



REVIEWS OF MAPLE, MATHEMATICA, AND MATLAB: COMING SOON TO A PUBLICATION NEAR YOU

By Norman Chonacky and David Winch

IN OUR INTRODUCTORY ARTICLE TO OUR UPCOMING REVIEW SERIES ON MAPLE, MATHEMATICA, AND MATLAB (*CiSE*, JAN./FEB. 2005, PP. 8–16), WE ASKED FOR FEEDBACK. THE LETTERS TO THE EDITOR WE SUBSEQUENTLY RECEIVED (SEE FROM THE EDITORS, PP.

3–4 in this issue) deliver two messages: first, that this review series will serve a real need, and second, that we must be thorough in our treatment of each product. We didn't intend the introductory article to be an in-depth account in any sense. Instead, we tried to set a context for the future reviews by relating certain background information—histories and design principles—and a brief overview of current uses for each of the three productivity packages. At the time, we had more such information about Mathematica and Maple than for Matlab. Subsequently, Matlab's inventor, Cleve Moler, authored an article that addressed much of the points we were missing.¹

In view of this information's emergence, and in response to the feedback we've received, we felt we should include some selections from Moler's article in this installment. Here, we will set everything on a more equal footing, and the full series will start in the May/June issue with a review of the 3Ms from the undergraduate education perspective. This will also give us more time to consult with users of these packages and to ensure that we gather equivalent information for all three. All quotations in the rest of this article are from Moler's article, which we encour-

age you to consult as we're providing only an interpretive summary here.

The Origins of Matlab

Unlike both Maple and Mathematica, which emphasize their origins in symbolic computing (as our historic accounts pointed out last time), Matlab's origins are in numerical computation: numerical linear algebra lies at its heart. That notwithstanding, Maple and Mathematica both have some numerical computational capabilities, and Matlab includes a symbolic computational toolbox. Significantly, this toolbox is built on Maple's symbolic computational kernel, making Matlab and Maple "kissing cousins" of a sort, and illustrating one advantage of Matlab's modular design.

The difference in origins explains why Matlab is so much more dominant in the engineering community than in the scientific community. As Moler explains:

I visited Stanford in 1979 and taught ... the graduate numerical analysis course. I had the students use MATLAB for some of the homework. Half of the students in the class were from math and computer science, (They) were not impressed by my new program. It was based on Fortran, ... and it did not represent current research work in numeri-

cal analysis. The other half were from engineering, and they liked MATLAB. They were studying subjects ... such as control analysis and signal processing, and the emphasis on matrices in MATLAB proved to be very useful to them.

Although Matlab's conceptual origins can be said to be in Moler's PhD thesis, written two decades before the product's first commercial release in the early 1980s, earlier versions gradually evolved as Moler learned more about programming and engineering practices.

The computational algorithms came first. In 1967, Moler and his former thesis adviser, George Forsythe, published *Computer Solutions of Linear Algebraic Systems* (Prentice-Hall), a textbook containing working code in three computing languages. J.H. Wilkinson and C. Reinsch followed with their *Handbook for Automatic Computation* (Springer-Verlag, 1971), which included the original Algol code for matrix eigenvalue computation that researchers at Argonne translated into Fortran to produce Eispack. Similarly, Moler helped create Linpack, a package of Fortran programs for solving linear equations.

Moler, then a mathematics professor teaching numerical analysis and matrix theory at the University of New Mexico, wanted his students to be able to use these packages without writing Fortran programs. In the late 1970s, he therefore taught himself to parse computer languages and developed the first version of what we might recognize as Matlab. This Fortran package could be compiled to run on many computers (mainframes

in the late 1970s and early 1980s).

In the early 1980s, Jack Little met Moler and hit on the idea of putting Matlab on the newly emerging IBM desktops. Reprogrammed in C, and including toolboxes and graphics capabilities that took advantage of the new hardware, Matlab emerged as the flagship commercial product of the MathWorks, which Little founded in 1984.

And Therefore ...

For reasons already stated—its historical development and design objectives—Matlab flourishes in industry for doing specific, practical tasks. Its modular architecture now hosts a long list of specialized toolboxes that reveal the wide scope of places the product has proven its usability and value. Another advantage of modularity is that users can get specific capabilities, prized especially by engineers, without buying the whole suite.

It is interesting to note that all three productivity packages appeared about the same time. Each one grew from quite different inspiration, and has since evolved to build on its original strength. Yet, each has in some ways grown closer to the others in the course of this evolution—we mentioned, for instance, that Matlab added symbolic computation capabilities. Each package has its own

high-level programming language, one generation removed from Fortran, C, and the like, which allows users to specify meaningful calculations in just a few steps rather than pages of code. Each product has an integrated set of tools that permit users to graphically display computational output. They can now use all three to produce interactive computer simulations of physical systems and to display the results in congenial, understandable forms. This is the basis on which we will compare them in three different use contexts—undergraduate education, research and development, and publishing and communication.

At this point, we remind you that we are launching this series of reviews as an experiment in review formats. We, the authors, are both educators and researchers with broad experience in science and education. We're undergraduate (and graduate and professional) educators, as well as researchers and publishers, but we're not experts in any of the three packages. Thus, we rely heavily on experienced users to provide us with the use data we need as a body of information to interpret and weigh based on our experience.

We have a list of volunteer consultants, some obtained on our own and

some recommended by the companies themselves, but we can certainly use more. We therefore renew our appeal for feedback from interested readers as we proceed with this experimental enterprise. If you're a dedicated user of one or more of these packages, please consider joining in this review process by contacting us to become an information provider. If you're a critical reader, please consider giving us a critique of how this review format informs or frustrates you. In either or both cases, send your messages to us at cise-editor@aip.org. **CS**

Reference

1. C. Moler, "The Origins of MATLAB," *Matlab News & Notes*, Dec. 2004; www.mathworks.com/company/newsletters/news_notes/clevescorner/dec04.html.

Norman Chonacky, most recently a senior research scientist in environmental engineering at Columbia, is currently a research fellow in the Center for UN Studies at Yale. His research interests include atmospheric physics, environmental sensors and the management of data derived from sensor arrays, the physicochemical behavior of material in environmental systems, cognitive processes in research and education, and applied optics. Chonacky received a PhD in physics from the University of Wisconsin, Madison. He is a member of the American Association of Physics Teachers (AAPT), the American Physical Society (APS), IEEE Computer Society, the American Society for Engineering Education (ASEE), and the American Association for the Advancement of Science (AAAS). Contact him at cise-editor@aip.org.

David Winch is emeritus professor of physics at Kalamazoo College. His research interests are focused on educational technologies. Winch received a PhD in physics from Clarkson University. He is coauthor of *Physics: Cinema Classics* (ZTEK, videodisc/CD 1993, DVD/CD 2004). Contact him at dmwinch@kitcarson.net.

Stay on Track

IEEE Internet Computing reports emerging tools, technologies, and applications implemented through the Internet to support a worldwide computing environment.

Internet Computing

www.computer.org/internet/

