Adding an Interactive Component to Computer Algebra in Differential Equations

Robert Decker
University of Hartford
rdecker@hartford.edu
uhaweb.hartford.edu/rdecker

Introduction

The need to add an interactive/dynamic component to computer algebra systems such as Maple, Derive or Mathematica is just beginning to be addressed. For example, Maple now has an Interactive ODE Analyzer, which provides a minimal graphical user interface (GUI) for working with differential equations.

I will discuss some (platform independent) interactive programs that I have developed for investigating the graphs of first and second order differential equations, as well as the graphs of functions, parametric curves and data points. With these programs one can dynamically change parameters or initial conditions, and see the results immediately in multiple views (phase or time plots). The results can then be imported into Maple for further refinement there, and for report writing. Algebraic results from Maple can also be exported back for further interactive graphical manipulation. Thus, for example, the exact solution to an ODE that has one or more parameters can be graphed and manipulated, and compared to numerical approximations.

In addition I will discuss how you can create your own customized applet using tools that I have developed. I will conclude with some projects that I have used in differential equations courses that demonstrate how students benefit from the synergy created by combining dynamic software and computer algebra.

The Course

The software that I have developed has been primarily directed toward investigating problems in a standard first course in ordinary differential equations for engineers and scientists (though it is also appropriate for calculus and precalculus courses). In this course, I give projects for which the students must submit a written paper. The students use Maple for the mathematics, the graphs, and for the write-up (though some prefer to paste the Maple output into Word and do the write-up there).

The papers are graded on the mathematics, the writing, and the presentation. In particular, graphs should be well-chosen, clearly labelled, and informative. The paper should tell a story, and the graphs and Maple output should be part of the story.

I found that the limitations of computer algebra made it difficult for my students to investigate the problems that I presented, and difficult to write papers that told a coherent story. In these projects one of the main themes

is to explain how a graph changes as a parameter or initial condition changes. Computer algebra systems have the following limitations:

Limitations of Computer Algebra

- Feedback to changes in parameters is slow
- It is easy to miss behavior for "in between" values of parameters
- It is hard to generate accurate phase portraits for DE's with multiple initial conditions

Programs and Development Tools

Several years ago I began writing small interactive programs for graphing functions and differential equations using the platform-independent language Tcl/Tk. These first programs could be called applets, and are similar to the java applets that some mathematics educators have developed and are using on the world wide web today (in this paper I will use the terms "program", "applet", and "application" interchangeably). I choose Tcl/Tk over the more common Java because I found that it was the easiest way to quickly develop mathematics programs with graphical user interfaces (GUIs) for someone without a lot of background in computer science and object oriented programming. Tcl/Tk is a standard tool in industry and academia for rapid development of GUIs.

Over a period of years, the programs became more elaborate, and I found it convenient to create "helper" functions which could be used to create the components of a typical applet. Thus I developed software tools which can be used to create customized applets directed at a particular problem, or more general programs for investigating any equation or system of 2 equations. The types of graphs that can be generated are:

Types of Graphs

- Function plots
- Parametric function plots
- First order differential equation plots
- Plots for systems of differential equations (second order equations included)
- Two variable data plots

Graphs can be created that combine any number of the above, and the results can be displayed on any number of coordinate systems simultaneously (thus creating different views of the same function). Parameters can be controlled by a slider, with all changes reflected in all coordinate systems immediately.

Also, initial conditions for differential equations can be changed interactively by clicking and dragging, again with changes reflected in all views of the differential equation (or system of differential equations).

In Figure 1 we show a screen shot of a typical program (applet) which graphs a first order differential equation using two different numerical methods (Euler and RK2), and a function (the exact solution to the differential equation). The differential equation is the standard logistic equation $p' = ap(1-p/b)$ often used for modelling population growth. This program is targeted at comparing two different numerical methods to the exact solution for this equation, with some surprising results for high growth rates (large a values). We will return to this example later.

While some mathematics educators may be interested in just using some of the programs (applets) such as the one in Figure 1, others may be interested in creating their own, which are targeted at a specific problem. We now briefly discuss the structure of a typical program, and look at some code.

The programmer must create data (via functions, parametric functions, differential equations or raw data points) with variable names for the data, and coordinate systems with variable names labelling the axes. Then one specifies which data gets plotted on which coordinate systems; if the variable names match, a graph appears. We summarize below:

Structure of the Programs

- Create coordinate systems and data
- Specify which data gets plotted on which window
- Graph will appear if variables match

In addition to the above, you need to specify the window arrangement for your applet, and some initial values for the slider.

In Table 1 we show some code that creates two coordinate systems; the first block creates sys1 and the second creates sys2. The values of the horizontal and vertical minimum and maximum are set here, as well as the axis variables.

In Table 2 we show some code that creates one function and one system of two differential equations. The first block of code creates the function, and the second creates the system of differential equations. The formulas for the function and de system are set here, as well as other relevant quantities such as minimum and maximum values for the independent variables, dependent and independent variable names, color, and which coordinate system(s) the data is to be plotted on. For the differential equation system, quantities such as numerical solution method, step size, initial values, direction field resolution and which coordinate system(s) the direction field is to be plotted on.

In Figure 2 we show the application that is created by this code. The subroutines called in the last statement in each of the code blocks in Tables 1 and 2 (coordsys creates coordinate systems, NewFunction creates a function,

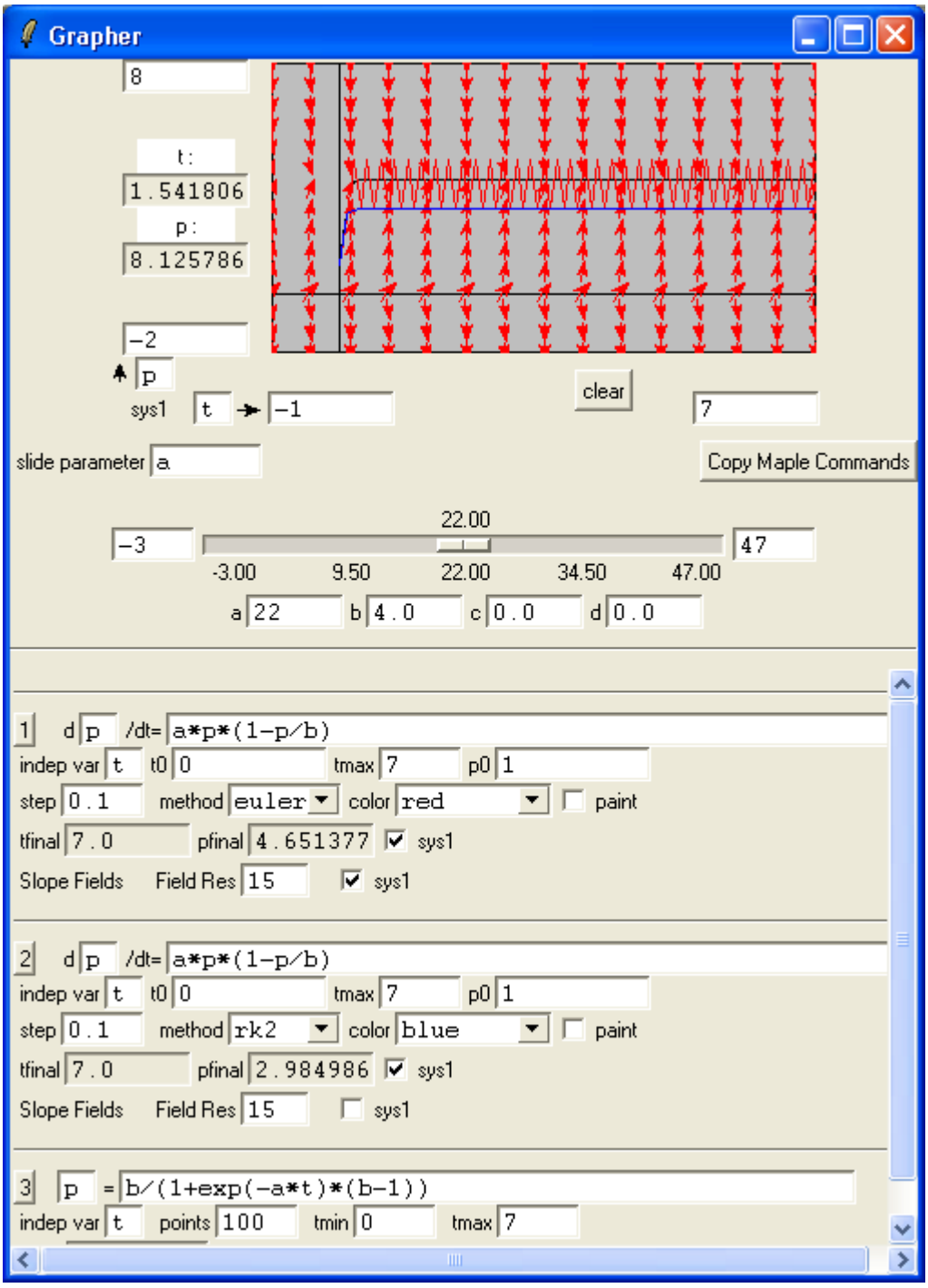

Figure 1: Three solution methods for $p' = ap(1 - p/b)$

```
#create coordinate systems
set sysNum 0

incr sysNum
set name sys$sysNum
set hmin($name) 0
set hmax($name) 10
set vmin($name) -10
set vmax($name) 10
set hvar($name) t
set vvar($name) x
set coordsysOut($name) [coordsys $graphWin $name $hmin($name) \
$hmax($name) $vmin($name) $vmax($name) $xpicmax $ypicmax \
$hvar($name) $vvar($name)]

incr sysNum
set name sys$sysNum
set hmin($name) -10
set hmax($name) 10
set vmin($name) -8
set vmax($name) 8
set hvar($name) x
set vvar($name) y
set coordsysOut($name) [coordsys $graphWin $name $hmin($name) \
$hmax($name) $vmin($name) $vmax($name) $xpicmax $ypicmax \
$hvar($name) $vvar($name)]
```

Table 1: Code to create coordinate systems

```
#create functions, de system
set graphNum 0

#a function
incr graphNum
set function($graphNum) a*sin(d*t)
set indepVar($graphNum) t
set depVar($graphNum) x
set indepMin($graphNum) 0
set indepMax($graphNum) 10
set numPts($graphNum) 101
set color($graphNum) yellow
set sysList($graphNum) [list sys1 sys2]
NewFunction $graphNum $funcsWin

#a de system
incr graphNum
set paint($graphNum) 0
set fDesys($graphNum) y
set gDesys($graphNum) -b*y-c*x
set indepVar($graphNum) t
set fDepVar($graphNum) x
set gDepVar($graphNum) y
set indepMin($graphNum) 0
set indepMax($graphNum) 10
set initialDep1($graphNum) 5
set initialDep2($graphNum) 5
set step($graphNum) 0.1
set solutionType($graphNum) rk2sys
set color($graphNum) red
set sysList($graphNum) [list sys1 sys2]
set fieldSysList($graphNum) [list sys1 sys2]
set fieldRes($graphNum) 15
NewDesys $graphNum $funcsWin
```

Table 2: Code to create a function and a de system

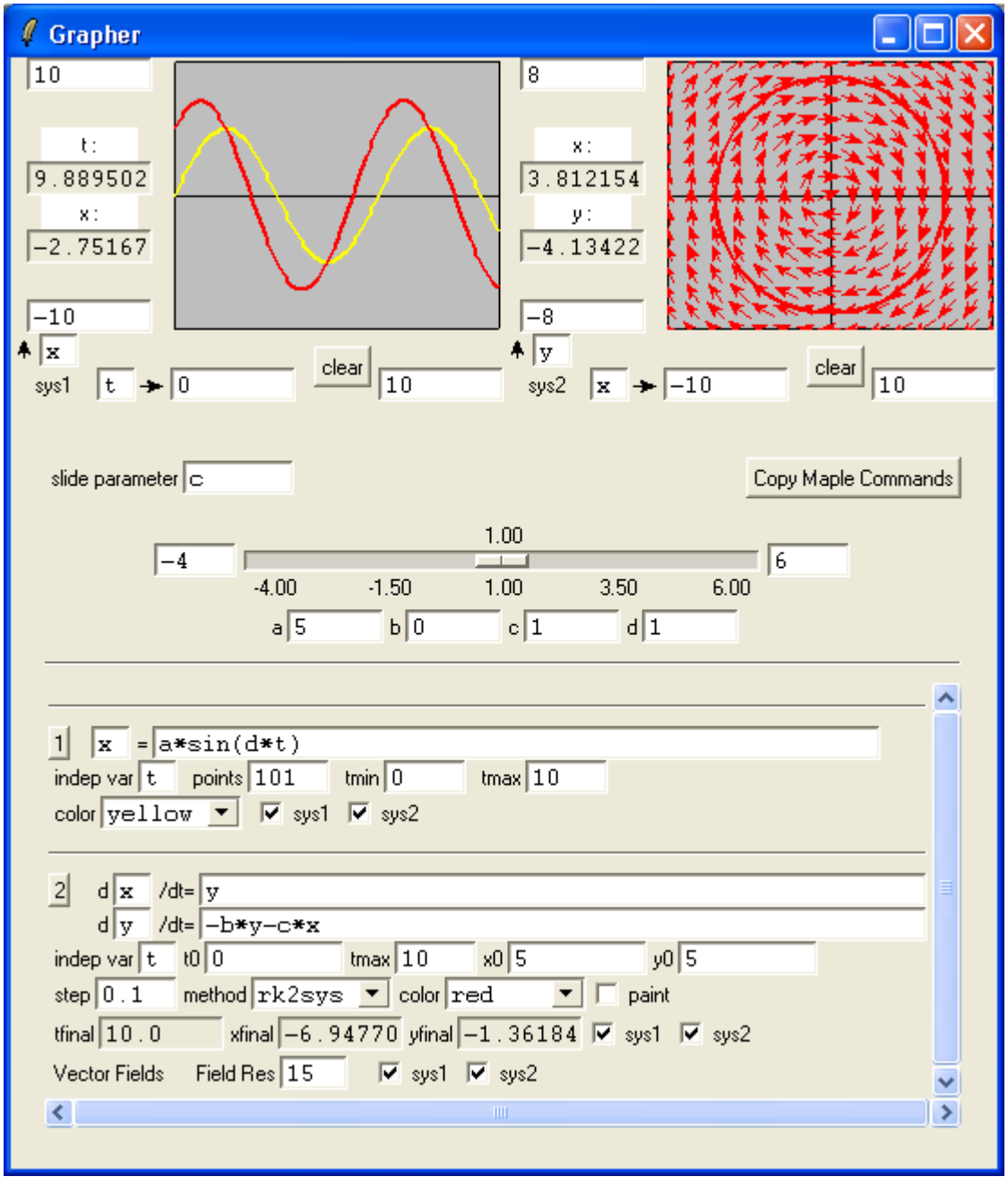

Figure 2: Two views of a function and a system of differential equations

and NewDesys creates a system of differential equations) comprise the tools that I have developed to make it easy to create applets. These subroutines get included by using "source" statements (the Tcl/Tk version of an "include" statement). The entire code that is needed to create this application, including a few additional statements for setting initial slider values and window placement, and the "source" statements referred to above, can be obtained at my website uhaweb.hartford.edu/rdecker.

All of the values that are set as described above are adjustable once the program starts. Notice that even though the function $x = a * \sin(d * t)$ is to be plotted on both coordinate systems one and two (sys1 and sys2), a graph only appears on sys1. This is because the variables for this function, x and t, match the variables on sys1 but not on sys2. For the differential equation system, a time plot appears on sys1 (since x is the first dependent variable, and t is the independent variable), and a phase plot appears on sys2 (since x and y are both dependent variables). A direction field appears on sys2 but not sys1 because a direction field does not make sense for a time plot.

Once the program is running (as in Figure 1 or Figure 2), the operation is pretty straightforward. We summarize below:

Basic Operation of Programs

- Edit any input boxes, then press Enter to see changes
- Parameters a, b, c, d can appear in any function or DE
- Any of the four parameters can be changed interactively using the slider
- Use any letters except for a, b, c, d, e for variable names in function/DE definitions and graph windows (use e and pi for the well-known constants)
- Use left-right arrows to trace curves; up-down arrows to move to other curves
- If error message, hit "skip messages" and hope for the best (these come from Tcl/Tk)

Syntax for functions is standard syntax for a TI graphing calculator or Maple (you must use $*$ for multiplication). In addition to the above, for differential equations there are few other important ways to control the programs:

DE Basic Operations

- Click once anywhere in a graph to set initial conditions of any DE's graphed there
- Click and drag to change an initial condition dynamically; changes are reflected in all views simultaneously

- Double click to keep an initial condition and hence generate multiple initial conditions (create a phase portrait this way)
- In paint mode (click the paint checkbox), clicking and dragging generates multiple initial conditions, but they do not persist when a parameter is changed, and cannot be exported to Maple ("paint" a phase portrait this way)

The programs described above and shown in Figures 1 and 2 can, by themselves, be used by students to investigate functions and differential equations with parameters. In conjunction with the computer algebra system Maple, however, a student gets the best of both worlds. First we describe how the graphs created with an applet can be exported to Maple, and how information can be passed back and forth between the two.

Export to Maple

- Hit "Copy to Maple" button, go to a Maple window, then paste at a Maple prompt
- Copy and paste from entry boxes in programs to Maple (use Ctrl-C to copy, Ctrl-V to paste), and from Maple to entry boxes in programs
- A few configurations will not export to Maple exactly as is (e.g. can't switch variables in a coordinate system to get a graph of the inverse)

By exporting results to Maple, one obtains all of the capabilities of a commercial, mature software program. We summarize as follows:

Interplay with Maple

- Maple is reliable
- Maple has output capability
- One can investigate with interactive programs, document in Maple
- One can find exact solutions in Maple, and compare to numerical approximations coming from the interactive programs

In Figure 3 we show the Maple output that results from pressing the "Copy to Maple" button, and then pasting at a Maple prompt, for the application shown in Figure 2. Since you have the Maple commands to work with at this point, you can edit the commands to add titles to the graph, or make minor changes to improve how the graph looks. In effect, you can use an applet as a graphical interface to Maple.

```
> with(DEtools): with(plots):
sys1[2]:=DEplot({diff(x(t),t)=y(t), diff(y(t),t)=-(0)*y(t)-(1)*x(t)},
{x(t),y(t)}, t=0..10, [[x(0)=5, y(0)=5]], x=-10..10, y=0..1,
stepsize=0.1,linecolor=red, color=red, thickness=2, scene=[t,x],
method=classical[rk2],obsrange=false,dirgrid=[15,15]):
sys2[2]:=DEplot({diff(x(t),t)=y(t), diff(y(t),t)=-(0)*y(t)-(1)*x(t)},
{x(t),y(t)}, t=0..10, [[x(0)=5, y(0)=5]], x=-10..10, y=-8..8,
stepsize=0.1,linecolor=red,color=red, thickness=2,
scene=[x,y],method=classical[rk2],obsrange=false,dirgrid=[15,15]):
sys1[1]:=plot((5)*sin((1)*t), t=0..10, view=[0..10,-10..10],
thickness=2, color=yellow):
sys2[1]:=plot((5)*sin((1)*t), t=0..10, view=[-10..10,-8..8],
thickness=2, color=yellow):
display(sys1[2] ,sys1[1]);
display(sys2[2]);
```

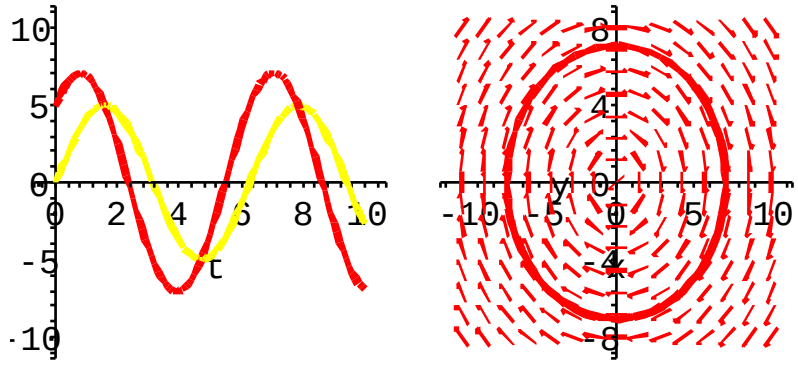

Figure 3: Maple output

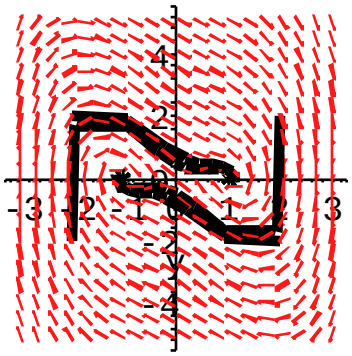

Figure 4: Typical phase portrait for $y'' + y' + y(y^2 - 1)$ generated by a student using Maple alone

Applications in for a Differential Equations Course

We now turn to some examples of the use of interactive applets in the context of a differential equations course. One of the things that students should be able to do in a modern first course in differential equations is to produce a detailed phase portrait for a system of two differential equations (or equivalently a single second order equation). Due to the wide availability of mathematics software, students can study important nonlinear equations in addition to the standard linear ones.

A good phase portrait should "fill out" the window in order to tell a complete story. This involves finding just the right initial conditions for the equation at hand, which is often a difficult task when done with computer algebra alone.

For example, in my class I sometimes have students study a second order differential equation which can be used as a model of a mass-spring system with a repelling force at the origin. The equation that they are asked to study and come to understand is $y'' + by + cy(y^2 - a) = 0$. I ask my students to create a phase portrait, carefully choosing the initial conditions so as to provide a complete description which starting points go where in the long run. In particular I tell them they need to include some points from each quadrant. In Figure 4 we show the type of phase portrait produced by a typical student using Maple, with $a = 1$, $b = 1$, $c = 1$. Such a phase portrait leads students to tell a very misleading story about this equation, such as "Every initial condition on the right side of the graph ends up at the fixed point on the left, and vice versa". Other misleading aspects of the Maple phase portrait are the jagged nature of the solution curves (due to an insufficiently small default step size), and the incomplete nature of the portrait due to not enough initial conditions.

Of course, the poor nature of the phase portrait shown here is in large part

due to the lack of persistence of a typical student in trying more and different initial conditions, and different step sizes (or even recognizing that step size is a problem here). The nature of computer algebra, however, is such that it is quite tedious to experiment with initial conditions or step size, and so a student who has to make decisions about how to allocate her or his time simply goes with the first picture that has some chance of being correct.

In Figure 5 we show an applet designed to investigate this same model. By clicking and dragging the initial condition a very short distance one sees the solution curve flipping back and forth between the two curves shown in Figure 5. This kind of investigation is ideal for understanding behavior near saddle points; finding two such initial conditions that illustrate saddle behavior using Maple alone is quite tedious.

Also, by looking at three views simultaneously, the student gets a better idea of how the phase plot is related to the time plots; for example, we also see how long it takes for a solution curve to "reach" a fixed point. Tracing a curve (left-right arrows) also helps to connect phase and time plots.

After setting a few more initial conditions which help to "fill out" the phase space, and then exporting to Maple we get the phase portrait in Figure 6. Of course, not every student will produce a phase portrait as bad as the one in Figure 4 without an interactive graphing tool, nor will every student produce one as good as the one in Figure 6 with an interactive graphing tool. The point is that I have found that the phase portraits produced by students are more likely to look like the one in Figure 6 if students have such a tool to work with.

It has become common to study differential equations with parameters in a first course, due in part to the availability of mathematics software. In particular, it is possible to have students investigate bifurcation problems using software; however, with computer algebra alone, it is tedious to change a parameter to see how a graph is effected. Immediate feedback, and the ability to see intermediate values is critical in conveying to the student an accurate bifurcation story.

For example, one of the projects I give my students is to investigate the differential equation $\frac{dP}{dt} = aP(1 - P/b) - c$, which is used to model the growth of a population of fish, with a constant amount of fish being removed per unit of time (fishing). As the parameter c ranges from 0 upward (a and b are given fixed values), the student needs generate a sequence of pictures which can be used to tell the story of how the behavior of the fish population changes as the amount of fishing changes. Students know about fixed points at this point in the course, and should incorporate them into their discussion.

The students are intended to discover, and their sequence of pictures should show that with $c = 0$, we get two fixed points, the larger being a sink, and the smaller, occurring at $y = 0$, being a source. As c increases, the larger fixed point decreases in value, and the fixed point which starts at $y = 0$ increases, while the stability of both remains the same. This means that if the initial population starts below the lower (unstable) fixed point, the fish population goes extinct in finite time. As c continues to increase, there comes a point at which the

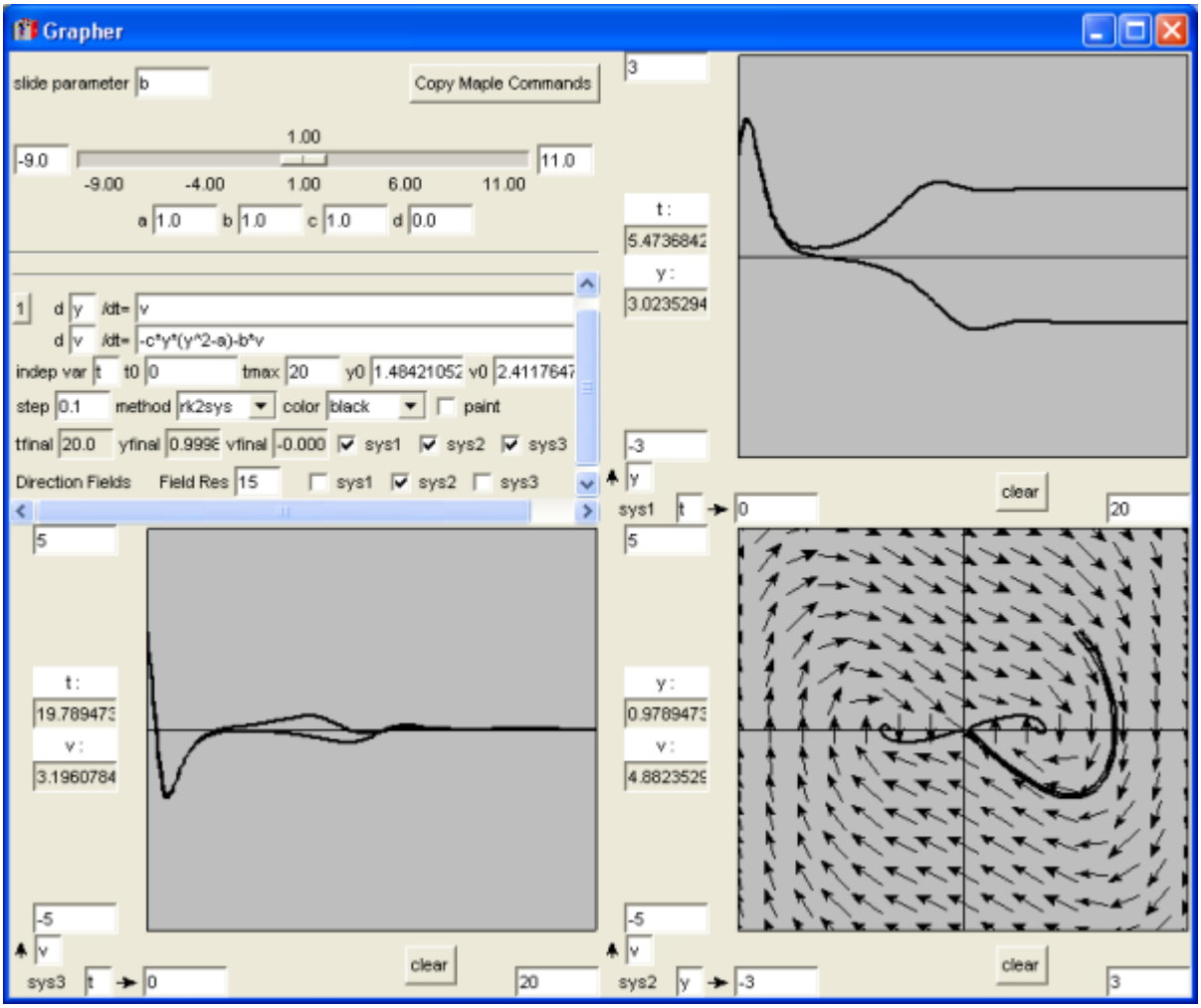

Figure 5: Applet for investigating $y'' + y' + y(y^2 - 1)$

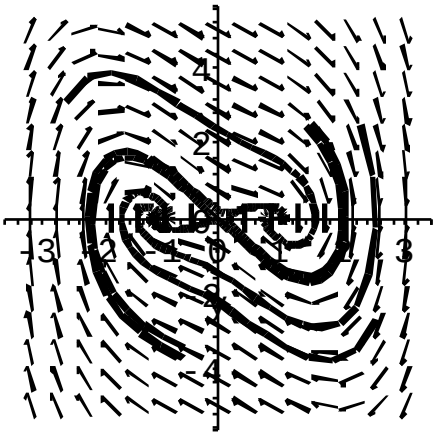

Figure 6: Phase portrait for $y''+y'+y(y^2-1)$ using an applet and then exporting to Maple (compare to Figure 4)

two fixed point come together to form a single fixed point (a node) after which there are no (real values) fixed points. The c value for which there is one fixed point is the bifurcation point; for c values above the bifurcation point, all fish populations go extinct in finite time.

In Figure 7 we see an applet designed to study this problem. There is a slot for one first order differential equation and two functions; my students must find the fixed points of the differential equation using Maple, and paste these into the applet as constant functions. As one moves the slider, one sees the constant solutions approach each other, and then disappear at $c = 0.5$, the bifurcation point (here $a = 0.5$ and $b = 4$). By taking a series of snapshots and exporting them to Maple we get the sequence of pictures in Figure 8, for values of c ranging from 0 to 0.7.

With Maple alone, the difficulty for the student is to choose which c values to include in the sequence of snapshots, and which initial conditions to choose in order to best illustrate the behavior. With computer algebra alone, I have found that students tend to choose c values pretty randomly, and as a result, miss the point of the exercise. Also, if an insufficient number of initial conditions are chosen, it can be easy to miss the solution curves that hit the t-axis in finite time, and hence the relevance of the lower fixed point (the cutoff, below which the population goes extinct). In fact, for many bifurcation problems (either first order equations or systems), the number of initial conditions required to get a good picture depends on the value of the parameter; with an interactive tool, it is easy to add more initial conditions when needed.

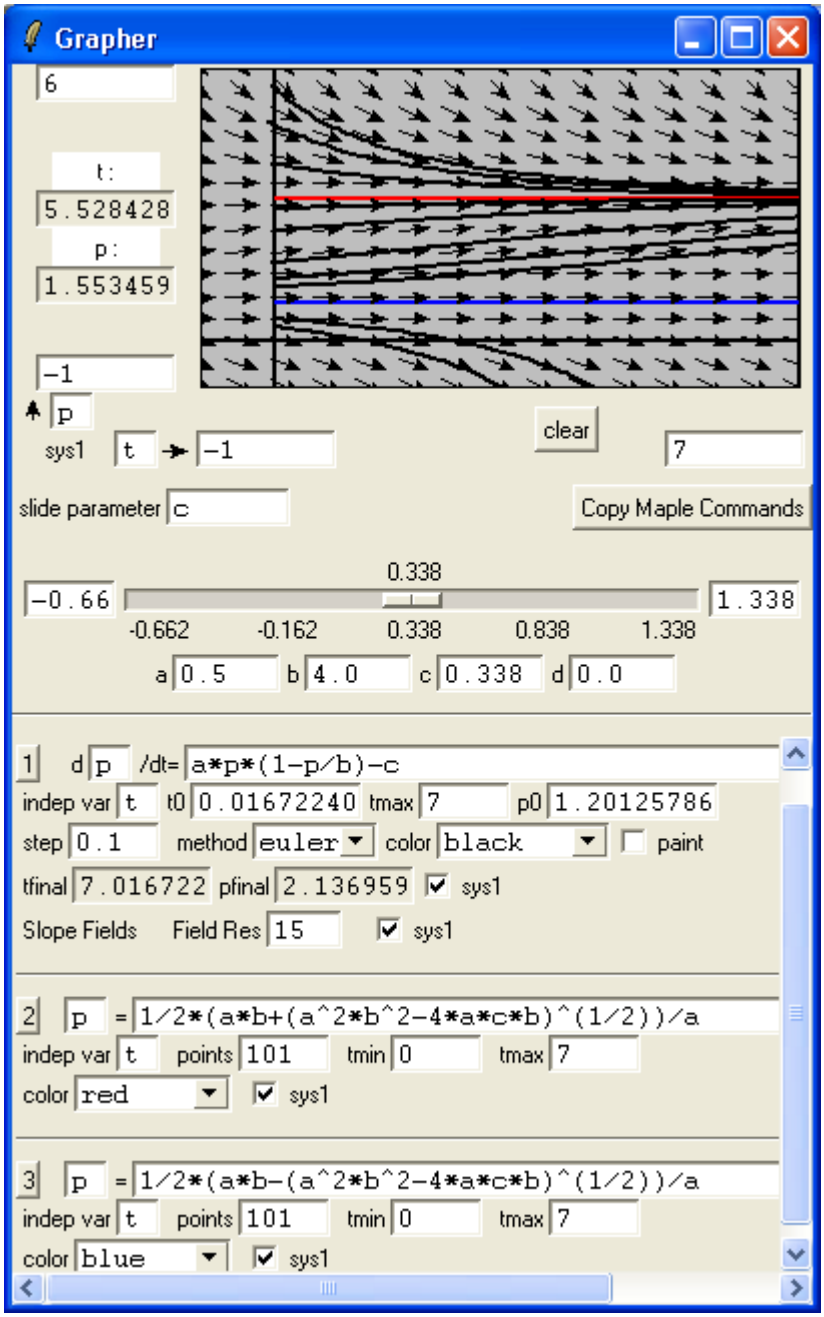

Figure 7: Applet for studying the population model $P' = aP(1 - P/b) - c$

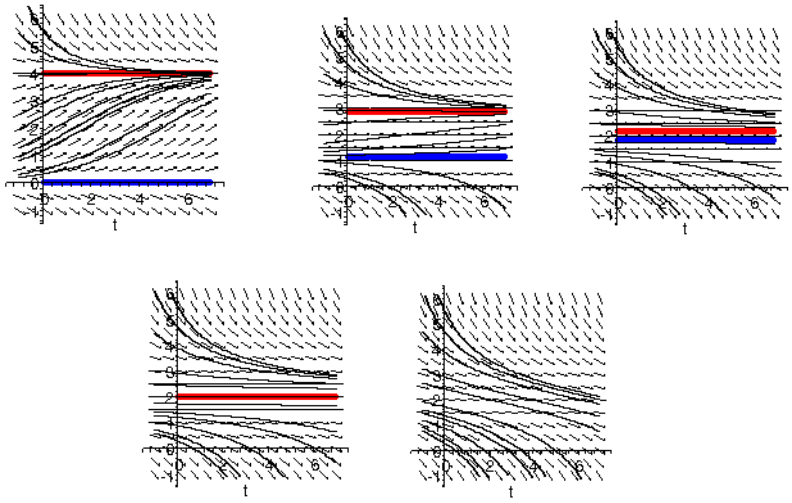

Figure 8: $P' = aP(1 - P/b) - c$ for c ranging from 0 to 0.7, with $a = 0.5$ and $b = 4.0$

I would like to return now to Figure 1 to point out some interesting features of that graph. One can see that the Euler approximation (the jagged red line), the second order Runga-Kutta approximation (blue line), and the exact solution (black line) are all different. The Euler approximation oscillates about the exact solution with period two, which is a well known phenomena when large growth rates are used for the logistic model. The RK2 approximation behaves pretty much as one would expect the exact solution to behave; it increases towards a fixed value, and then remains there. The problem is, it is not the *correct* fixed value that it approaches. This could present a real problem for a numerical method, in that the approximate solution is correct for some parameter values, but when it breaks down for other parameter values, it does so in a way which is not obvious.

It is most interesting to take the applet from Figure 1, start the parameter a at the value 0.5 or so, and increase it using the slider (both step sizes are fixed at 0.1). The two approximate solutions and the exact solution coincide up to about $a = 10$, though the long term behavior remains the same up to $a = 20$, at which point there is a clear bifurcation in both numerical methods. The Euler path becomes periodic with period two (which is stable), and the RK2 solution goes from one stable fixed point for $a < 20$ to two stable fixed points for $a > 20$, with the two stable fixed points coinciding with the two values of the period two orbit of the Euler path. All of this can be confirmed using the standard techniques of discrete dynamical systems. I certainly would never noticed this phenomena, however, if I had not been experimenting with an interactive tool.

I believe that one of the by-products of using interactive/dynamic software tools in studying differential equations would be that more students (and teachers) would begin to discover interesting phenomena, such as the one described above, which could lead to student projects and research. This is similar to what has already happened with increased use of interactive geometry tools such as Geometers Sketchpad. I will conclude with one other interesting observation that I have come across using these applets.

One of my goals in a first course in differential equations is to teach students the standard techniques of linear differential equations so that they can better approach and understand nonlinear one. An ideal model to study with this idea in mind is the standard model of a damped, rigid pendulum given by $y'' + by' + c \sin y = 0$, as it behaves like a linear mass-spring system for small y. Those familiar with this equation will recognize its phase portrait, shown in Figure 9 , with the values $b = 0.3$ and $c = 1.0$. Less commonly depicted is the phase portrait for this equation for larger values of the damping constant b. In Figure 10 we see the same equation with the values $b = 0.3$ and $c = 1.0$. This time I have used the "paint" feature, which means that you click on the box labelled paint, and then click and drag to set many initial conditions at once (I dragged along the top of the picture). What clearly emerges is the shape of a sin curve. One can show that as b gets larger, this shape approaches the shape of the curve $-\frac{c}{b} \sin y$. Again, something unexpected (for me at least, and certainly for students) happens while experimenting with a well known problem.

All of the applets, and the source code for them, can be obtained at my website uhaweb.hartford.edu/rdecker. If you are running Windows, download the .exe file and away you go; if you are running Mac OS or Linux or Unix, you must install Tcl/Tk and then run the source code (called a script). If you have any trouble running the programs, or want to try writing your own applet, please feel free to email me at rdecker@hartford.edu.

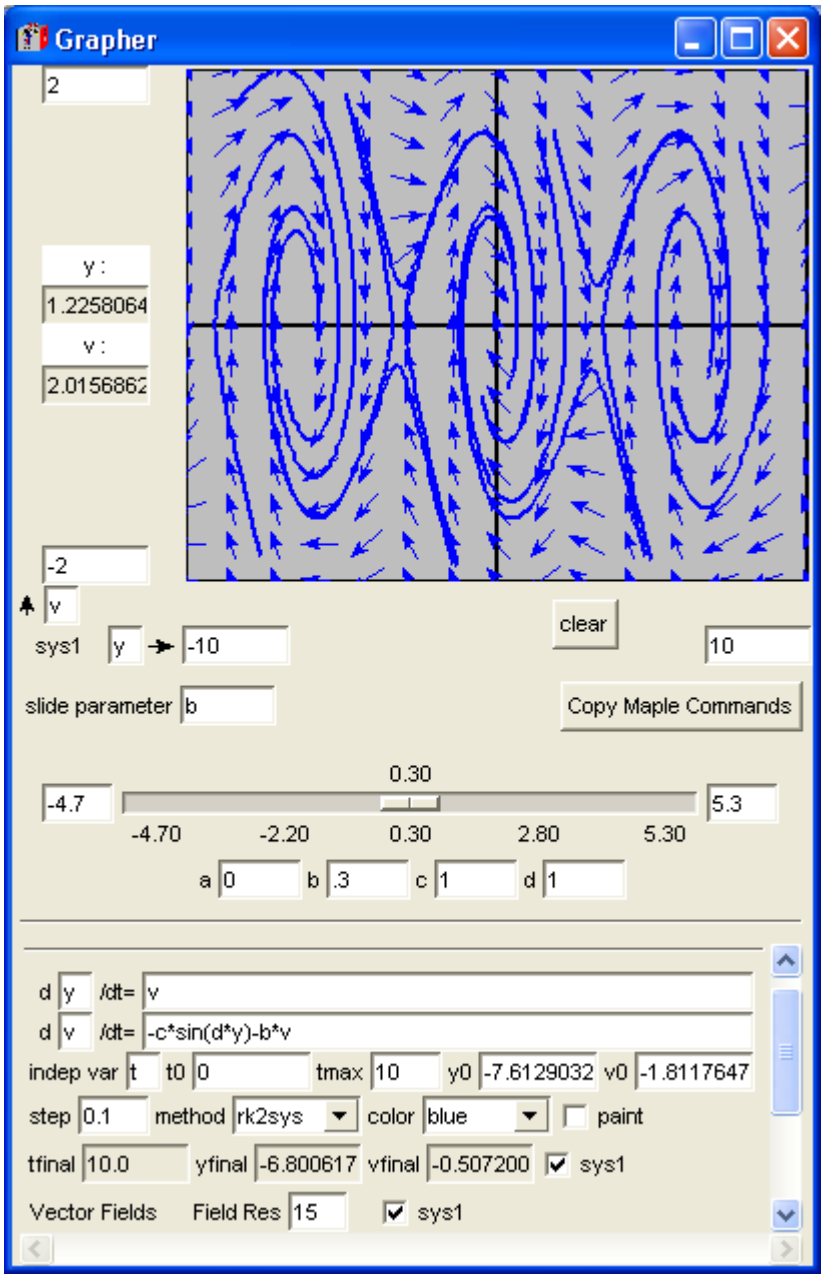

Figure 9: Pendulum equation $y'' + by' + c \sin y = 0$, with $b = 0.3$ and $c = 1.0$

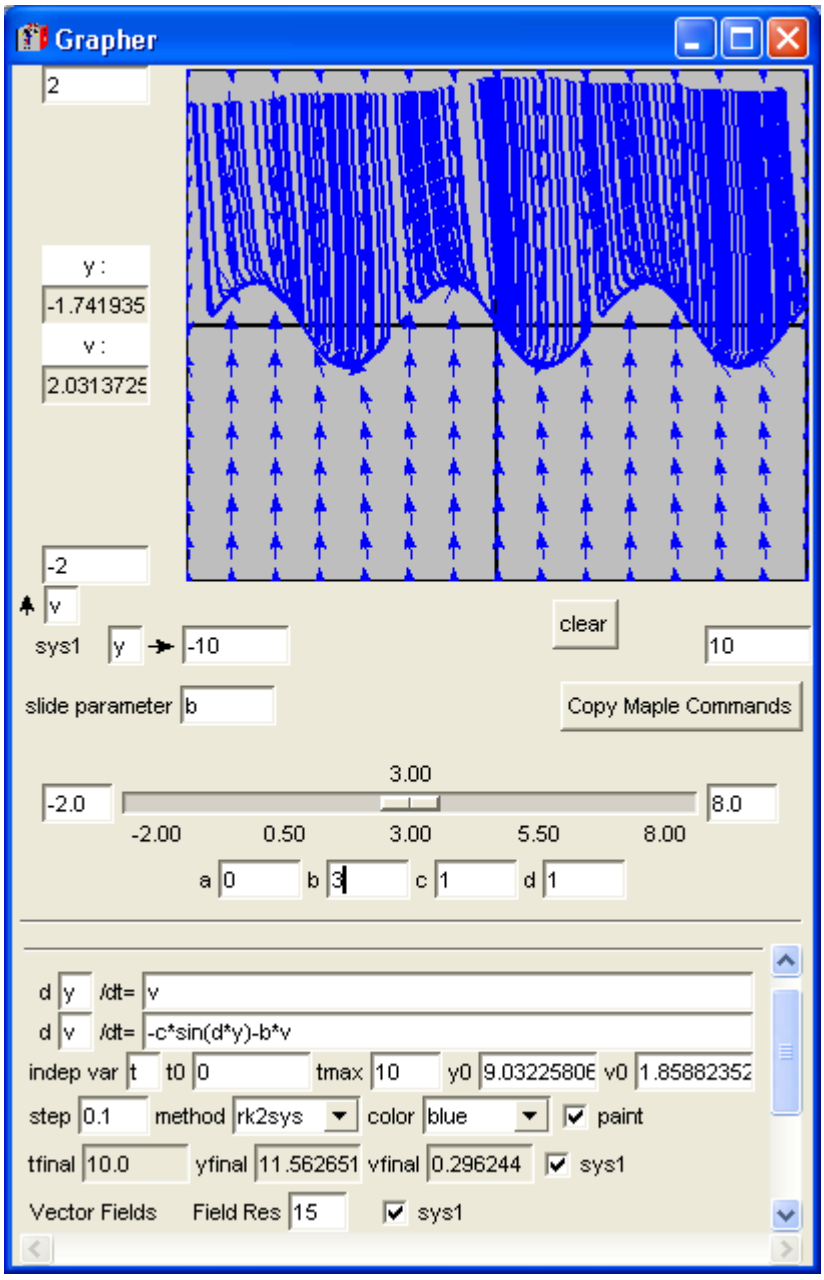

Figure 10: Pendulum equation $y'' + by' + c \sin y = 0$, with $b = 3.0$ and $c = 1.0$