Teaching Dynamical Systems with Maxima

Jaime E. Villate

University of Porto, Faculty of Engineering
Rua Dr. Roberto Frias, Porto 4200-465, Portugal,
villate@fe.up.pt

WWW home page: `http://villate.org/`

We report on our experience in teaching a dynamical systems course to second-year engineering students. The course has been taught with a hands-on approach, using a Computer Algebra System: Maxima. We have developed some additional programs to help students explore the dynamics of the systems, in a more intuitive way and without losing too much time with computer programming. The result has been very encouraging; students are better motivated using this approach, as compared to the more traditional methods that we used in the past.

1. Mechanics for Engineering students

Our engineering students in the University of Porto have to take two courses in Physics during their first or second year of studies. The first course is a course on Newtonian mechanics. When they attend that course, they have already taken one or two semesters of calculus, and during their high-school studies they must have already studied particle kinematics and dynamics. The main goal of the first course in physics is then to solve the equations of motion for some mechanical systems; for example, for a one-dimensional system the equations of motion are:

$$\dot{x} = v \tag{1}$$

$$\dot{v} = \frac{F(x,v)}{m} \tag{2}$$

where x is the position, v the speed, m the mass and F is the total external force, which might depend on x and v.

In general, equations 1 and 2 lead to a non-linear system. In order to solve those equations analytically, a traditional introductory course in mechanics will only deal with two cases that lead to linear systems: constant forces and forces which do not depend on v. In the first case, we usually study in detail the motion of projectiles in a constant gravitational field neglecting friction. In the second case we study in detail the harmonic oscillator without damping.

The problem with that traditional approach is that students end up memorizing the results for motion under a constant force and for the harmonic oscillator and loose track of the more general picture of the equations of motion. On the other hand, those cases that lead to linear systems correspond to idealized systems which are very unrealistic. Consider for instance the case of projectile motion; if we ask a student to calculate how far a ball will travel if he throws it horizontally from his shoulder height and with an initial speed of 20 m/s, the student's intuition will tell him/her that the answer should be very different for different kinds of balls: tennis, ping-pong, baseball, etc.

Nowadays that students have personal computers and calculators it does not make sense to limit ourselves to problems that have an analytical solution; by teaching them to solve differential equations numerically, we can ask them to solve more realistic problems including friction and other effects. They can even be exposed to more modern topics in dynamics, such as chaos and fractals.

We can also take advantage of the use of the computer in the classroom to use a new approach to the study of differential equations which has been advocated by several teachers of that field [1]. That new methodology uses graphical analysis extensively to give the students a better insight into the nature of a system of differential equations and its solutions. The course we describe in this paper has been dubbed Dynamical Systems, and substitutes for the traditional course on mechanics.

2. Computer Algebra Systems

Students will get distracted from the goals of the dynamical systems course if they have to make their own computer programs. While if they are given results obtained by somebody else they will not gain the experience necessary to comprehend the concepts involved.

A good alternative is to use a Computer Algebra System (CAS). A CAS allows students to manipulate differential equations, find eigenvectors of a matrix and do some other essential calculations for the analysis of a dynamical system [2]. A CAS also includes a high-level programming language that will allow students to do some simple programming without having to

learn the more complex syntax of a general-purpose programming language. They can thus spend more time analyzing the physics of the solutions.
A CAS includes programs to solve differential equations numerically. There are several libraries written in various different programming languages, which can also be used to solve differential equations numerically. The advantage of using a CAS, over using one of those libraries directly, is that the differential equations to be solved can be passed to the program directly and could be the result of some other algebraic simplification made with the CAS. The equations being solved and the solutions obtained can also be graphed or manipulated to do some further analysis, for instance, to determine the stability of the system.
A CAS will also provide a way for students to type in mathematical equations. Consider for instance a partial derivative:

$$\frac{\partial^3(5xy^2)}{\partial x \partial y^2}$$

In Maxima, this would be typeset as:

```
diff(5*x*y^2, x, 1, y, 2)
```

Using this kind of notation, students have to be more rigorous. In some cases, they even get a better insight into the mathematical notation when they have to type an equation with the keyboard and in a single line of text.
A CAS system allows students to spent less time with algebraic calculations, leaving more time for physical analysis; it also allows students to get a geometric visualization of the solutions of a differential equation in phase space.

3. Maxima

Maxima [3] is a CAS distributed with a Free Software license. It is a descendant of one of the oldest Computer Algebra Systems, Macsyma, developed in the late 1960's at the M.I.T.
We use Maxima as the basic tool for our course on dynamical systems. Being free software, we have been able to study its source code and create two additional packages for our course: **plotdf**, which plots the direction field for differential equations with two state variables, and **dynamics** which includes several programs to plot fractals and bifurcation diagrams and a program to solve differential equations numerically. Those two additional

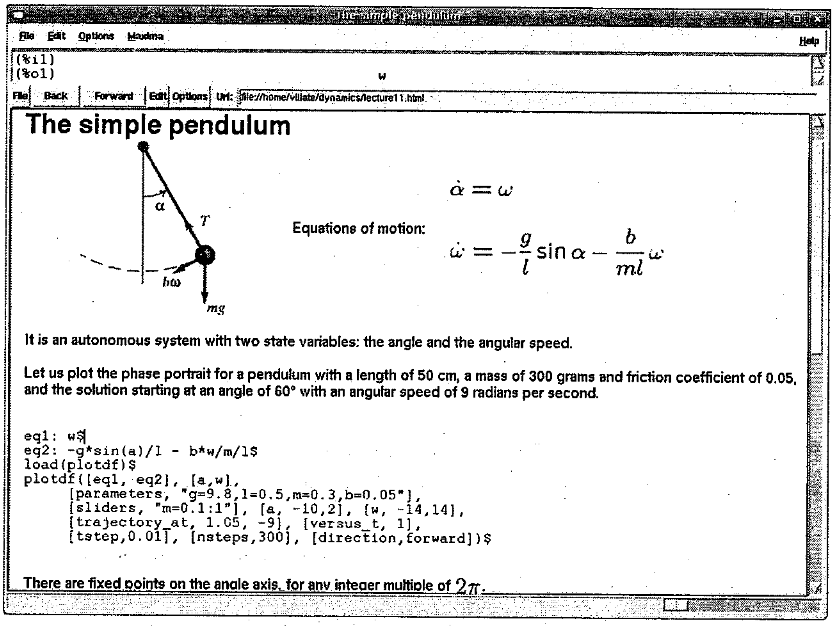
packages are distributed with the official distribution of Maxima. The documentation included in Maxima describes those two packages in detail.
Maxima is written in Lisp and the graphical interfaces we use are written in Tcl/Tk. There is a very active development team who have helped us with the programming of those additional packages.
Creating additional packages for Maxima can be made using its own programming language or in Lisp, which is the language in which Maxima is written. A good reference for programming in Lisp for Maxima is the book by Norvig [4].
As any other CAS, Maxima has a syntax that at first can be unfamiliar to students. For instance, they have a tendency to use the equal sign as an assignment operator, as done in most computer languages, and not with the usual mathematical meaning. They also have to become familiar with the greater variety of objects that can be manipulated in a CAS: rational numbers, algebraic expressions, mathematical equations, etc.
Once they overcome that initial stage, they get very motivated by the many features of a CAS: factoring algebraic expressions, solving equations, differentiating and integrating functions, and so on.

4. Some sample sessions

The version of Maxima that will be used in the examples on this section is 5.12, released in April of 2007.
Figure 1 shows a sample session where the simple pendulum is studied, as shown in the graphical interface Xmaxima. The maxima commands at the bottom can be clicked by the student to enter them into Maxima. After entering the two equations and loading the package `plotdf`, The command we used to plot the direction field for that system was:

```
plotdf([eq1, eq2], [a,w],
      [parameters, "g=9.8,l=0.5,m=0.3,b=0.05"],
      [sliders, "m=0.1:1"], [a, -10,2], [w, -14,14],
      [trajectory_at, 1.05, -9], [versus_t, 1],
      [tstep,0.01], [nsteps,300], [direction,forward])$
```

The first argument for the program `plotdf` is a list with the two dynamical equations, followed by another list that names the two state variables, in the same order that their derivatives were given. The option `parameters` is used to fix numerical values for the parameters in the equations. Option `sliders` will draw a sliding bar that will allow changing the value of the parameter m in real time when the plot is seen. Following the sliders option,



```
(%i1)
(%o1)                                w
```

The simple pendulum

Equations of motion:

$$\dot{\alpha} = \omega$$

$$\dot{\omega} = -\frac{g}{l}\sin\alpha - \frac{b}{ml}\omega$$

It is an autonomous system with two state variables: the angle and the angular speed.

Let us plot the phase portrait for a pendulum with a length of 50 cm, a mass of 300 grams and friction coefficient of 0.05, and the solution starting at an angle of 60° with an angular speed of 9 radians per second.

```
eq1: w$
eq2: -g*sin(a)/l - b*w/m/l$
load(plotdf)$
plotdf([eq1, eq2], [a,w],
      [parameters, "g=9.8,l=0.5,m=0.3,b=0.05"],
      [sliders, "m=0.1:1"], [a, -10,2], [w, -14,14],
      [trajectory_at, 1.05, -9], [versus_t, 1],
      [tstep,0.01], [nsteps,300], [direction,forward])$
```

There are fixed points on the angle axis, for any integer multiple of 2π.

Figure 1 Sample session in Xmaxima, about the simple pendulum.

we defined the domain that should be shown for the two state variables a and w (which stand for α and ω).
Option `trajectory_at` is used to draw, on top of the direction field, the solution of the system with the initial values given. More solutions can be plotted interactively just by clicking on the initial point in the direction field. Option `versus_t` will produce a second graph, where the state variables are plotted as a function of time, for the solution that was defined with `trajectory_at`. Options `tstep` and `nsteps` were used to modify the default values for the time intervals and number of steps used to obtain the solution numerically. Finally option `direction` obtains the numerical solution for times intervals increasing from the initial instant, and not for times before and after that instant, which is the default.
The result is shown in Figure 2. In the graphics windows where those plots appear there is also a menu with several options, including saving the plots as Postscript files.
A session as the one shown in Figure 1 corresponds to one of the weekly two-hour sessions that the students attend in the course. Those sessions are

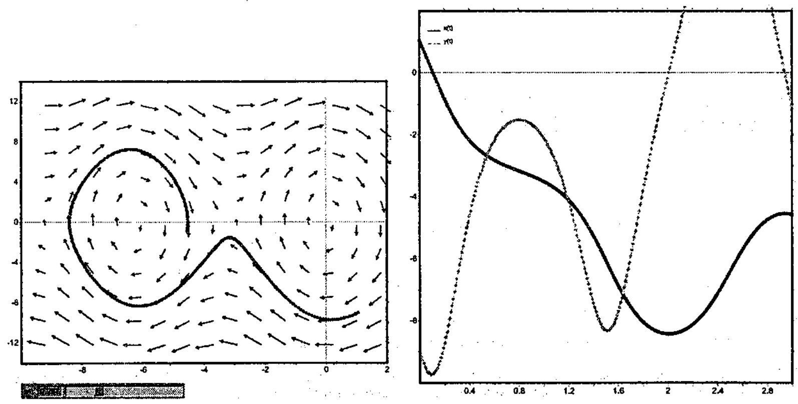


Figure 2 Direction field for a simple pendulum, showing a particular solution on the phase space (left) and as a function of time (right).

held in groups of around 20 students, at computer rooms equipped with one computer for every two students. They can work either in GNU/Linux or the Windows operating system. In each of those sessions students will log in to a **Moodle** server where they can find the session they have to study that week, as well as a list of problems that they are expected to solve during the session. Moodle [5] is a very popular Course Management System, namely, an e-learning system that helps teachers create a course Web page with several predefined activities that can be used, for instance, forums, quizzes, chat sessions and many more.
The students must open the session with Xmaxima, and follow the steps. After that, they can work on the problems proposed. At the end of each session, each student will submit to the Moodle server a batch file with the commands used to solve the problems. The lecture notes provided to the students have become a book that can be downloaded from the Web [6].
Another example, solved in one of the sessions, is the discrete mapping:

$$x \mapsto 0.6\,x\,(1 + 2\,x) + 0.8\,y\,(x - 1) - y^2 - 0.9 \tag{3}$$

$$y \mapsto 0.1\,x\,(1 - 6\,x + 4\,y) + 0.1\,y\,(1 + 9\,y) - 0.4 \tag{4}$$

which leads to a chaotic system. After saving the last side of the two equations above into two Maxima variables `f` and `g`, and after loading the additional package `dynamics`, The students can obtain a plot of several iterations of the system using the Maxima command:

```
evolution2d([f, g], [x, y], [-0.5, 0], 50000, [style, dots])$
```

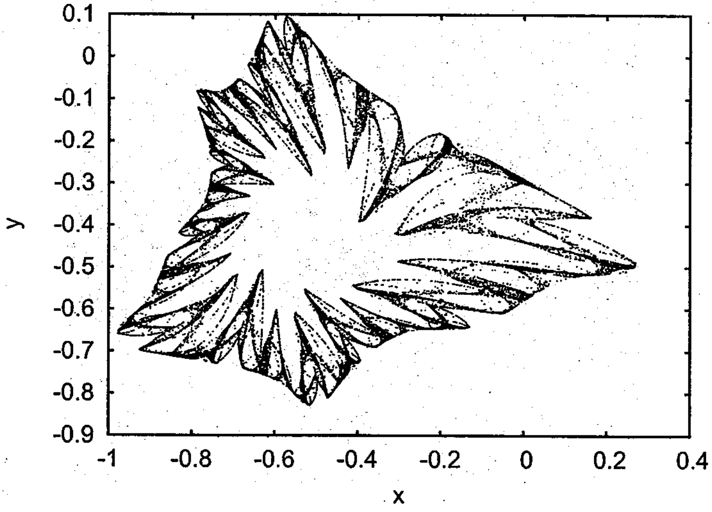


Figure 3 The orbit of a chaotic discrete system in two dimensions, showing the state at 50000 steps.

Figure 3 shows the graph obtained for 50 000 iterations with starting values $(x, y) = (-0.5, 0)$. The students can enlarge a portion of the graph, to see its fractal nature.
The package dynamics also includes a program to draw the Julia and Mandelbrot sets. For example, to draw the Julia set for the complex number $(-0.75 + 0.1\,i)$, we would use a command as the following:

```
julia(-0.75, 0.1, [levels, 160], [radius, 1])$
```

Figure 4 shows the result. 160 iterations were used, and the option radius was used to show only the region of the complex plane z with: $-1 < \mathrm{Re}(z) < 1$ and $-1 < \mathrm{Im}(z) < 1$ (the center is by default at the origin).

5. Results and Conclusions

The Dynamical Systems course we have described is attended by first-year undergraduate students majoring in Computing and Information Engineering. There is an average enrollment of around 120 students.

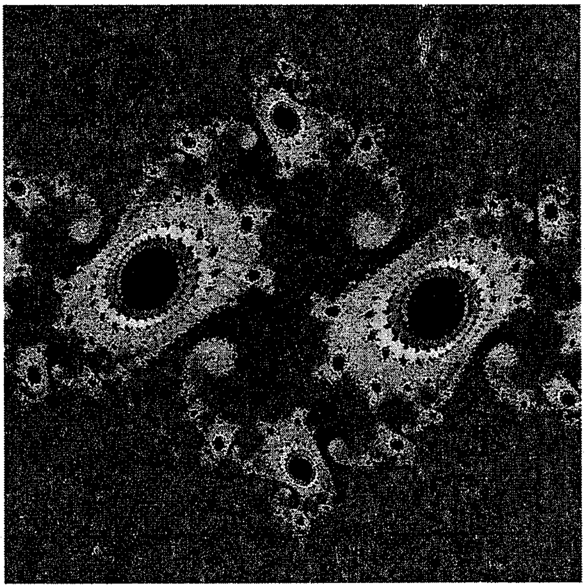


Figure 4 A graphical representation of a Julia set.

The evaluation of the course is made through multiple-choice quizzes, which are then graded in the Moodle server, and with a term paper on a project given to each student. Those term projects involve some bibliographic research and analyzing some dynamical system using Maxima.
We have been teaching this course on dynamical systems in our School since the Fall of 2003. The traditional course on Mechanics, as well as most courses on Physics and Mathematics taught at our School, used to have very low approval rates. During the years that we were involved with the teaching of the mechanics course, the approval rate was close to 50%.
The year when we introduced the new course on Dynamical Systems, the approval rate increased dramatically to over 90%. That improvement is due in part to the use of a more active teaching methodology and to the change in evaluation method, using a term paper instead of a final exam. However, it is very clear that the students' motivation has increased and they are much more interested in the modern topics included with the new course. The feedback from the students has been very positive and the results from yearly student surveys conducted at our School show that they are very satisfied with this new course.

Computing tools have proved to be a very valuable aid for teaching the subject of dynamical systems. Maxima has been used successfully in a class with several students. Using CAS tools some tasks are much easier for the students, but that does not mean they have to work less: by lowering the initial barriers, students move deeper into more advanced topics which are more demanding in other skills. What happens is that the reasoning expected from the students has shifted from mastering algorithms and analytical methods, into getting a better understanding on how a system evolves.

Bibliography

[1] Kallaher, M. J., editor: Revolutions in Differential Equations. Exploring ODEs with Modern Technology, The Mathematical Association of America, ISBN 0-88385-160-1, U.S.A. (1990)
[2] Redfern, D., Chandler, E. & Fell, R. N.: Macsyma ODE Lab Book. Jones and Bartlett Publishers, U.S.A. (1997)
[3] Homepage of the Maxima Project: http://maxima.sourceforge.net
[4] Norvig, P.: Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp. Morgan Kaufmann publishers Inc., U.S.A. (1992)
[5] Homepage of Moodle: http://moodle.org
[6] Villate, J. E., Introduction to Dynamical Systems: A Hands-on Approach with Maxima, ISBN 972-99396-0-8, Porto (2006)
http://fisica.fe.up.pt/maxima/dynamicalsystems/