CODEE Journal

Volume 7 *Vol. 7 (2009-2010)* Article 2

8-21-2010

Teaching Introductory Differential Equations with ScicosLab

Stephen Campbell

Ramine Nikoukhah

Teaching Introductory Differential Equations with ScicosLab

Stephen L. Campbell
North Carolina State University

Ramine Nikoukhah
INRIA

Keywords: ScicosLab, Scicos, software
Manuscript received on July 8, 2010; published on August 21, 2010.

Abstract: Perhaps the largest and most capable open source software for doing applied mathematics is the open source package Scicoslab. Widely used in Europe and Asia it is less well known in the United States. In this article we will explain how ScicosLab can be easily used to help teach differential equations and to also introduce students to a software package that they can take with them for the rest of their careers.

1 Introduction

As teachers we like software to be something that helps the student learn and not something that stands in their way. Typically the available software falls into two categories. If software is free, it is limited in capability and there is not good support. If the software has a lot of capability and support and careful documentation, then it is often quite expensive. Matlab, Mathematica, and Maple fall into this second group. There is one notable exception to this dichotomy which is Scilab and Scicos. To understand this software, a little history is needed.

The original Matlab was developed at MIT with federal support. Later the corporate entity Matlab came into being and has developed the Matlab and Simulink available today. This software has become widely used in US academia and industry. It is very capable and also very expensive. The full version with all the toolboxes can cost several thousand dollars per license. In France a different group took the same original Matlab kernel which was in the public domain because of the federal funding and begin building their own software package called Scilab. Initially they were following the Matlab business model and sold the software to individuals and industry. But after a few years they switched to an open source model. The software Scilab is the analogue of Matlab and has the same syntax. Generally commands in Matlab and Scilab are the same or very similar although Scilab has more built in numerical integrators and a slightly friendlier syntax for defining and using functions. Scicos is the analogue of Simulink. A few years ago a corporate consortium was formed to encourage development of Scilab and also to develop a somewhat different package like Scicos. The French research institute, Institut National de Recherche en Informatique et en Automatique (INRIA), where Scilab and Scicos were originally developed, has continued the development of a package called ScicosLab. ScicosLab is the particular package we shall talk about.

ScicosLab is a free software package. Its icon is the puffin . The gtk refers to the compiler used. ScicosLab includes both Scilab and Scicos. It has capabilities equivalent to Matlab and Simulink and most of the toolboxes in Matlab. It has additional capabilities as discussed in Section 3. .The official ScicosLab website is `http://www.scicoslab.org/`. Links are there to obtaining downloadable versions for a wide variety of different machines and operating systems. The software has full online documentation in both English and French. In addition there is an introduction to the latest version of ScicosLab [3]. Scilab has commands for drawing phase portraits. `portrait` draws a two dimensional phase portrait, `portr3d` draws a three dimensional phase portrait, and `boucle` draws a two dimensional phase portrait with observer. Here we shall focus just on showing how to quickly let students use Scicos to study differential equations.

2 Simulating an ODE and Input-Output

Scilab has numerical integrators and graphing capabilities and syntax much like Matlab. However, using this syntax is not always easy for beginning students. We will talk here about using Scicos where the manipulations are performed in a more graphical and visual manner. To get to Scicos, one launches ScicosLab and then types `scicos` on the command line. Using Scicos does two things at the same time. For one it simplifies the simulation of a differential equation. Secondly, it encourages the student to think of the differential equation not as an equation but as an input-output relationship which is very important in many areas of applications.

Suppose that we wish to work with a linear system of differential equations

$$x' = Ax + Bu \tag{2.1a}$$
$$y = Cx + Du \tag{2.1b}$$

where $x' = dx/dt$, A, B, C, D are matrices and u, x, y are vector functions. Equation (2.1a) is called the process or the plant. Equation (2.1b) is the output equation.

In Scicos one graphically assembles blocks. A user can define their own blocks and create blocks made up of other blocks. However, Scicos has a large number of predefined blocks arranged in “palettes." A list of all palettes is found under `Palette` or `Pal tree` in the menus at the top of the Scicos window. `Pal tree` is shown in Figure 1.

We click and drag a block for the system (2.1) from the `Linear` palette which looks like Figure 2. The block we want is the first one in the second row and is called the `Continuous state-space system` block. Note that difference equations are in the lower right and that there are Laplace transform blocks also.

There are places to make connections on both ends of this block. These are for inputs and outputs with the arrow head showing the direction. On the left we need to provide

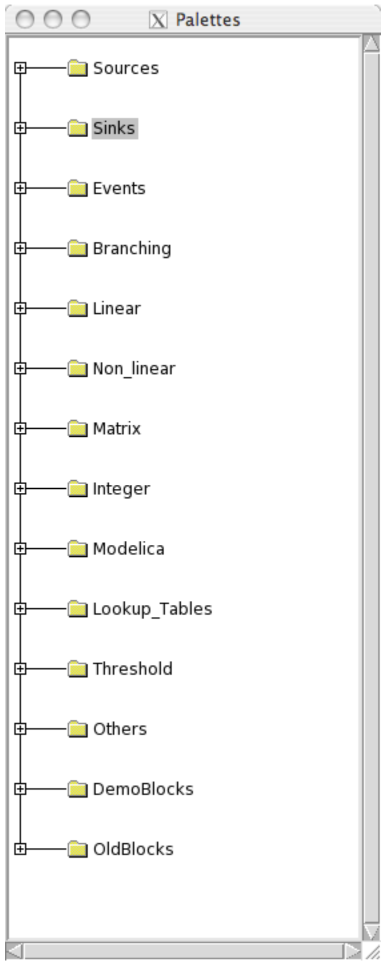


Figure 1: List of Palettes.

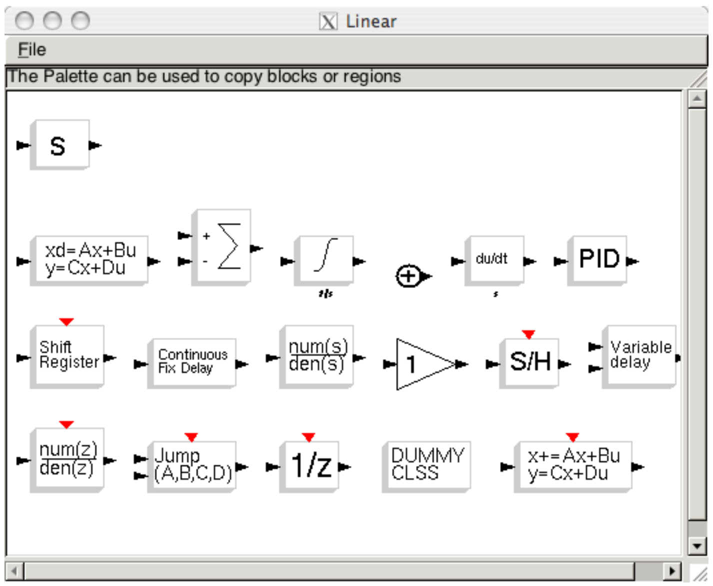


Figure 2: Linear Palette.

a block that provides u. Options will be found on the `Source` palette. For y we need to tell where the output goes. It could go to a file or another system but we will send it to a graph. This is done by going to the `Sink` palette. We select that sink which looks like a graph. This is the `Single Display Scope` block.

The simulation is completely controlled by numerical software. However, we will need to give the initial conditions, describe how often the solution is sampled for the graph, and set the axis etc. Suppose to start that we wish to simulate the following system (2.2):

$$x' = -\frac{1}{10}x + \sin t, \quad x(0) = 4 \tag{2.2a}$$
$$y = x, \tag{2.2b}$$

and graph y. The sine function is provided by the `Sinusoid generator` block.

The needed blocks are selected and moved onto the diagram as shown in Figure 3.

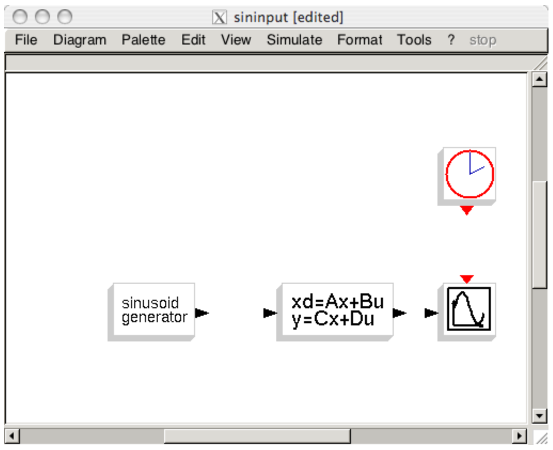


Figure 3: Blocks on diagram for simulating (2.2).

We then connect the inputs and the outputs as shown in Figure 4. This is done by clicking on one arrow head and then moving the curser to the next arrowhead and clicking again.

To specify what the blocks are we click on a block and fill in the needed information. For example we can click on the `Continuous state-space system` block and fill in what A, B, C, D and the initial condition $x(0)$ are. Scilab, or equivalently Matlab, notation is used for matrices. The `Continuous state-space system` block window is shown in Figure 5.

Clicking on the `Sinusoid generator` block enables us to change the amplitude and phase. The default is $\sin(t)$. The `Clock` block pointing into the graph block is used to set how often the solution is sampled. To run a simulation we go to the `Simulate` menu and click `Setup` and fill in the information. To run we select `Run` in the `Simulate` menu. The result is shown in Figure 6.

The student can easily experiment with different values of the coefficients and changing the sinusoidal input. The diagrams can be saved so that they can be set up by the instructor

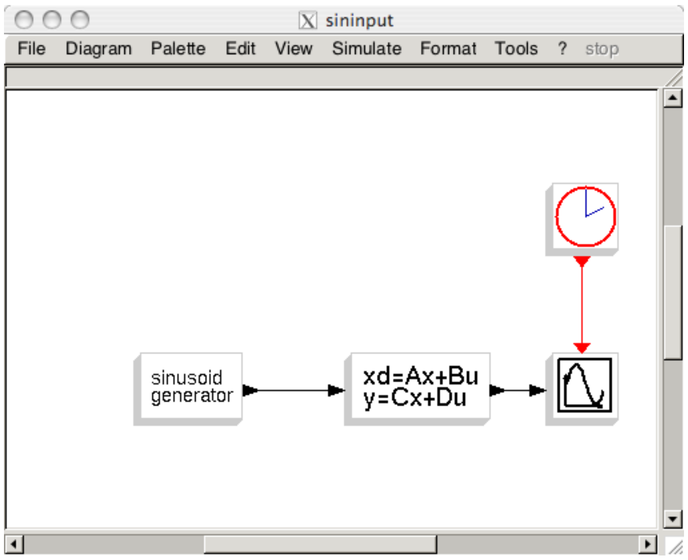
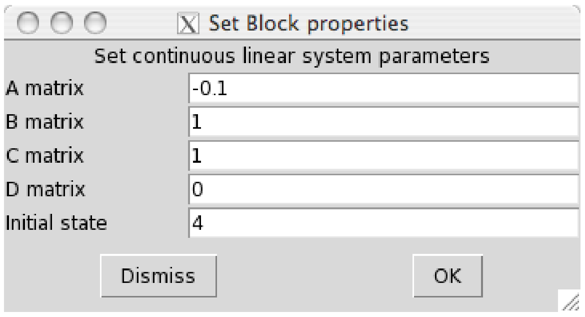


Figure 4: Final diagram for simulating (2.2).



Set Block properties

Set continuous linear system parameters

A matrix	-0.1
B matrix	1
C matrix	1
D matrix	0
Initial state	4

Figure 5: Specifying A, B, C, D for (2.2).

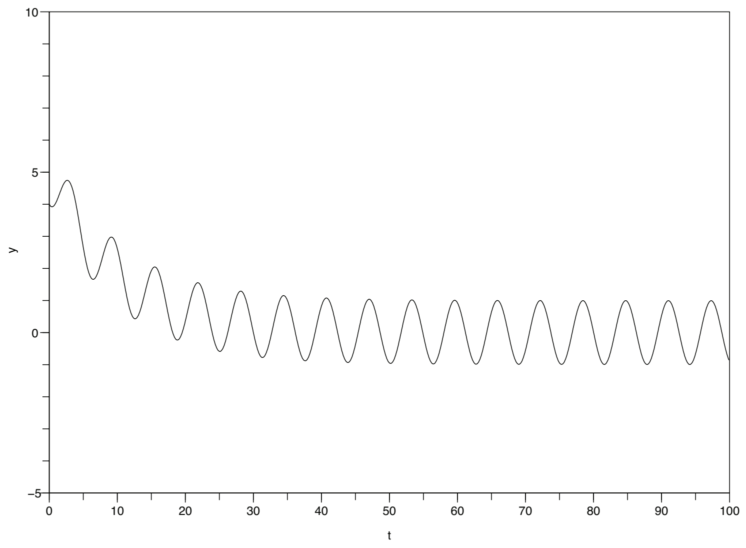


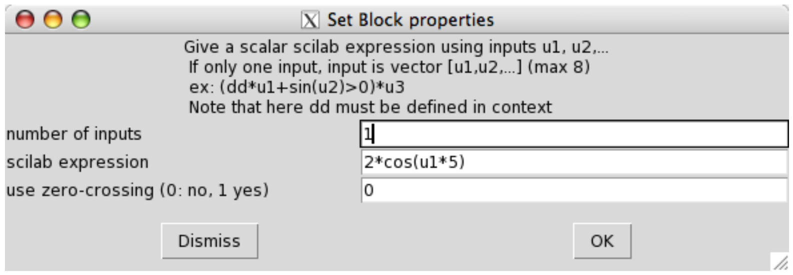
Figure 6: Solution of (2.2).

and then shared with the students or students can share with each other. Students and instructors do not need to be working on the same computer systems nor do they need to have an internet connection which is required for some Matlab site licenses.

Suppose that instead of (2.2) we want to change the input and consider instead

$$x' = -\frac{1}{10}x + f(t) \quad x(0) = 4 \tag{2.3a}$$
$$y = x, \tag{2.3b}$$

where the $f(t)$ is a user defined function. We illustrate with $f(t) = 2\cos(5t)$. This example could be done with a `Sinusoid generator` block but it suffices to illustrate the general case. We remove the `Sinusoid generator` block by selecting and deleting it and replace it with the `Mathematical expression` block. Clicking on the `Mathematical expression` block we enter the details as shown in Figure 7. The `Mathematical`


Set Block properties

Give a scalar scilab expression using inputs u1, u2,...
If only one input, input is vector [u1,u2,...] (max 8)
ex: (dd*u1+sin(u2)>0)*u3
Note that here dd must be defined in context

number of inputs	1
scilab expression	2*cos(u1*5)
use zero-crossing (0: no, 1 yes)	0

Figure 7: `Mathematical expression` block with function $2\cos(5u)$ entered.

`expression` block needs to know what the variable $u1$ is so we attach a `Time` block to it which inputs the time t. The result is the diagram in Figure 8.

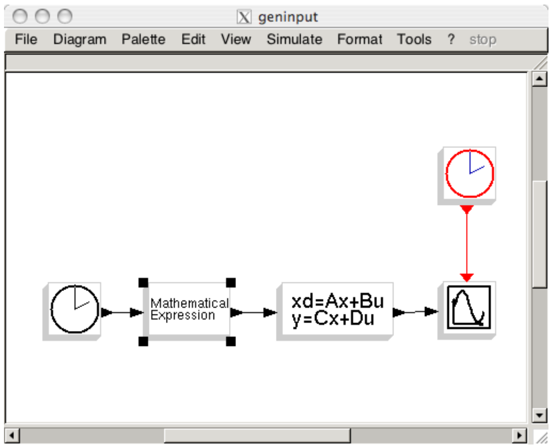


Figure 8: Diagram for (2.3) using user defined function.

When we run the simulation we get the Figure 9. Note that in Figure 9, `ymin` was set to zero in the graphical block so that the graph is clipped off below zero. Setting `ymin` and `ymax` is useful in showing smaller but interesting parts of graphs which have much larger values elsewhere. Students (even graduate students) often produce graphs with poorly chosen intervals or ranges thereby hiding interesting behavior. It is good for them to practice in making well designed graphs.

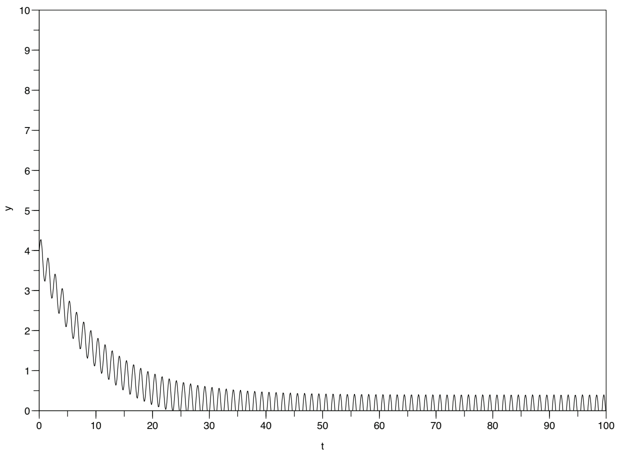


Figure 9: Simulation of Figure 8.

Given the diagrams the students can easily run a variety of simulations and experiment. They can also create more complex diagrams by hooking the output of one into the input of the next. In so doing they learn to think of a differential equation as not just a question of antidifferentiation but as an operator.

2.1 Nonlinear Systems

Scicos can also easily handle nonlinear systems such as

$$\frac{dx}{dt} = f(x). \tag{2.4}$$

This is done by diagraming the integral equation version of (2.4) which is

$$x(t) = \int_0^t f(x(t))dt + x(0). \tag{2.5}$$

This is shown in Figure 10. If the instructor wants to do so, this gives them a nice chance to discuss the important integral equation form of a differential equation which is the basis of many numerical methods. Or the instructor can just give the diagram.

In the `Mathematical expression` block we enter the function f as $f(u1)$. If we have a time dependent function $f(x,t)$ then we modify Figure 10 by having a second input to the `Mathematical expression` block which is a `Time` block and this gives us the definition of the variable $u2$.

As a particular example, we can integrate

$$\frac{dx}{dt} = x - x^2, \quad x(0) = 0.5, \tag{2.6}$$

by setting $f(u1) = u1 - u1 * u1$ in the `Mathematical expression` block and setting the initial value to 0.5 in the `Integration` block. The result of the simulation is in Figure 11.

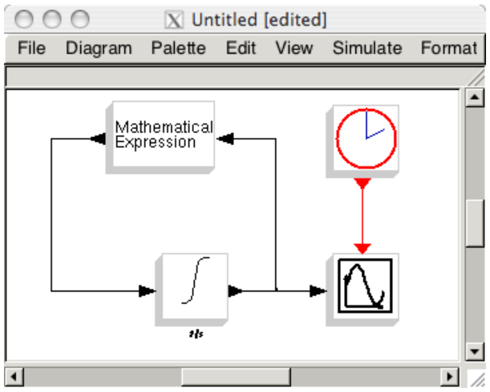


Figure 10: Diagram for (2.4) or (2.5).

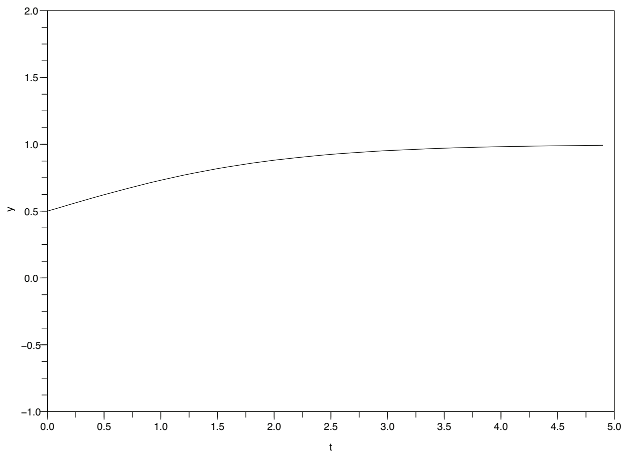


Figure 11: Simulation of (2.6) using Figure 10.

3 Simulating Directly from a Diagram

Scicos has another feature which Simulink does not have. In the proceeding section we needed to set things up in an input-output manner. While many processes are like this, not all models are. Consider an electrical circuit. It is initially modeled using Kirchhoff's laws [2]. This usually gives a number of equations which must be reduced to an ODE. The initial model, before reduction, is a type of equation called a differential algebraic equation [1] or DAE. In the last two decades considerable work has been done on differential algebraic equations and Scicos has the ability to solve some differential algebraic equations and hence some implicit models. To describe how to define the implicit blocks in the language Modelica [4] is discussed in [3]. Fortunately there are already palettes for electrical devices and for hydraulic systems. We can find the `Electrical` palette under `Modelica`. This palette is shown in Figure 12. In addition to the usual resistors, inductors, and capacitors, the `Electrical` palette includes operational amplifiers, diodes, nonlinear resistors, grounds, switches, transformers, and several other types of devices.

A simple RLC circuit with a harmonic voltage source is shown in Figure 13. In

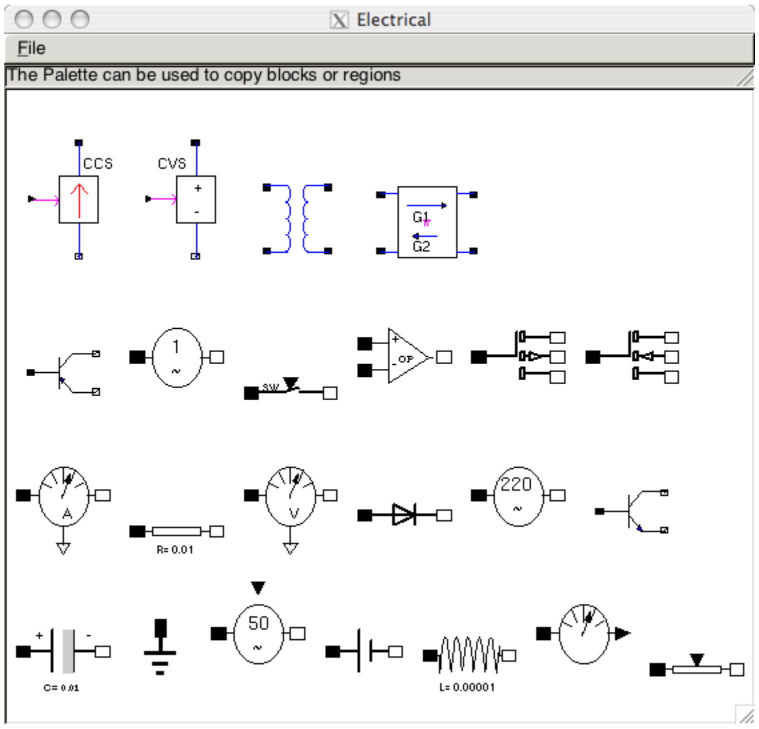


Figure 12: Electrical palette.

the diagram in addition to the `Resistor`, `Inductor`, and `Capacitor` blocks we have inserted a `Electrical voltage sensor` block which sends the value of the voltage to the graph generator. We have also inserted an `Electrical AC voltage source` block. Clicking on the voltage source allows the user to specify the amplitude and frequency. The initial conditions are determined by the `Capacitor` block since in addition to capacitance it has a specified initial voltage. Since general circuit models deal with nodal differences there needs to be some reference to uniquely determine the values. This is done by adding

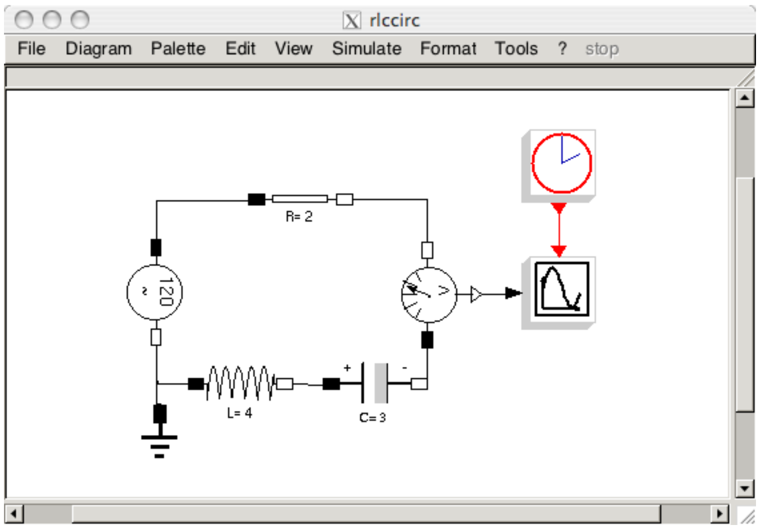


Figure 13: Circuit diagram for RLC circuit with alternating current voltage source.

the `Ground`. The simulation is in Figure 14. Note that we get the expected result for a linear circuit with this input—namely a change in phase and amplitude but not frequency.

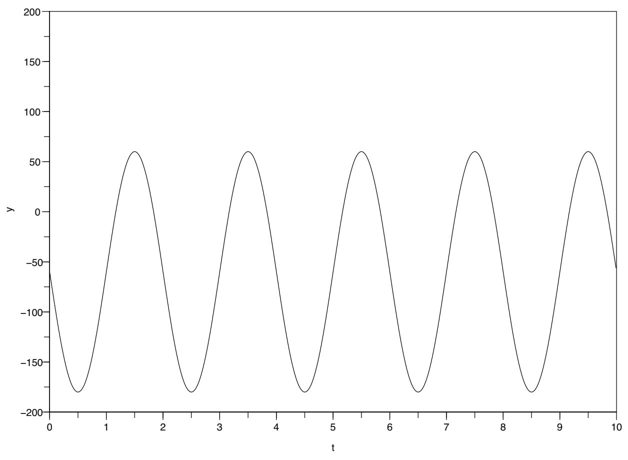


Figure 14: Simulation of circuit in Figure 13.

One of the advantages of this type of graphical interface is that it encourages the student to experiment. In fact a high school student who does not even know what a differential equation is can try out different circuit designs.

We start with our previous circuit and add more devices and loops. This more complicated circuit is given in Figure 15 with simulation in Figure 16. It is just as easy for the student as the original circuit. No reduction in variables and elimination of equations is required.

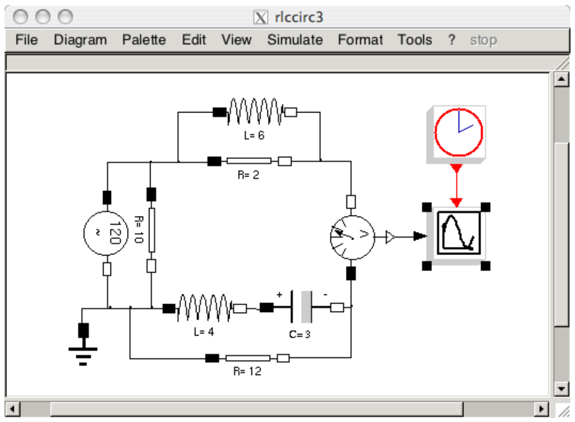


Figure 15: Larger circuit diagram.

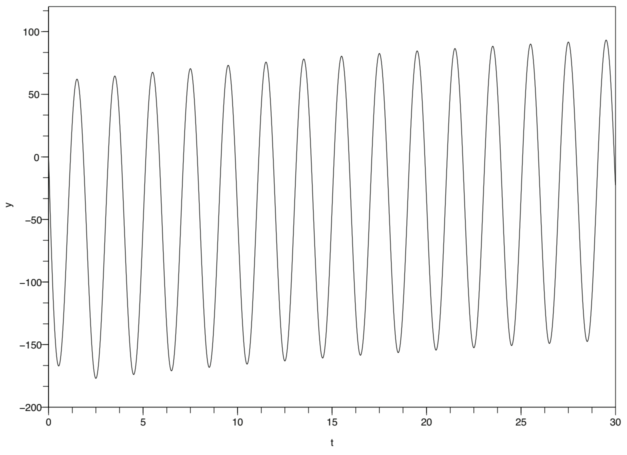


Figure 16: Simulation for Figure 15.

4 Conclusion

Using ScicosLab enables the teacher to do a number of things. First it puts a powerful software package that can be easy to use in every students hands at no cost. It encourages students to experiment and ask what if questions. It also starts students thinking of inputs and outputs rather than just calculus. It provides a simple way to discuss modeling and simulation of applications such as circuits. It provides a natural way to discuss the integral equation form of a differential equation. For students who move on to more mathematics it will continue to be useful. Like Matlab it easily handles linear algebra and many other mathematical tasks as described in [3].

References

[1] Kathryn E. Brenan, Stephen L. Campbell, and Linda R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations.* Classics in Applied Mathematics. Society for Industrial Mathematics, second edition, 1995.

[2] Stephen L. Campbell and Richard R. Haberman. *Introduction to Differential Equations with Dynamical Systems.* Princeton University Press, 2008.

[3] Stephen L. Campbell, Jean-Philippe Chancelier, and Ramine Nikoukhah. *Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4.* Springer, second edition, 2009.

[4] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.* Wiley-IEEE Press, 2004.