2.3: Modeling Complex Systems

The challenge in developing a model becomes particularly tough when it comes to the modeling of complex systems, because their unique properties (networks, nonlinearity, emergence, self-organization, etc.) are not what we are familiar with. We usually think about things on a single scale in a step-by-step, linear chain of reasoning, in which causes and effects are clearly distinguished and discussed sequentially. But this approach is not suitable for understanding complex systems where a massive amount of components are interacting with each other interdependently to generate patterns over a broad range of scales. Therefore, the behavior of complex systems often appears to contradict our everyday experiences.

As illustrated in the examples above, it is extremely difficult for us to come up with a reasonable model when we are facing something unfamiliar. And it is even more difficult to come up with a reasonable set of microscopic rules that could explain the observed macroscopic properties of a system. Most of us are simply not experienced enough to make logical connections between things at multiple different scales.

How can we improve our abilities to model complex systems? The answer might be as simple as this: We need to become experienced and familiar with various dynamics of complex systems to become a good modeler of them. How can we become experienced? This is a tricky question, but thanks to the availability of the computers around us, computational modeling and simulation is becoming a reasonable, practical method for this purpose. You can construct your own model with full details of microscopic rules coded into your computer, and then let it actually show the macroscopic behavior arising from those rules. Such computational modeling and simulation is a very powerful tool that allows you to gain interactive, intuitive (simulated) experiences of various possible dynamics that help you make mental connections between micro- and macroscopic scales. I would say there are virtually no better tools available for studying the dynamics of complex systems in general.

There are a number of pre-built tools available for complex systems modeling and simulation, including NetLogo [13], Repast [14], MASON [15], Golly [16], and so on. You could also build your own model by using general-purpose computer
programming languages, including C, C++, Java, Python, R, Mathematica, MATLAB, etc. In this textbook, we choose Python as our modeling tool, specifically Python 2.7, and use PyCX [17] to build interactive dynamic simulation models. Python is free and widely used in scientific computing as well as in the information technology industries. More details of the rationale for this choice can be found in [17].

When you create a model of a complex system, you typically need to think about the following:

1. *What are the key questions you want to address?*

2. *To answer those key questions, at what scale should you describe the behaviors of the system’s components?* These components will be the “microscopic” components of the system, and you will define dynamical rules for their behaviors.

3. *How is the system structured?* This includes what those microscopic components are, and how they will be interacting with each other.

4. *What are the possible states of the system?* This means describing what kind of dynamical states each component can take.

5. *How does the state of the system change over time?* This includes defining the dynamical rules by which the components’ states will change over time via their mutual interaction, as well as defining how the interactions among the components will change over time.

Figuring out the “right” choices for these questions is by no means a trivial task. You will likely need to loop through these questions several times until your model successfully produces behaviors that mimic key aspects of the system you are trying to model. We will practice many examples of these steps throughout this textbook.

Exercise \( \PageIndex{1} \)

Create a schematic model of some real-world system of your choice that is made of many interacting components. Which scale do you choose to describe the microscopic components? What are those components? What states can they take? How are those components connected? How do their states change over time? After answering all of these questions, make a mental prediction about what kind of macroscopic behaviors would arise if you ran a computational simulation of your model.

3 For those who are new to Python programming, see Python’s online tutorial at docs.python.org/2/tutorial/index.html. Several pre-packaged Python distributions are available for free, such as Anaconda (available from continuum.io/downloads) and Enthought Canopy (available from enthought.com/products/canopy/). A recommended environment is Anaconda’s Python code editor named “Spyder.”