Procedia Computer Science

Volume 51, 2015, Pages 1967–1976

ICCS 2015 International Conference On Computational Science

An Introductory Course in the Computational Modeling of Nature

Kathie A. Yerion[1]

[1]Gonzaga University, Spokane WA U.S.A.

yerion@gonzaga.edu

Abstract

This introductory course in computational modeling of nature contains the development of three kinds of models of phenomena in nature -- agent-based models and simple finite difference models using the environment of the NetLogo language and complex finite difference models using the language of C++. No prior programming experience is assumed. The natural phenomena modeled include some standard ones (e.g. ants following pheromone trails, the interaction of sheep and wolves) and some non-standard ones (the creation of the world, 3 dogs playing games, and formation of stripes and spots in the skins of animals). The emphasis of the course is on the modeling process. A distinguishing feature is that students are able to compare and critique these models.

Keywords: modeling, agent-based, system-dynamics

1 Introduction

doi:10.1016/j.procs.2015.05.461

The background for the development of the course was the involvement of faculty at a set of colleges in the northwest of the United States in an NSF grant (CNS-0829651) for “computational thinking across disciplines” during the years 2008-2011. During that time, the author completed a finite difference approximation for a system of partial differential equations modeling the formation of stripes and spots in animal skins (Bassiri, 2009), attended a workshop on computational science by the National Institute of Computational Science, and developed a teaching module for a model of formation of stripes and spots in animal skins (Yerion K. , 2012). This introductory course in computational modeling of nature was first offered in 2012 and has undergone several revisions in each year since. Although the course was developed for students in the liberal arts, it is an elective for computer science students. Currently, the mixture in the course is eighty percent computer science students and twenty percent liberal arts or business majors. However, it is taught assuming no prior programming experience. Although it also has no mathematics pre-requisite, students have always had a calculus background. This allows the course to include differential equations and their finite difference approximations. For students without this background, the instructor would need to spend additional time teaching the students about recurrence relations, the concept of a derivative, and how they relate to each other.

In the first offering of the course, the author received significant help from Matthew Dickerson of Middlebury College who teaches an agent-based modeling course for environmental studies using the NetLogo language. In addition, NetLogo contains a Models Library with many examples. As the course evolved, new agent-based models were added.However, since the emphasis of this course is modeling, it was important to include multiple types of modeling techniques. Having learned system dynamics modeling while attending a computational science workshop, the author decided to add this modeling technique to the course. System dynamics modeling allows almost direct translation of first order difference equations with initial conditions through click and drag of items to a NetLogo window. In the third edition of the course, the model of the formation of animal patterns was added. Currently, seven weeks of the course are spent on agent-based models using NetLogo, five weeks on system dynamics models using NetLogo, and two weeks on finite difference techniques using C++. As their experience with different types of models increases, students are able to analyze the advantages and disadvantages of each type of model.

2 Modeling types

There are numerous types of discrete models: cellular automata, agent-based models, system-dynamics models of first order difference equations, more complicated finite difference models, and finite elements. As an introductory course with limited pre-requisites, this course concentrates on models accessible to beginning students: agent-based modeling, system dynamics modeling, and the experience of using a more complicated computational algorithm. In each case, the modeling process is emphasized: start with a phenomenon in nature, approximate the phenomenon with a model, use the model to answer questions about the original phenomenon and critique the answers obtained from the model. At the end of the course, the critique of the type of model is added. At least two sufficiently complex phenomena – interaction of sheep and wolves and the spread of malaria – are modeled by both agent-based and system dynamics models so that an analysis of their advantages and disadvantages can be done. Also, the underlying approximation technique of a system dynamics model is discovered and better approximation techniques are presented and used. Finally, students use an interactive C++ program that contains a finite difference approximation that models the formation of stripes and spots in animal skins. Depending on the user's entered parameters, different patterns (e.g., different patterns of stripes or patterns for spots) are obtained.

2.1 Agent-based models

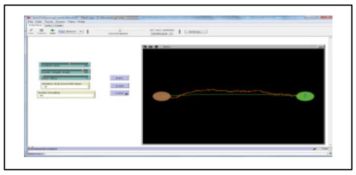
Agent-based models approximate phenomena by interacting agents. In the NetLogo language (http://ccl.northwestern.edu/netlogo/), patches are agents that can be many different colors and sizes but do not move and turtles are agents that can be many different shapes, colors, and sizes but move. One of the first non-simple models of the course is that of the author's three dogs searching for hidden dog treats in a large back yard, eating cherries from branches of trees, and also distinguishing sweet grass from weeds. There are different behaviors of each dog that must be modeled. As an item is eaten, it must disappear. The dogs are "turtles" of different colors and sizes and the cherries, treats, and grass are "patches" of different colors. In most agent-based models, the movement of a moving agent has some randomness attached. A section of code (Gravelle, 2014) for two procedures for the dogs to move and smell the cherries is in Figure 1.The user interface shows cherries, treats, grass and weeds.

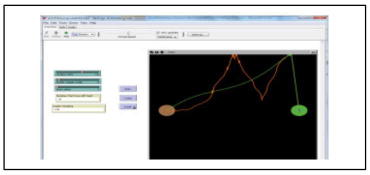
```
to move-dogs
  right (random 41 – 20)    forward 1  end
to smell-cherry-trees
  let nearby-trees patches in-radius 7 with  [ pcolor = red ]
  if count nearby-trees > 0 [
    let close-tree min-one-of nearby-trees [ distance myself ]
    setxy [ pxcor ] of close-tree [ pycor ] of close-tree
    eat-cherries    ] end
```

Figure 1

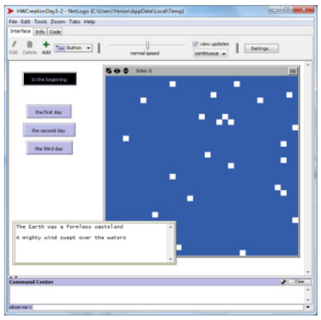
The first model that allows for interesting experimentation is one of ants following a pheromone trail of a leader as the ants leave their nest in search of food (Wilensky, 1997). The angle amount that they may wiggle or turn before each motion can vary with a user-controlled slider "leader-wiggle-angle" on the user interface. The rest of the ants follow the trail of the leader. When the students use the slider to find the wiggle angle of the leader that allows for most direct path to the food, they are surprised with the result. It would seem that a smaller wiggle angle would lead more directly to the food. Instead through experimentation, they find it is a larger wiggle angle that leads more directly to the food. In Figures 2, the path of leader ant is orange while the path of last ant is green. The left graph is when the leader wiggle angle is large.

Figures 2





An interesting phenomenon to model is the seven day creation of the world according to Genesis (The New American Bible, 1971). It allows for multiple ways to model time and encourages the creative animation of each day's events. It is not possible to show animation here, but the white squares on the left of Figures 3 move to simulate "a mighty wind swept over the waters" making white caps with the associated code on the right.



```
The Earth was a formless wasteland
A mighty wind swept over the waters
```

```
to create-wind-and-waters
   ask patches [ set pcolor blue ]
   let time 0
   while [ time < 10000 ]  [
      ifelse time mod 2 = 0 [
        ask n-of 50 patches
                    [ set pcolor white ]
     ] [
        ask patches [ set pcolor blue ]
      ]
  set time time + 1
] end
```

Figures 3

The seven days of creation continue with the creation of dry land, plants, fruit trees, swimming creatures, birds, cattle, and humans. Each patch must be placed appropriately and each turtle must move carefully.

This section ends with a model of erosion (Dickerson, 2012). In this model of raindrops running down the side of a mountain, students discover the limitations of the graphics to show this erosion well.

The course is taught in a lab environment where the students follow steps with the instructor in developing models. Then in assignments, students are assigned steps to complete on their own. For example, in the model of creation of the world, the first three days are modeled in labs while the students model days four through seven.

2.2 System dynamics models

Each system dynamics model is essentially a computational translation of first order finite difference equations or first order differential equations. A fantastic resource for phenomena and their associated models, perfect for system dynamics, is the textbook by Shiflet and Shiflet (Shiflet, 2006). A warm-up example is a model of money in a savings account at 3% interest with initial deposit of $1000. NetLogo has a built-in systems dynamics modeler with drag and drop options. Mathematically, the model of amount of money after n years, A_n, is $A_{n+1} = A_n + 0.03A_n$, $A_0=1000$. The systems dynamics model in NetLogo looks like Figure 4.

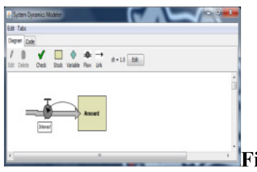
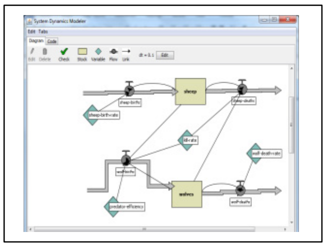


Figure 4

There is a Stock for the Amount with initial value 1000, a Flow of Interest with the expression 0.03 * Amount, and a Link from Amount to Interest as drag and drop items. NetLogo code is automatically generated. The next model consists of the interaction of sheep and wolves (Wilensky, 2005) to produce the diagram in Figure 5 of Stocks sheep and wolves, Variables sheep-birth-rate, wolves' kill-rate of sheep, predator-efficiency, wolf-death-rate, and Flows sheep-births, sheepdeaths, wolf-births, wolf-deaths and associated links. The associated differential equations are in the form $\frac{dS}{dt} = bS - pS \cdot W$, $\frac{dW}{dt} = r \cdot S \cdot W - dW$, the famous Lotka-Volterra system of nonlinear differential equations with graph in Figure 6 (Wikipedia, 2015).

Figure 5



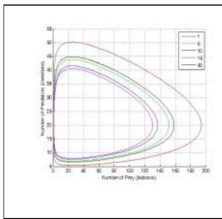
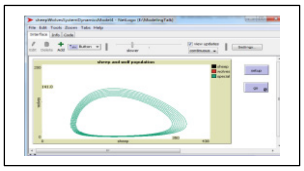


Figure 6

When the graphing option is used on the user interface for sheep vs. wolves, similar curves appear as in Figure 7. The fact that students with limited background produced the curves similar to those from an advanced topic like nonlinear differential equations is quite amazing!

Figure 7



We spend quite some time learning to interpret these graphs. We also compare and analyze the reason for the smoothness of the curves in the systems dynamics modeler compared with more jagged curves from a corresponding agent-based model of the same phenomenon as shown in Figure 8.

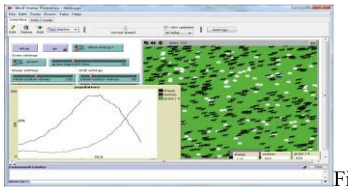


Figure 8

The last major systems-dynamics model is the model of malaria (Shiflet, 2006). We spend considerable time developing the model together in class and answering questions from it. As shown below in Figure 9, it is the most complex model of the system-dynamics models of the course.

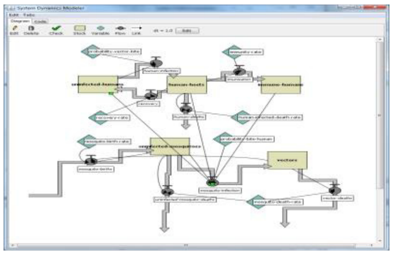
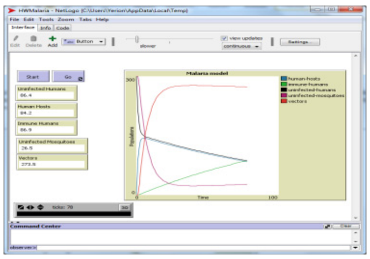


Figure 9

At any moment in time, the value of each of the five populations can be read from monitors and from the graphs as shown in Figure 10. The students build a table of time vs. populations. They notice the population values contain fractional parts and must analyze why this is so.



Uninfected Humans: 86.4
Human Hosts: 84.2
Immune Humans: 86.9
Uninfected Mosquitoes: 26.5
Vectors: 273.5

Figure 10

In order to compare, we build an agent-based model of this same phenomenon. As is common with moving agents, their movement and interaction contains randomness. As a result, when the students build a similar table of time vs. populations, they find significant differences from the previously-made table from system dynamics. They also see only integer values for populations and more jagged graphs. Advantages and disadvantages are discussed.

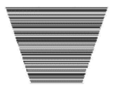
2.3 Differential equations and finite differences

The course switches to a short introduction to three common approximation techniques of first order differential equations with initial conditions. For example, for $y' = y, y(0) = 1$ the goal is to find $y(1)$ (solution is $y(t) = e^t$). The techniques of Euler's Method, Modified Euler's Method, and Runge-Kutta method (Gerald, 1999) are used to approximate. To help students understand these methods, they are presented using geometry, slope, and equations of lines. For the above example, the "slope at (t, y)"$is\ \ y.$ In each case, the method approximates the solution at t = 1 by a line through the point (0,1) .

	Slope	Averaged At t =	approximation	error
Euler's	1	0	2	0.718
Modified Euler's	1.5	0 and 1	2.5	0.218
Runge-Kutta	1.7	0 and 0.5 and 1	2.7	0.018

The winner is clear. The course returns to the model in Figure 4.Using the system dynamics model, students discover that it uses Euler's Method, the least accurate. Thus, if more accurate results are required, the method of Runge-Kutta should be used.

Finally, the model of the formation of animal patternsl (Bassiri, 2009) is considered. The model is a finite difference approximation of nonlinear partial differential equations which students of limited calculus background will not understand. Thus, we concentrate on modeling a leg or tail by a tapered cylinder which is then cut to form a trapezoidal region. The students are given a program in C++ which implements the algorithm. Looking at the loops, they find that one time step is about 10,000 multiplications. In order for patterns to emerge, a steady-state must occur after 10,000 time steps. They are warned that each execution of the program will take about ten minutes. Depending on the parameters entered, different patterns will emerge. They are asked to determine possible animals that match the patterns.

Figures 11


3 Conclusion

The students will say that agent-based modeling is “more fun” than the other techniques because of the sheep and wolves or ants running around a screen. At the end of this course, they have a breadth of modeling experiences, and they learn not just to develop and work with models but to critique them. One area for the future is to assess student learning about modeling through pre- and post- questions. At present, the author only has comments: “This course helped me appreciate the usefulness of computational models, and how applicable … modeling can be to non computer science fields of study,” and “It was a lot more math heavy then I expected. It started to show where mathematics and computer science really come together.”

References

Bassiri, E. B. (2009). A finite difference method for modeling the formation of animal coat patterns. *Nonlinear Analysis Real World Applcations*, 1730-1737.

Dickerson, M. (2012). Personal Communication. erodeifstos.nlogo.

Gerald, C. W. (1999). *Applied Numerical Analysis.* Reading: Addison Wesley Longman.

Gravelle, B. P. (2014, February). Personal Communication doggies3.nlogo.

Shiflet, A. S. (2006). *Introduction to Computational Science.* Princeton University Press.

The New American Bible. (1971). Catholic Publishers.

Wikipedia. (2015). Retrieved from http://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equation.

Wilensky, U. (1997). Ant Lines in NetLogo Models Library.

Wilensky, U. (2005). Wolf Sheep Predation from NetLogo M odels Library.

Yerion, K. (2012). Alan Turing, Animal Spots, and Algorithms. *The Journal of Computing Sciences*, 169-176.