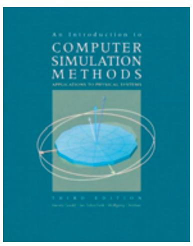
Editor: Mario Belloni, mabelloni@davidson.edu
Send announcements to bortega@computer.org

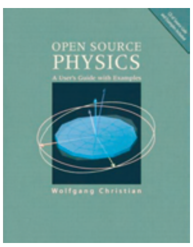
COMPUTATIONAL METHODS WITH DEPTH AND FLAIR

By Stephen Weppner

H. Gould, J. Tobochnik, and W. Christian, *An Introduction to Computer Simulation Methods: Applications to Physical Systems*, 3rd ed., Addison-Wesley, 2007, ISBN: 978-0805377583, 720 pages.

W. Christian, *Open Source Physics: A User's Guide with Examples*, Addison-Wesley, 2007, ISBN: 978-0805377590, 224 pages.





An *Introduction to Computer Simulation Methods* and *Open Source Physics* and the tools they describe are welcome additions to undergraduate computational science courses. They're also valuable resources to help supplement upper-level undergraduate physics courses with computational modeling components.

A Solid Foundation

The authors of an *Introduction to Computer Simulation Methods*, Harvey Gould and Jan Tobochnik, are well-respected contributors to the field of scientific computing. The first edition of the book, released roughly 20 years ago, was instrumental in showing that a computational science textbook could, from the beginning, expose students to interesting simulations with graphical output. The pedagogical approach the authors took is that students learn scientific modeling by doing. For the third edition, the authors have continued with these methods and embraced the Java programming language as well as Wolfgang Christian's Open Source Physics Library (www.opensourcephysics.org). The library makes it easy to draw, plot, control, and manipulate objects, and gives users a good basic set of numerical tools. *Open Source Physics* is a companion manual that describes the library. The latest *Open Source Physics* materials (code library, programs, and curricular material) are freely available and housed in the OSP collection on the ComPADRE digital library site (www.compadre.org/osp).

The code used in the books is open source, allowing instructors to use the libraries, models, and applications in courses without purchasing the books. I urge instructors who haven't yet tried Java as an instructional language to download the library and auxiliary materials and explore the developed components' power. Ideally, the open source paradigm will celebrate the spirit of the instructional computational science community by inviting developers to augment and improve the Open Source Physics Library's source code. For example, Douglas Brown, of Cabrillo College, has developed wonderful tools for manipulating video, creating XML applications, and launching software; Andrew Gusev and Yuri B. Senichkov at Saint Petersburg Polytechnic University in Russia have added higher-order ordinary differential equation solvers; and Christian has borrowed some control interfaces from Francisco Esquembre's excellent Easy Java Simulation tool to augment the library.

A Worthy Approach

The Open Source Physics Library is the central resource that the latest edition of *An Introduction to Computer Simulation Methods* requires for its high-level graphics, controls, and integrative objects. Using this library, students can devote time to scientific simulations without having to prematurely master higher-level programming details. I agree philosophically with this approach. In the past, when I taught computational science courses, I opted for no required textbooks. Instead, I used a variety of resources, including many from the Open Source Physics Library's Web site. *Computer Simulation Methods* broaches relatively complex interactive scientific models quickly,

 Copublished by the IEEE CS and the AIP 1521-9615/08/$25.00 © 2008 IEEE

letting students become participants in the process from the start. This will be my first choice for a textbook for future courses.

In *An Introduction to Computer Simulation Methods*' preface, the authors state that only one semester of physics and calculus is needed and no background in computer programming is required. However, the learning curve in the beginning, especially for students with scant programming experience, is demanding—students will have to learn Java and master an integrative development environment (IDE) at the same time. There are simple Java IDEs, such as BlueJ (www.bluej.com) for novice programming students that will make it easy for them to run their first programs. It also better facilitates understanding programming inheritance and encapsulation through an intuitive graphical interface. However, for the large multipackage system developed by these authors, the syntax checking and package drop-down menus that the recommended Eclipse IDE offers will be a time-saver when students have to design new classes by themselves. In my courses, I begin with the instructional BlueJ IDE and then move to the more advanced Eclipse system after the first few classes.

The authors offer flexibility, advising instructors to go through the first three chapters as a core and then explore other chapters independently. After an introductory chapter that explains the authors' choices, motivations, and goals, students are plunged in chapter two to traditional falling ball and harmonic oscillator simulations in which the results are printed to the console. They're then introduced to object-oriented programming and the Open Source Physics Library. The average student won't comprehend everything discussed in this chapter the first time through, so the authors have created exercises that gradually introduce students to the Open Source Physics Library, Java, and most important, using the computer as a tool in science.

Chapter three is another exciting whirlwind. Students are introduced to interface and array programming techniques, and to differential equation and visualization interfaces (both 2D and 3D) in the Open Source Physics Library. This is then applied to simulations of objects with drag (coffee filters, steel balls, pebbles, and curve balls), nuclear decay, motion in electrical fields, and the cooling of a coffee cup. The standard numerical analysis treatment of the predictor-corrector and Runge-Kutte methods appear in the chapter appendix, which exemplifies the authors' pedagogical choice to introduce the material by illustration first. The chapter references show the wealth of material the authors have drawn on to create realistic and beguiling simulations and invite further study.

Chapter exercises, in general, aren't brief—for example, chapter three has 17 exercises that often include four or more parts that are scattered throughout the chapter and relate directly to the material just discussed. Exercises vary from simple code modification to writing utilizations based on similar applications. Problem difficulty is correlated to placement in the chapter; some of the later problems are full stand-alone, week-long projects, for instance. For instructors who like to assign in-depth programming projects, this book doesn't disappoint. Students acquainted with this textbook for a semester will emerge competent in introductory programming and be well prepared for intermediate work in Java or other programming languages.

Breadth

An attractive aspect of the book is the informal writing style and the joyfully creative problems the authors include beyond the standard exercises: coffee cooling, coffee filter dropping, baseball statistics, nerve impulses and neural networks, polymers, interference patterns, moving charges, percolation, stock market fluctuations, fractal growth models, genetics, and quantum Monte Carlo. Make no mistake, though—this textbook is best suited for undergraduate physics majors, but the ease with which the authors jump from seemingly diverse topics underlies the strength of this pedagogy and the Open Source Physics Library, and showcases the potential it has for future work in chemical, biological, and ecological models.

An Introduction to Computer Simulations Methods' real strength is that it offers much more than a standard one-semester course in computational science; it has an almost encyclopedic breadth to it. I could envision a course developed from chapters one to six, and from additional chapters chosen by the instructor. Later individual chapters could also be investigated in other upper-level physics courses (classical mechanics, statistical mechanics, thermodynamics, optics, electrodynamics, quantum mechanics, relativity, and cosmology are all included). It should be especially noted that the statistical mechanics sections have a rich depth because of the authors' research interests.

Weaknesses

A few limitations inherent in the authors' approach exist. Some of the text's strengths could be considered flaws by those who favor a concise self-contained approach. Because of the text's size, inherent flexibility, application diversity, and the number of numerical and programming techniques introduced quickly, some instructors and students might find it too overwhelming and untidy to use for a traditional one-semester modeling textbook. The Open Source Physics Library is stable, easy to use, well documented, but young. When compiled using Sun Java 6, hundreds of warnings (due to Java's continuing development) appear that might confuse novice programmers. These warnings don't affect running the programs, but, stylistically, they're a nuisance. The library also uses native Sun routines so it can't be used with other Java compilers. There are also areas in which further expansion would be welcome—the numeric routines for linear algebra, special functions, and statistics functions are sparse, for example. Fortunately, several well-developed open source libraries are out there that instructors could assimilate into this base set without too much work.

Overall, this is an extremely valuable set of texts that prove computational science courses don't have to spend an inordinate amount of time teaching programming before delving into interesting scientific simulations. I highly recommend *An Introduction to Computer Simulation Methods* for a modeling course and to augment upper-level physics courses with computer modeling projects. The Open Source Physics Library and its companion reference book are also highly recommended for developers who want to build their own Java applications with easily constructible high levels of controls, interfaces, and graphical output.

Stephen Weppner is an associate professor at Eckerd College in Saint Petersburg, Florida. His research is in computational nuclear physics. Weppner teaches scientific computing and has used many of the resources that can be found on the Open Source Physics Web site, including the Open Source Physics Library, Easy Java Simulation, Tracker, and the BQ Database. Contact him at weppnesp@eckerd.edu.