Mathematical Modeling in the Undergraduate Curriculum

Abstract: Mathematical modeling occupies an unusual space in the undergraduate mathematics curriculum: typically an “advanced” course, it nonetheless has little to do with formal proof, the usual hallmark of advanced mathematics. Mathematics departments are thus forced to decide what role they want the modeling course to play, both as a component of the mathematics major and as a course with potential cross-disciplinary appeal.

This article takes the position that a modeling course de-emphasizing mathematical formalism in favor of scientific computing, data analysis, and communication skills is both an appropriate upper level mathematics course and an effective means of training real-world problem solvers. Designing a course that serves a broad range of students while remaining practically relevant and mathematically sophisticated can be a challenge, however. Here the author distills key features of his recent course with a group of 20 undergraduates from widely different majors. Comments about course outcomes are contextualized by a broader discussion about real-world modeling practices. The piece concludes with a list of concrete ideas about target competencies and ways to achieve them.

Keywords: modeling, education, computation, data analysis

1 INTRODUCTION

In Spring 2011 I taught a course in mathematical modeling to a group of 20 undergraduates at the University of Puget Sound, a small, primarily undergraduate residential liberal arts college in the Pacific Northwest. Having worked in both pure and applied mathematics, as well as in industry, I had a reasonable sense of the various guises the activity of mathematical modeling could assume, but as my students represented many different majors, it was not obvious to me which elements of this vast activity should form the backbone of my course. Should we be doing proofs or numerical simulations? Should we focus on one subject area or many? Which underlying mathematical techniques were “fundamental” and which could be cut in the inevitable sacrifices to time?

Although I tentatively took positions on these issues and forged a syllabus accordingly, my thoughts evolved as the course progressed, both in response to student feedback and to my own emerging sense of what was interesting, accessible, and useful across divergent disciplinary backgrounds. Although I started the course believing that my goal was to focus on the mathematical end of mathematical modeling, I gradually began to see that the course could (and should) extend along many other important axes, including such non-mathematical ones as learning how to do literature searches, read technical articles, do rapid prototyping on a computer, work in groups, write up results, and give presentations. I eventually managed to squeeze these activities into the course, but the realization that they might profitably have played a more prominent role led me to re-conceptualize this course as something that can fill an important cultural void: it offers not only a rare link between the pure and the applied, but also highlights the essential unity in the process of discovery, both scientific and mathematical.

The purpose of this paper is to use my experiences in the classroom as the point of departure for a broader discussion about the possible roles of a modeling class, both as a core offering within the mathematics curriculum and as a course of core interest to students from a range of fields. In its guise as the former, I argue that even though most upper division undergraduate mathematics courses seek to develop proficiency in producing and analyzing proofs, that the modeling course can profitably deviate from this paradigm and still form an essential component of undergraduate mathematical training. Moreover, I argue that it ought to deviate from this paradigm to the extent that it aspires to serve as a course of central interest for students from many disciplines. While recognizing that the particular form a modeling class takes will vary by institution and instructor, I suggest that by deliberately courting non-mathematics majors, the modeling course opens up a variety of doors for inter-departmental dialogue, and that the dividends of this dialogue can be far reaching.
In order to set the stage for this discussion, I devote the next section to describing my particular experience teaching this course. Although to some extent the challenges I faced were an artifact of the setting in which I was teaching, teachers in other settings will probably recognize common elements. In the following section I contextualize these experiences by discussing how the real-life practice of modeling plays itself out along the three axes of mathematics, data analysis, and computation. This section is broad, bordering on philosophical, and draws heavily from my own experiences in the field. The fourth section argues that a modeling class whose energies are evenly dispersed along these three axes is a good way to accommodate the welter of legitimate but conflicting claims for such a course, and can benefit students, faculty, and institutions alike. The fifth section takes these objectives as its starting point and focuses on concrete ways to achieve them. While it steers clear of prescribing what a modeling course should look like, it does propose a number of key competencies and offers specific ideas for how they can be developed.

2 PORTRAIT OF A MODELING CLASS

At the University of Puget Sound mathematical modeling is a 400 level course, meaning it is for junior/senior level undergraduates who have a reasonable level of mathematical maturity. Prerequisites are minimal, however: the only required classes are vector calculus and linear algebra, although a course on probability is “recommended.” The course is cross listed as a 400 level computer science class, but there are no computer science prerequisites.

Historically, the course has been offered once every two years and has mostly catered to mathematics majors. Class sizes at the University of Puget Sound are small (typically less than 25 students) but even by these standards the modeling class has been tiny: enrollments during the last two iterations of the class have been three and five, respectively. Perhaps in response to growing recognition of the centrality of the field, enrollment rose radically in the year I taught the course. Of my 20 students, only one was an “un-hyphenated” mathematics majors; the remainder either were majoring in something else, or were incorporating mathematics as one component of a double or dual major. Focus areas included computer science, physics, geology, economics, business, and chemistry. All of the students had completed the calculus sequence (up through vector calculus) and all but one had taken linear algebra, but only about half had taken differential equations, few had formal programming experience, and only a handful had taken probability or statistics.

2.1 Implementation Details

Although there were no biology majors in my course, I chose to use Leah Edelstein-Keset’s book Mathematical Models in Biology as our basic text [6]. My rational for using a subject-specific book (as opposed to a more general text on modeling such as [9] or [7]) was twofold: one, I felt that homogenizing the applications would help highlight the underlying mathematics, and two, this particular subject happened to align with my own research interests. Though the text had no biological prerequisites and covered what I considered a nice range of modeling techniques, the simple fact of using a biology book had both positive and negative aspects: on the one hand, the content was largely immaterial to the disciplinary interests of the students, but on the other, it provided a suite of related examples and ended up provoking a lot of interest in the field. We covered various facets of difference and differential equations.

One of the features of this specific text is that it is not particularly computer oriented: although it does contain exercises for students to complete using a computer, there is no guidance on language syntax and no example code. Given the dominant role that the computer plays in both science and industry, I felt that I had to incorporate a computational dimension to the course, but I did not want to spend a lot of time teaching programming skills, since programming per se was not the focus of the course. Since low-level languages like C++ or Java have a high learning overhead, adopting them for the course was not feasible, but higher level languages like Matlab, Maple, or Mathematica are costly, which meant that most students could only access them in on-campus labs. I ended up striking what I felt was a reasonable compromise by allowing my students to do their work in whatever language they wanted, with the caveat that they were responsible for teaching themselves how to use it. I facilitated this process by providing a graded sequence of low-level exercises (simple loops, control structures, graphing exercises, Monte Carlo simulations), each of which was easy enough to look up in a help manual but which together form a powerful set of tools. I
complemented this set of assignments by providing example code in the Matlab programming language, a syntax that can be run verbatim in Octave, an open-source Matlab clone. I also helped students install Octave on their laptops (there are distributions for Mac, Windows, and Linux) and directed them to an on-line tutorial that covered the language elements I expected them to master.

Having chosen a text and coding policy, I began to think about how to structure assignments. I wanted to foster not just individual competence, but also the capacity to work effectively in groups, a goal designed to reflect the workplace realities in many areas of both science and industry. Accordingly, each time I passed out a homework assignment I randomly partitioned the class into five groups of four students. The randomization was carried out live in the classroom using a random number generator on my computer, and each group was instructed to turn in a single assignment with all four names attached. Since everyone within a group got the same grade for any given assignment, students were motivated not just to do their best work but also to devise a way to get their colleagues to do the same. One of the ideas behind this system was that although random fluctuations in group composition might effect the quality of the work, a student’s tendency to pull his or her group’s grade up or down would become visible when averaged over many assignments. I assigned six projects over the course of the semester, and although I graded them without first checking authorship, the final distribution of project grades largely confirmed this idea.

Somewhat controversially, I chose to insist that project reports be produced in the typesetting language Latex. Although Latex has historically been the purview of mathematicians, its popularity is growing in other fields and it is widely viewed as a good choice for a wide variety of communications, both technical and otherwise. Since the outcome of the modeling activity is often qualitative and discussion-oriented, I felt that emphasizing scientific writing was a reasonable course objective, and that a working knowledge of Latex might be a valuable job skill in whatever sphere these students ended up working in. I found that a one-hour template based crash course was enough to get them going, and although technical questions inevitably surfaced later on, in general the writing process went smoothly. Consistent with the theme of cultivating a solid scientific writing style, I graded the projects not just on mathematical and scientific grounds, but also on grammar, spelling, and diction.

Finally, in order to make sure sure that no student simply leveraged his or her group’s strengths while failing to master elementary skills, I had one mid-term exam and one final project. The mid-term exam was a take-home project that covered select topics regarding the analysis and simulation of systems of difference and differential equations. I required that the exam be submitted in Latex. The final project was a major piece of work (a 10-20 page report) on a topic of the student’s choosing. Most students chose to pursue a topic that lay in their primary field of interest. Students were required to present their results in a 15-minute presentation.

2.2 Notable successes, notable failures

Certain class elements worked better than others. The group homework structure, for example, proved highly effective, at least to judge by student feedback: most students appreciated the opportunity for truly collaborative work, and felt that the opportunity to share ideas and allocate tasks like writing and coding was unique in their mathematics experience. A benefit of this approach was that it closely mirrored the working structure of the Mathematical Competition in Modeling (MCM), a national and highly visible competition for undergraduates in which our students routinely participate. Although I did not require my students to participate in the MCM, those who did reported that they felt well prepared, and all ended up placing at least in the top 20%. I also felt that this sort of teamwork was good job training, as it is similar to the kind of collaborative work one might find in a technical job as a research analyst. The weakness of this approach is that individual contributions can get blurred. My solution of interspersing these group assignments with focused individual tasks (a mid-term, a final project) gave me some powers of discernment, but another option would be to assign one student per group to a “leadership” role, ensuring that each student was tapped for such a role at least once. This approach would ensure that each student got at least a taste of “management”, another desirable job skill.

My choice of a discipline-specific text was not contentious, exactly, but some students did express a desire to see examples more closely related to their own fields. By presenting the material in the setting of a subject I know something about (mathematical biology), I hoped to leave the students in a good position to translate the material into the language of their own subjects for their final project. Moreover, as the
course evolved I felt increasingly strongly that it was less about any particular mathematical technique than the delicate art of figuring out what some particular community considered relevant to the problem at hand. As a consequence, at some point mid-semester, we abandoned the book altogether and focused on reading (judiciously chosen) research articles, with the idea of gaining exposure to a wide range of authorial and disciplinary voices. This shift was challenging, but students learned to extract content from technical material slightly beyond the scope of their understanding. This skill came in handy for their final projects, which were generally small variants of results students found in technical journals.

Computer-driven analysis ended up playing a significant role in the course, and to the extent that this mirrors current practice in many areas of science and industry, I think that the computational emphasis was legitimate and fruitful. Many students reported that this course provided their first real exposure to “scientific computing”, i.e. the art of using the computer to solve, analyze, or dissect problems from the physical world. In my opinion, the capacity to use a computer in this way is one of the most useful and marketable skills an undergraduate can acquire, and accordingly I was happy to use the course as a window into that world (the University of Puget Sound does not at present offer a separate scientific computing course.) On the other hand, allowing everyone to use a different programming language was a bad idea. Although it did save some students the time needed to learn a new language, it made group work difficult. It also undermined the usefulness of my example code, since it was readily intelligible only to students who knew the language in which it was written (Matlab, as it happened; other languages in play included Mathematica, Sage, and Java.) In the future, I will definitely choose a single language as the class standard, and insist that all students use it.

Lastly, the focus on polished write-up was highly rewarding. By generating electronic versions of their homework, groups were able to collaborate and distribute their work easily. Students mastered the basic elements of Latex very quickly, and although it took them some time to realize that I was serious about grading for spelling, syntax, and grammar, they eventually learned to create elegant and aesthetically pleasing technical documents, a skill that will probably serve them well regardless of their professional trajectories. The final presentations proved a nice opportunity to drill “technical conversation” skills: I asked that students make a point of asking questions after presentations, incentivizing this involvement with a “participation” grade. By seeing many talks, students learned to discern an appropriate level of detail for technical explanations, and by being required to participate in the dialogue, students learned to extract significant content from challenging and unfamiliar material.

3 THREE AXES OF MATHEMATICAL MODELING

As a mathematician teaching in a mathematics department, I felt that the ostensible purpose of my modeling class was to teach “mathematics.” As the above description of the class makes clear, however, students in my class ended up learning many things that were only tangentially related to conventional math: science, computer coding, typesetting, formal group work, how to give a technical talk, how to read a research paper. Were these additional factors merely the “overhead” needed to get to the core of the discipline, or were they in some sense essential, i.e. as valuable in their own right as the underlying mathematics?

A sensible way to start thinking about what is “essential” to this hybrid discipline is to consider how modeling is actually practiced in the real world. Cataloguing modeling’s various disciplinary guises is difficult, however, precisely because of the phenomena Eugene Wigner dubbed the “unreasonable effectiveness of mathematics” [10]: the list of disciplines that make use of mathematical models is very large. Indeed, to the extent that a “model” is just a mathematical formalism used to describe, explain, or predict phenomena, and a modeler anyone who traffics in such models, either by creating them, analyzing them, or using them for some specific end, most academic and commercial enterprises can legitimately claim to be involved in some aspect of mathematical modeling.

In the face of such disciplinary diversity, it is useful to focus on how modeling practices differ along what might be considered three principal axes of the field: data, mathematics, and computation. These axes are not orthogonal, but they represent major centers of activity familiar to most modelers. Although in some sense modeling is the artful interplay of these threads, different modelers favor some over others, and the flavor of their work varies accordingly. Understanding the scope of “modeling” as a cultural practice can thus be seen as an exercise in grasping the range of activities along each of these three axes.
3.1 Mathematics

The mathematical dimension of mathematical modeling has two components: model creation and model analysis. Model creation is the process of writing down formal relations to characterize phenomena. While this task requires mathematical maturity, it is not mathematics in the classic sense, for it does not (and cannot) involve formal, logical proof. Model analysis, on the other hand, can lend itself to the sort of theorem-proof work of conventional mathematics.

Depending on the sort of work a modeler is doing, she may be involved in one, both, or neither of these elements. An ecologist, for example, might endeavor to write down a difference equation that describes observed population growth of a certain species, but may very well leave the analysis of that equation to a mathematical collaborator or a computer simulation. A mathematical collaborator may by wholly ignorant of the physical intuition behind some model, but nonetheless succeed in proving closed form results about asymptotic stability. A graduate student with access to data may use statistical techniques to estimate some parameter for a model suggested by a thesis advisor. All three people are involved in mathematical modeling, and all three endeavors can result in high quality work, but what is “mathematical” about their work varies hugely.

To some extent, this range in mathematical engagement can be understood as the difference between theorists and experimentalists, but this distinction tends to oversimplify the situation, as the vast majority of modelers need to be both. To consider a concrete example, medical image processing is a heavily mathematical field that draws heavily from harmonic analysis and approximation theory. Modelers interested in theoretical results regarding the invertibility of data acquired under some particular sensing regime essentially work as mathematicians, i.e. their work is about theorems and proofs. But to the extent that their work is modeling (and not pure mathematics) it needs to acknowledge the primacy of empirical observation, which is to say that the theory needs to be in demonstrable support of something practical. On the other hand, there are many modelers working in more or less the same area whose daily work involves tinkering with algorithms on their computer. Algorithms that show promise on test data may attract interest even without theoretical guarantees, but saying something theoretical can shore up a piece of work and make a significant difference in attracting interest. Good experimental work relies on theory in exactly the same way that good theory relies on experiments.

3.2 Data

To the extent that modeling attempts to grapple with the world of “phenomena”, it necessarily incorporate a data component. The nature of what that data might be can vary radically, however. For radar engineers, it might be a sequence of voltage trains; for psychologists, it might be patterns of luminescence within fMRI data; for ecologists, it might be the number of spawning salmon counted in a certain stream.

There are three basic ways for modelers to be involved with data: data design (figuring out what can and should be measured to test a given model), data acquisition (the rather plodding process of getting the data), and data analysis (figuring out what to make of the data once it has been acquired.) The first and third of these activities can be quite mathematical, and many modelers spend a good portion of their professional lives engaged with them. An aerospace engineer, for example, might work on figuring out what constellation of remote sensing devices will yield the best observability for a fleet of low earth orbit satellites. Her colleague might be involved in using the data from these sensing devices to fit orbits, a process that involves complex physical models, powerful software, and statistical savvy. The first kind of work probably entails high level calculations regarding atmospheric conditions, orbital patterns, and other things, but very little in the way of sensing data. The second kind of work probably entails wading through large sets of such data. Both jobs require a solid understanding of underlying physical models, but the relation to data is very different.

For modelers involved with data, the central challenge is to connect data to a model. The connections can be arbitrarily complicated, and might involve a vast array of mathematical tools. For example, a fisheries scientist studying atlantic cod might be interested in understanding what the optimal rate of harvesting might be for a certain cod fishery. Data at her disposal might include yearly estimates for population size for the last 10 years, as well as yearly estimates for “fishing effort”. The challenge is to couple these factors with an intrinsic growth model for the species to derive an optimal management policy. The growth model might involve a system of difference or differential equations that describes yearly growth as a function of
population size. The population estimates probably involve a regression on “count data”, where what is counted might be the number of fish caught per hook-line hour in a certain region of a certain fishery. The fishing effort estimates probably involves permit records from a government agency. Note that there are multiple layers of modeling, and the data is subject to a high level of uncertainty. Tackling this problem requires a good grasp of elements from statistics, biology, sampling theory, and discrete optimization, at the very least.

3.3 Computation

The idea that a modeler needs to have some relation to both mathematics and data analysis is in some sense built in to the definition of a mathematical model. The modeler’s relation to a digital computer is less obvious. One way to understand the centrality of the computer is to think of it as the most natural means of connecting the other two strands: the computer can play a huge role in model analysis, data processing, and visualization even for problems too difficult to admit analytic headway. I would argue, however, that the real significance of the computer lies in the nascent notion of “computational thinking” as a fundamental paradigm in modern problem solving. This position is gaining support from a number of quarters. Jeanette Wing, a computer scientist at Carnegie Mellon University, argues in [12] that training students to become fluent in these thought processes is an essential task for scientific educators. Borwein and Bailey, both mathematicians, argue in [4] that the computer plays an essential role in checking conjectures, forming hypothesis, and devising proofs, activities typically seen as belong to the domain of “pure mathematics.” The basic tenet of these positions is that the digital computer is a fundamental analysis tool with which aspiring problem solvers of all sorts need to become conversant.

Just as there are many ways for modelers to approach mathematics and data, however, so there are many ways to leverage the computer. On one end of the spectrum lie modelers for whom the computer is really just a means to perform elementary tabulations and visualizations. A production management modeler, for example, might calculate cost functions in a simple spreadsheet like Excel, and use the same software for visualization. In the middle of the spectrum lies use of some broad-spectrum mathematics package like Matlab or Mathematica. These sort of packages may not be the fastest options, but they have a wide range of functionality and are excellent for rapid prototyping. The most sophisticated computer users might be involved in lower level languages, C++ or Fortran, for example, where more of the component algorithms will need to be written from scratch. For numerically intensive applications, the added speed of these compiled languages can be a bonus, but using them requires a good deal of programming skill.

One of the difficulties in performing scientific computing is in knowing what can be computed and what cannot. At what point does a dynamic programming problem become too large to solve on a desktop? How many sample points does a good Monte Carlo simulation need? Should numerical integration used fixed or variable step size? A scientist or mathematician who attempts to use a computer to solve practical problems needs to be prepared to deal with these questions, though they will be more pressing in some applications than in others. In every case, modelers who use a computer need to ensure that their code does what they claim it does. This requirement generally translates more into correct execution than formal proof, but requires at least a minimal degree of “programming literacy.” It is also critically important that the programming work be done in a way that is reproducible and configurable. For this reason, scripting languages are far more powerful than programs that only accept command line inputs or graphic interfaces.

4 AIMING WIDE: BENEFICIARIES OF A BROAD-BASED MODELING CLASS

Good modeling can apportion its energies in many different ways along the three axes of mathematics, data, and computation, but all three of these things generally need to make an appearance. To the extent that any one of them gets entirely omitted, the activity strays from the modeling mainstream.

This fundamentally interdisciplinary nature is important to keep in mind, especially when teaching modeling as a mathematics class. Whatever mathematical modeling may be, it is not a sub-branch of mathematics. Legitimate “branches” of mathematics are characterized by a body of related techniques or points of common interest, while the set of techniques and interests that fall under the heading of “modeling” is a vast jumble. The AMS 2010 subject listings [1] do not include “modeling” as a mathematical specialty (although they do include the theory of modeling, as well as subject-specific instantiations in fluid mechanics, quantum theory,
systems theory, and mathematical education.) Moreover, while there are journals devoted to mathematical modeling as a broad enterprise, many more are devoted to some discipline specific variant of the business (e.g., ecological modeling, supply chain modeling, etc.) These discipline specific journals often contain formidable mathematics, but they generally eschew the theorem-proof structure so familiar to mathematicians, and only a fraction is monitored by MathSciNet. It thus seems clear that whatever mathematical modeling might be, it spills well beyond disciplinary lines.

Real-world implementation details need not dominate curricular considerations, of course. But a modeling class that mirrors the broad, interdisciplinary nature of the subject has a lot of potential advantages, not all of which are limited to the classroom. This section examines possible beneficiaries of such an approach.

4.1 Students: training and job skills

In the preface to his book *An Introduction to C*-Algebras, Kenneth Davidson states that “when approaching a new problem, you always know both too much (about the wrong things) and too little (about the question at hand). A big step towards the solution is figuring out exactly what it is that you really need to know” [5]. While Davidson formed these remarks specifically in the setting of a mathematical problem, they apply to the art of solving problems in general: the capacity to figure out what to figure out is a fundamental skill for would-be problem solvers, be they scientists or politicians.

In the setting of a mathematical modeling course, these comments suggest that training students to recognize and be comfortable with a wide range of solution techniques is a valuable focus, even when such solution techniques are not “mathematical”, per se. For example, one modeler might work on an analytic proof that a certain system of ordinary differential equations admits no limit cycles, while another might run a number of numerical simulation that suggest the same result (but don’t prove it.) Which of these approaches is most useful depends on the context: the analytic proof is a stronger result, but if it takes six months of focused effort while the numerical simulation can be whipped off in an afternoon, engineers (at a certain level in the production chain) might well prefer the simulation. Likewise, whether the results refer to the population growth of a fish colony or to the control system for a commercial jet may influence their relative worth. Modelers who know what they need to know for a particular problem and a particular community are in a good position to work effectively in a wide range of settings.

In a similar vein, a modeling course that trains students to collaborate across disciplinary bounds makes a significant contribution to preparing them for what is an increasingly interdisciplinary world. Undergraduates who adhere to strictly disciplinary course regimes may emerge with a reasonable glimmering of their chosen fields, but little experience in applying their understanding in the broad, unstructured environment that characterizes research, engineering, or social problem solving. Successful problem solvers need to know how to fuse scattered bits of knowledge, but these “fusion” skills have few curricular breeding grounds. The wonderful thing about a modeling course is that it provides a rare opportunity not just to drill multiple skills, but to illustrate how these skills complement and support one another. Whether students go to graduate school, take jobs in consulting or technology, head to finance or work in the public sector, this combination of skills will be highly valued. By striking a strategic middle ground between subject-specific mathematics, computational heuristics, and communication skills, the modeling course can make a significant contribution towards producing students well-prepared for the technical work force.

4.2 Faculty: research and interdepartmental dialogue

To the extent that a modeling course aspires to attract and train students from a range of disciplines, it needs to accommodate the needs and best practices of those disciplines. Developing a modeling course that reaches out to students from many departments thus represents an opportunity for faculty from these departments to discuss their work and their methods. These discussions might be about curriculum or research: just as consensus about software can expedite a class, so can a more in depth discussion about the kind of research faculty are doing, and how that research courts student involvement. These kind of conversations can harmonize course offerings and spur various levels of faculty collaboration. This is especially important in the setting of liberal arts colleges, where research areas often have little in common and motivations to engage in technical discussions are to be encouraged.
4.3 Institutions: structures and programs

Since mathematical modeling is, in some sense, the core activity of most of science, it is easy to see the modeling course as a foundation on which to construct large scale programmatics. For example, ensuring that mid-level science classes that feed into the modeling class adopt similar software would be a first step towards a powerful sort of departmental coordination, one that might spin off into new, more effective visions for how to coordinate the math and science curriculum in general. A modeling class could serve as a linking thread for an institution’s undergraduate research activity, e.g. by imposing the class as a prerequisite for summer research awards, or by encouraging undergraduates who have done summer research to present in the class. Alternatively, the modeling class could materialize as an ongoing seminar, one in which mathematics and science students interested in summer research could be encouraged to participate over multiple semesters.

While the best role for the modeling class will depend on both the institution and the interests of its faculty, it is worth noting that efforts to develop significant cross-disciplinary research and education programs are, in the main, fundable. The National Science Foundation, for example, claims that the “NSF also understands that the integration of research and education through interdisciplinary training prepares a workforce that undertakes scientific challenges in innovative ways. Thus, NSF gives high priority to promoting interdisciplinary research” [2]. Undergraduate programs that actively court cross-disciplinary critical thinking are attractive to funding bodies precisely because so much scientific work these days has a cross-disciplinary flavor. Allowing funding potential to dictate programmatics is in general a bad idea, of course: as Karban and Huntzinger put in [8], “would you let a committee of scientific peers approve your choice of a partner over the next three to five years?” On the other hand, the benefits of funding can be far reaching, so if developing these kinds of programmatics is consistent with institutional identity and faculty interest, funding potential can be a useful goad to action.

5 KEY COMPETENCIES

I have argued that mathematical modeling is by nature a hybrid activity, and that there are compelling reasons to approach this class as something much broader than “just another mathematics class.”

This section addresses the issue of implementation. It proposes a short checklist of key competencies that in my opinion are central to the process of mathematical modeling. The list is not intended to be exhaustive, but it covers skills whose value will resonate well beyond the classroom. Each proposed competency is accompanied by concrete suggestions about how it might be achieved. To a certain extent, these suggestions are rooted in my particular circumstances as a professor of mathematics at a liberal arts college, but I hope that teachers at different sorts of institutions or with different interests will find them valuable as a point of departure.

Key Competencies for Effective Modeling:

• Translation. Describing physical processes in the formal language of mathematics is fundamentally an act of translation. Just as with literary translation, mathematical translation admits many “correct” renderings, only a subset of which are also elegant and fruitful. The task for the aspiring modeler is, in some sense, to refine his “taste”: to develop not just a body of tools that be used for a wide range of problems, but to understand the aesthetic and analytic implications of any particular choice.

As a classroom stratagem, focusing on problems that are described in vague, non-quantitative terms is a nice way to help students work on model formulation. For example, rather than merely introducing the logistic equation and explaining how it “applies”, I might ask students to write down a differential equation representing the density dependent growth of a population. This leads to a discussion of what “density dependent” might mean, and can produce a range of equations, each with a variety of parameters. By collecting and comparing these differential equations, the student begins to get a sense for both how free the modeler is and also how important it is to know something about the system being modeled.

One of the particular difficulties in teaching students to become good mathematical translators is that good judgement implies broad mathematical experience, something most undergraduates will not have. While it may be tempting to solve this problem by focusing on specific mathematical topics,
my feeling is that the capacity to compare models, even very simple ones, is more important than the capacity to use one or two models with great skill, and that the course should avoid making homework problems vehicles for exercising particular techniques. Some technique-specific drilling is unavoidable, of course, but my approach is to advance to issues of model comparison as quickly as possible. For example, I would prefer to introduce difference equations and differential equations on the same day, discussing how and when to use one or the other, than to develop either of these topics in depth but in isolation. Once both are on the table, they can be developed in parallel.

• **Formulating theorems.** Mathematicians have a uniquely rigorous sense of what a “result” is: a theorem with a correct proof. While this paradigm is less common in the applied sciences, that sense of rigor and finality can be a great boon to these fields, at least if applied discretely. Modelers who can draw rigorous analytic conclusions about some physical system (subject to the constraints of a model) have the potential to make significant contributions to their fields.

  While opportunities for theorem-proving abound in applied modeling, recognizing these opportunities, as well as the appropriate language to use, might not be. For example, students might remember from a course on ordinary differential equations that a system whose Jacobian has eigenvalues with negative real part is locally stable. In studying a dynamical system with uncertain dynamics or unknown parameters, students might find that they need to know something about the eigenvalues of a full class of matrices. The class may be too narrow to have attracted the interest of pure mathematicians, but that in no sense means the class is not worth studying: showing that all the matrices of this class have a certain eigenvalue pattern may have significant ecological implications. Being able to write these implications down as a theorem (or at least as a strong, definitive statement) is important.

  Examples like this drive my own approach to teaching students about theorem formulation. I try to make sure that any abstract mathematical theorem that we study in class gets translated into the language of some applied problem, e.g. “predators will always be able to invade in a two patch system with the following dynamics.” Conversely, I try to formulate formal conjectures for every model we write down in class. The conjectures may be empirically baseless and/or impossible to prove, but the mere process of forming conjectures gets students to think critically about the range of things that they can (and should) try to say.

• **Using the computer as an experimental tool.** The digital computer is a valuable modeling aid, both for analysis and for visualization. As Bailey et al. put it in [4], “the computer provides the mathematician with a “laboratory” in which she can perform experiments—analyzing examples, testing out new ideas, or searching for patterns.”

  Training students to make effective use of the digital computer as a “laboratory” does not, unfortunately, require that they become expert programmers. What it does require is that they engage in “computational thinking” [11], which Jeanette Wing describes as “a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.” Elements of this thinking include using abstraction, understanding recursion, and being comfortable with approximation.

  Since programming is a means to an end, whatever language is chosen should be easy to learn and widely available. Additional criteria include a robust linear algebra package, the ability to program basic for and while loops, random number generation for Monte Carlo simulations, and routines for optimization and integration. It is also very important that the language admit scripts, i.e. allow a sequence of commands to be placed in a text file and run (critical for repeatability and analysis.)

  There are many commercial and open source languages that satisfy these criteria. Good commercial choices include Matlab (an engineering standard), Maple and Mathematica (common in academic environments.) Reliable open source options include Octave, R, and Sage. Lower level languages like C, C++, or Java have the advantage of being free, powerful, and reliable, but they are probably too complex for students to pick them up “casually” in the midst of a modeling course.

  Teachers of modeling classes will need to reflect on which criteria are most relevant for the needs of their own students. Personally, I feel that open source options are to be preferred, not only because all students have access, but also because I find the open source community an admirable paradigm of resource pooling and collaboration. Octave is what I use in my class, since it has intuitive syntax, admits scripts, and runs on all major platforms. The down side of Octave is that I have experienced occasional difficulties getting the software installed on student computers, and the graphics are cumbersome. R is based on the syntax of the statistical language S. It is a powerful statistical package with a growing
user base. Like Octave, R admits scripts and runs on all major platforms, but its syntax is somewhat complicated.

- **Acquiring data.** Understanding where data comes from and how it relates to model validation are critically important skills for a modeler. Although science students can be expected to have experience acquiring data, most mathematics students will have had very little exposure to the process. This is unfortunate, as “developing good modeling skills is a “doing” activity, not a “watching” activity” [3].

In my opinion, the best way to get students to think about data and its relation to a physical system is to have them design and implement experiments. Concrete examples from different subject areas include:

- **Ecology:** Estimate wildflower presence by performing counts within randomized squares in some local wilderness area.
- **Physics:** Investigate air resistance by measuring the amount of time it takes for balls of various compositions to fall a fixed height.
- **Digital image processing:** Form a digital image data base with images from several classes (nature, people, still life) and use this data base as a test bed for investigating deconvolution algorithms.
- **Psychology:** Use interviews or some controlled experiment with traceable materials to investigate social network connectivity on campus.

Although these examples span a wide range of territory, their objectives are the same: to get the student to think about how the data is acquired, whether the data can be used to support a given model, and how errors in the data propagate into parameter uncertainty.

- **Working in groups.** The capacity to work in groups is fundamental for most jobs in industry and science. While students gain this experience in a range of contexts, it is important they understand group work dynamics in the particular setting of technical problem solving, particularly when the analysis of such problems typically entails long hours of analysis. It is also important that students learn to work with a range of collaborators, not just their friends, and to assume leadership roles as circumstances warrant.

In my class, each group turns in a single report, and all members receive the same grade. Assigning a group leader might be a good idea, where the project grade is slightly overweighted for the leader: this would mimic the “project leader” structure common in industry, and compel students to come to terms with the dynamics of command. Leaders would need to be rotated, and the rotation and overweighting creates certain organizational challenges, but the result is a class structure whose group dynamics closely reflect those likely to be encountered in the world of work.

- **Writing.** Unlike pure mathematics, modeling is often qualitative and inconclusive. Successful defense of a model often involves learning how to tell a story about the underlying process, the modeling techniques, and the data. Students who are to be well prepared for the outside world need practice telling such stories, and because of its fundamentally cross-disciplinary nature, a modeling class is a uniquely good place for them to get this practice.

In my class, writing practice comes in several forms. For each project that I assign, the students are expected to turn in what is essentially a small paper, i.e. a document that has a title, an abstract, a conclusion, and a bibliography, and which has been vetted for grammatical and spelling errors. Graphics are to be inserted tastefully, and I pay strict attention to captions, axes, and units. Assumptions are to be clearly stated, model parameters need to be unambiguously identified, and conclusions need to be drawn in the muted language of statistics. I have found that insisting students use Latex poses no major obstacle: a brief template based introduction usually suffices to get them going, and the group work helps insure that the more knowledgeable share their skills.

- **Presenting.** As with any human endeavor, good work must be well communicated to be appreciated. Accordingly, after each project, I have a representative from each group give a brief (10 minute) presentation on their work. In general, the presentation involves several powerpoint slides, and consists mostly of graphics that illustrate the gist of the results. The presentation is not a place to talk about the details of their mathematical derivations: it is a forum for discussing assumptions, approaches, and solutions. Striking a balance between keeping things general enough to be understood by everyone but
specific enough to have technical merit is a challenge, but I believe it is a skill that will serve students well in whatever trajectory they follow.

6 CONCLUDING REMARKS

It is important to keep in mind that a mathematician’s perspective on the role of mathematics in the modeling process is probably very different from that of the rest of the modeling community. Mathematicians are trained to appreciate rigor and formal elegance, standards which can be brought just as easily to bear on equations that describe real phenomena as equations that do not. While elegant proofs have cachet, they generally do little to support phenomenological understanding. Thus to the extent that mathematicians use modeling as an excuse to write theorems and develop proofs, they pursue an objective tangential to that of the rest of the modeling community.

I find it helpful to think of modeling as a “mixed media” art form. The tools of this art include mathematics, computation, and data handling, but inevitably other elements enter the picture, things like team work and persuasion. Teaching an undergraduate audience how effectively to manipulate these elements is a large task, not because any one of them is intrinsically difficult, but because they all have to work in harmony. Questions such as what elements should be included in the model, what constitutes a “result”, and to what extent these results capture the truth of the underlying system admit multiple answers, as they are more matters of taste than technique. Moreover, mathematical modeling is used so widely and in so many different ways that any attempt to teach it in a way that is consistent with its real-world implementation necessarily strays well beyond the bounds of conventional mathematics. This observation lies at the root of the position I have taken in this article, namely that this course is best conceived less as a mathematics course than as a practical course in real-world problem solving, i.e. a course that teaches students to artfully leverage a range of tools (mathematical, computational, heuristic) to gain understanding about issues in the world.

This position merits some caveats. First, it is certainly the case that the more mathematics one knows, the better armed one is for constructing and analyzing mathematical models. Adherence to the idea that the modeling course is not a good forum for introducing a lot of new subject matter thus shifts the responsibility for learning that material to other courses. This shift may be difficult for a variety of personal, programmatic, or institutional reasons. Second, I have proposed that this course should actively cultivate a broad array of meta-mathematical objectives, a laudable goal in principle but one inevitably hampered by time constraints: depth, as well as breadth, are required for excellence. Although my experience in the classroom suggests that merely attempting this synthesis under the umbrella of a single class can offer a tremendous undergraduate learning experience, other teachers may find that a more focused approach has different but equally valuable results. Lastly, the opinions I’ve expressed here are largely predicated on my personal experience, both as a researcher who had done modeling in several different settings, and as a teacher in a small, liberal arts institution whose educational menu adheres relatively tightly to disciplinary paradigms. There is considerable latitude in how to conceive of mathematical modeling and what its role might be in the undergraduate curriculum, and I fully expect that teachers at different sorts of institutions or with different professional experiences will have divergent opinions. I believe, however, that like many topics in education, this is one that benefits from a multiplicity of voices.

REFERENCES


