Java Meets Numerical Analysis

Review of *Java Number Cruncher: The Java Programmer's Guide to Numerical Computing* by Ronald Mak

Review by David H. Bailey, Lawrence Berkeley National Laboratory

To appear in *Scientific Programming*

Introduction

Java for numerical computing? Are you kidding!? Until recently, such a combination would indeed draw well-deserved howls of laughter from serious practitioners of numerical computing. The huge overhead of Java's traditional interpreted-code implementation has resulted in run times that have been typically at least an order of magnitude higher than traditional languages such as Fortran-90 or C/C++. As a result, Java has been a non-starter for large, numerically intensive scientific computation.

But times are changing. Partly because of the extensive interest in Java programming in the business and Internet world, "just-in-time" compilers and the like have been developed that result in much faster execution times, nearly competitive with C and Fortran code in many cases. As a result, computational scientists are now seriously looking at potential uses of Java for scientific computing. But who is going to write scientific Java? Perhaps some numerical scientists will become proficient in Java. But maybe we should also try to teach Java programmers the basics of numerical computing. Indeed, given the increasing scarcity of computer science graduates who are also trained in numerical analysis, perhaps this is a more realistic route to take.

Ronald Mak's book is very timely in this regard. It is targeted directly at beginning (or even not-so-beginning) Java programmers who would like to become familiar with numerical computing. It does not pretend to be an traditional course in numerical analysis, which quite frankly many present-day computer science undergraduates avoid like the plague. Rather, it teaches Java programmers what they need to know to be numerically literate, so as to be equipped to take on serious technical computing tasks when needed.

The book starts out by describing in detail the IEEE-754 floating-point standard, both single and double formats. The author first drives home the point that IEEE arithmetic is *not* the same as the real number system—for example, there is potential for significant loss of accuracy when two nearby floating-point values are subtracted. The book continues with topics such as the potential for difficulties when a large number of floating-point values of different sizes are summed, finding roots of equations using basic iterative techniques such as Newton's iteration, finding interpolating and data-fitting polynomials, and linear regression.

Chapter 7, for example, discusses the trapezoidal rule and Simpson's rule for integration. It also presents techniques for numerical solutions to differential equations, including Euler's method and the Runge-Kutta scheme. Again, the restraint that the author exercises here is remarkable. I'm sure that most numerical analysts writing such a book could not resist the temptation to include here a furtive write-up of their favorite advanced

techniques, say for numerical quadrature. Space could also have been devoted to detailed discussion of techniques for numerical solutions of 2-D and 3-D partial differential equations. Instead, Mak continues to stick to his formula of providing a very detailed and readable account of basic numerical methods.

Beginning in Chapter 9, the author discusses matrix computations. Here, as in some previous chapters, the author provides a Java software package, which in this case is for matrix operations. With this facility, the author can focus on the concepts of matrix computation rather than on the detailed mechanics of carrying out such computations. Issues such as matrix condition numbers are discussed in Chapter 11. Even here, the focus is on concepts rather than on theorems, proofs or advanced implementation techniques.

Part IV (beginning with Chapter 12), entitled "The Joys of Computation," starts out by saying "Numerical computation isn't all work and no play." In these chapters the author gives several examples of computations using a "BigNumber" and a "BigDecimal" package, which perform high-precision integer and floating-point computation, respectively. Some of us would choose to differ with Mak's characterization of this material as "play," since in recent years numerous important mathematical and scientific results have been obtained using such high-precision computations. Besides, much of our Internet commerce relies on security schemes based on high-precision arithmetic. In any event, this material is actually quite well written. The author covers topics such as large integer factorizations, computing mathematical constants and functions to high precision and fractals.

In summary, this book would make an excellent undergraduate course in numerical computing, suitable for a wide range of students in computer science, physical science and engineering. It is also very well suited to professional Java programmers who would like to become more familiar with the world of scientific computing. It has a practical, down-to-earth approach that avoids exotic material, opting instead for a thorough and understandable coverage of basic material. It includes (and in fact relies on) software available from a website.

Many students or other readers, after completing Mak's book, will continue their careers with a greater appreciation of the issues and techniques of numerical computing, although perhaps they will not specialize in this arena. Others may find the topic sufficiently engaging that they will pursue more serious coursework and career paths in scientific computing. Either way, the computational science community will be enriched as a result.