensional chaotic systems. The program provides the users with a GUI that allows us to analyze any continuous three-dimensional system with a minimal input (the symbolic ordinary differential equations of the system along with some

Y. Shi et al. (Eds.): ICCS 2007, Part II, LNCS 4488, pp. 211–218, 2007.

relevant parameters). Such an analysis can be performed either numerically (for instance, the computation of the Lyapunov exponents, the graphical representation of the attractor or the evolution of the system variables over the time) or symbolically (for instance, the Jacobian matrix of the system or its equilibrium points). This paper describes the main components of the system as well some of its most remarkable features. Some examples of the application of the program to analyze several chaotic systems are also given.

2 Program Architecture and Implementation

The program introduced in this paper is comprised of four different components:

1. *a set of numerical libraries* containing the implementation of the commands and functions designed for the numerical tasks. They have been generated by using the native *Matlab* programming language and taking advantage of the wealth of numerical routines available in this system. Usually, these *Matlab* routines provide full control on a number of different options (such as the absolute and relative error tolerance, stopping criteria and others) and are fully optimized to offer the highest level of performance. In fact, this is one of the major strengths of the program and one of the main reasons to choose *Matlab* as its optimal programming environment.
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.
3. *the graphical commands for representation tasks*. The powerful graphical capabilities of *Matlab* exceed those commonly available in other CAS such as *Mathematica* and *Maple*. Although our current needs do not require to apply 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, coloring and others are also automatically inherited from the *Matlab* windows system.
4. *a GUI*. Matlab provides a 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 the end users to apply the program with a minimal knowledge and input, thus facilitating its use and dissemination.

Regarding the implementation, this program has been developed by the author in *Matlab* v6.0 by using a Pentium IV processor at 2.4 GHz. with 512 MB of RAM. However, the program supports many different platforms, such as PCs (with Windows 9x, 2000, NT, Me and XP) and UNIX workstations. 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*.

3 Some Illustrative Examples

In this section we show some applications of the program through some illustrative examples.

3.1 Visualization of Chaotic Attractors

This example is aimed at showing the numerical and graphical features of the program. Figure 1 shows a screenshot of a typical session for visualization of chaotic attractors. The different windows involved in this task have been numbered for the sake of clarity: #1 indicates the main window of the program, from where the other windows will show up when invoked. The workflow is as follows: firstly, the user inputs the system equations (upper part of window #1), which are expressed symbolically. At this stage, only continuous three-dimensional flows - described by a system of ordinary differential equations (EDOs) - are considered. For instance, in Fig. 1 we consider Chua's circuit, given by:

$$\begin{cases} x' = \alpha[y - x - f(x)] \\ y' = x - y + z \\ z' = -\beta y \end{cases}, \tag{1}$$

where

$$f(x) = bx + \frac{1}{2}(a - b)\left[|x + 1| - |x - 1|\right] \tag{2}$$

is the 3-segment piecewise-linear characteristic of the nonlinear resistor (Chua's diode) and α, β, a and b are the circuit parameters. Then, the user declares the system parameters and their values. In our example, we consider $\alpha = 8.9$, $\beta = 14.28$, $a = -1.14$ and $b = -0.71$, for which the system exhibits chaotic behavior [1]. In order to display the attractor and/or the evolution of the system variables over the time, some kind of numerical integration is required. The lower part of window #1 allows the user to choose different numerical integration methods [15], 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 [14] 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, window #2 appears and some additional options (such as the absolute and relative error tolerance, the initial and maximum stepsize and refinement, the computation speed and others) can be set up. Once chosen, the user proceeds to 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

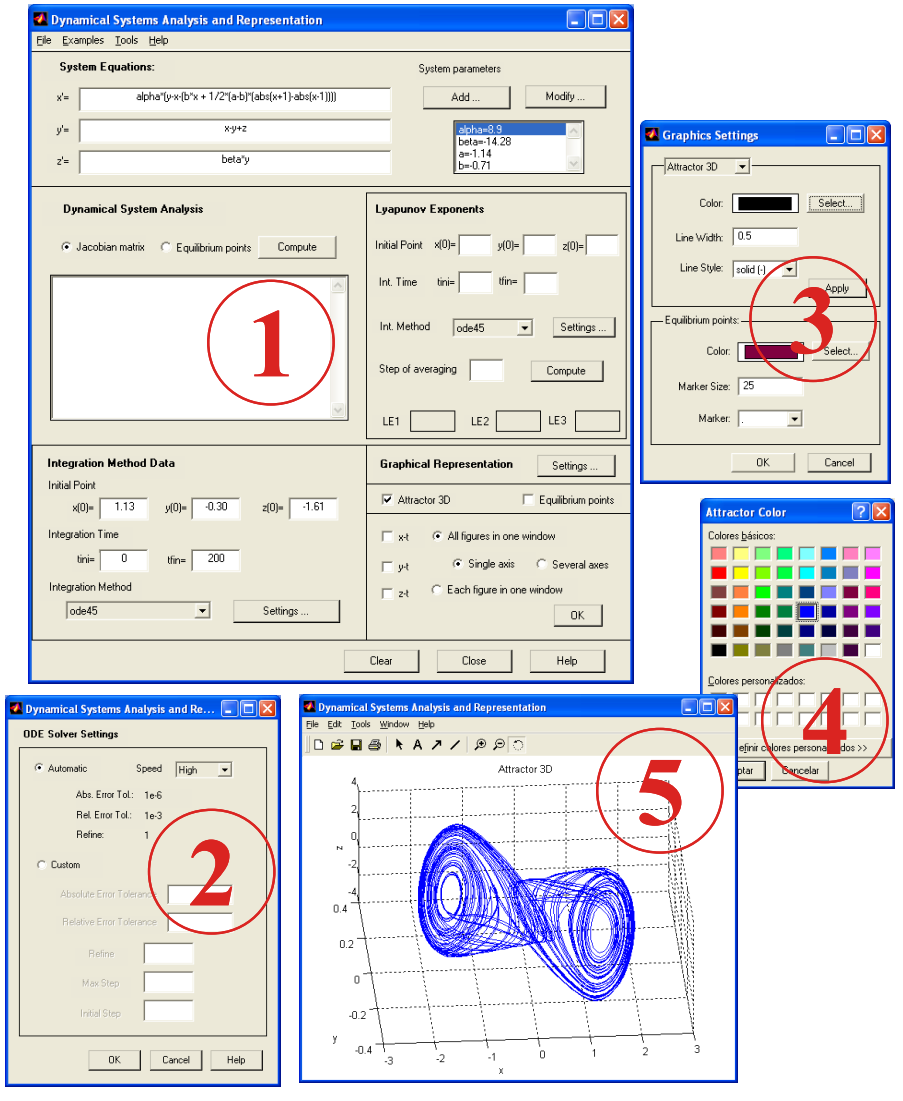

Fig. 1. Screenshots of the Matlab program for the analysis of chaotic systems: general setup for the visualization of chaotic attractors

be depicted on the same or on different axes and windows. The "Graphical Representation settings" button opens the window #3, where different graphical options such as the line width and style, markers for the equilibrium points, and others (including some coloring options leading to window #4) can be defined. The final result is the graphical output shown in window #5 where the double scroll attractor is displayed.

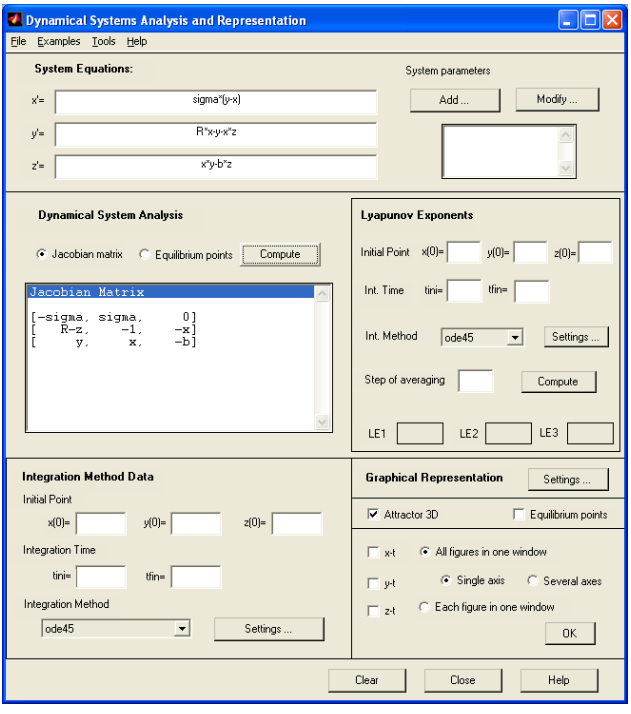

Fig. 2. Symbolic computation of the Jacobian matrix for the Lorenz system

3.2 Symbolic-Numerical Analysis of Chaotic Systems

An appealing feature of this program is the possibility to analyze the chaotic systems either symbolically or numerically. Figure 2 shows an example for the well-known Lorenz system [13], given by:

$$\begin{cases} x' = \sigma(y - x) \\ y' = Rx - y - xz \\ z' = xy - bz \end{cases}, \tag{3}$$

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 three-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. Figure 2 shows the Jacobian matrix for the Lorenz system, which depends not only on the system parameters but also on the system variables. Otherwise, the computations are performed numerically. For instance, once some

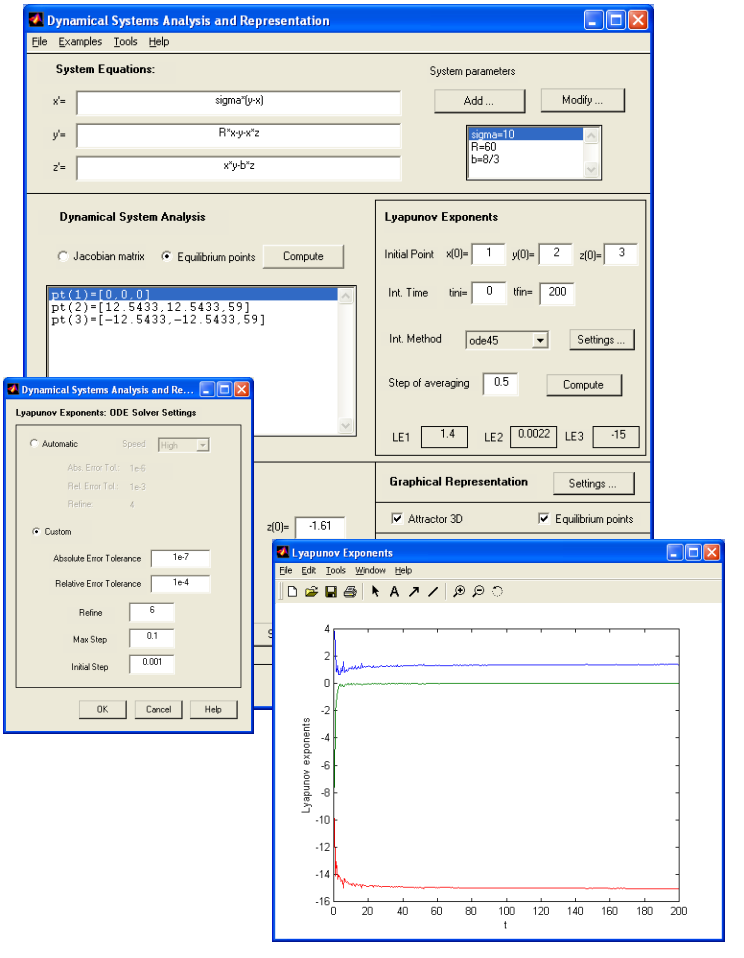

Fig. 3. Numerical computation of the equilibrium points and the Lyapunov exponents of the Lorenz system

parameter values are given ($\sigma = 10$, $R = 60$ and $b = \dfrac{8}{3}$ in this example), the Lyapunov exponents (LE) of the system can be numerically computed. To this purpose, a numerical integration method is applied. Figure 3 shows the window at which the different options for this numerical integration process can be chosen (left window) along with the graphical representation of the three Lyapunov exponents over the time (right window). As shown in the figure, the numerical values of these LE are 1.4 and 0.0022 and -15 respectively. Roughly speaking, the LE are a generalization of the eigenvalues for nonlinear flows. They are intensively applied to analyze the behavior of nonlinear systems, since they indicate if small displacements of trajectories are along stable or unstable

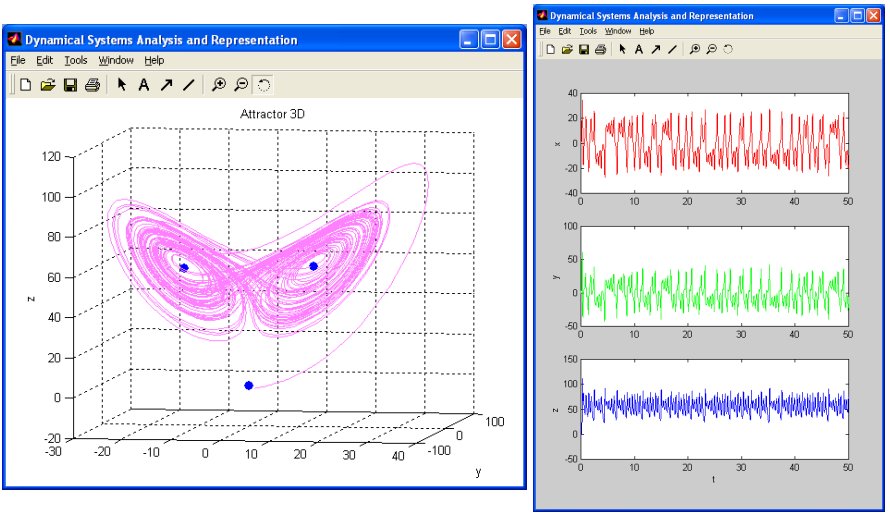

Fig. 4. (left) Attractor and equilibrium points of the Lorenz system analyzed in Figure 3; (right) evolution of the system variables over the time

directions. 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. Because in our example we find positive LE, the system exhibits a chaotic behavior. This fact is evidenced in Figure 4 (left) where the corresponding attractor of the Lorenz system for our choice of the system parameters is displayed. The figure also displays the equilibrium points of the Lorenz system for our choice of the system parameters. Their corresponding numerical values are shown in the main window of Figure 3. Finally, Figure 4 (right) shows the evolution of the system variables over the time from $t = 0$ to $t = 200$.

4 Conclusions and Further Remarks

In this paper, a new numerical-symbolic *Matlab* program for the analysis of three-dimensional continuous chaotic systems has been introduced. The system allows the user to compute the Jacobian matrix, the equilibrium points and the Lyapunov exponents of any chaotic three-dimensional flow, as well as to display graphically the attractor and/or the system variables. Some examples of the application of the program have also been briefly reported. Future works include the extension of this program to the case of discrete systems, the implementation of especialized routines for the control of chaos [6,7,8] and the synchronization of chaotic systems [10,11]. This research has been supported by the Spanish Ministry of Education and Science, Project Ref. #TIN2006-13615.

References

1. Chua, L.O., Komuro, M., Matsumoto, T. The double-scroll family. IEEE Transactions on Circuits and Systems, **33** (1986) 1073-1118
2. Dhooge, 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
3. Dhooge, A., Govaerts, W., Kuznetsov, Y.A.: Numerical continuation of fold bifurcations of limit cycles in MATCONT. Proceedings of CASA'2003. Lecture Notes in Computer Science, **2657** (2003) 701-710
4. Gálvez, A. Iglesias, A.: Symbolic/numeric analysis of chaotic synchronization with a CAS. Future Generation Computer Systems (2007) (*in press*)
5. Govaerts, W., Sautois, B.: Phase response curves, delays and synchronization in Matlab. Proceedings of CASA'2006. Lectures Notes in Computer Science **3992** (2006) 391-398
6. 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
7. 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
8. Iglesias, A., Gutiérrez, J.M., Guémez, J., Matías, M.A.: Chaos suppression through changes in the system variables and numerical rounding errors. Chaos, Solitons and Fractals, **7**(8) (1996) 1305-1316
9. 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
10. Iglesias, A., Gálvez, A.: Analyzing the synchronization of chaotic dynamical systems with Mathematica: Part I. Proceedings of CASA'2005. Lectures Notes in Computer Science **3482** (2005) 472-481
11. Iglesias, A., Gálvez, A.: Analyzing the synchronization of chaotic dynamical systems with Mathematica: Part II. Proceedings of CASA'2005. Lectures Notes in Computer Science **3482** (2005) 482-491
12. Iglesias, A., Gálvez, A.: Revisiting some control schemes for chaotic synchronization with Mathematica. Lectures Notes in Computer Science **3516** (2005) 651-658
13. Lorenz, E.N.: Journal of Atmospheric Sciences, **20** (1963) 130-141
14. The Mathworks Inc: Using Matlab. Natick, MA (1999)
15. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes (2nd edition), Cambridge University Press, Cambridge (1992)
16. Sarafian, H.: A closed form solution of the run-time of a sliding bead along a freely hanging slinky. Proceedings of CASA'2004. Lecture Notes in Computer Science, **3039** (2004) 319-326
17. Zhou, W., Jeffrey, D.J. Reid, G.J.: An algebraic method for analyzing open-loop dynamic systems. Proceedings of CASA'2005. Lecture Notes in Computer Science, **3516** (2005) 586-593