(CISE EXCLUSIVE CONTENT)

TECHNOLOGY REVIEWS

3Ms for Instruction: Reviews of Maple, Mathematica, and Matlab

Norman Chonacky and David Winch

Most CISE readers have probably used Maple, Mathematica, or Matlab for several years. With this review series, our goal is to help you decide whether one of the others is better suited to your temperament and current practice than your original choice. For those of you new to integrative computing packages, our goal is to enable you to make an informed first choice.

In this installment, we begin to examine how these tools serve the professional work of undergraduate education. Within this context, we'd like to raise several significant issues for those teaching undergraduates to be scientists and engineers. We point to some exemplary materials and offer our own paradigms for major educational uses, which provide a framework for discussing the packages and drawing some implications for those issues in a concluding installment. In subsequent issues, we’ll explore how the tools serve scientific and engineering research and communication.

Undergraduate Education

We begin with the premise that science and engineering undergraduates should have experience in using modern computational tools. Indeed, this is already an explicit criterion for engineering schools? curricula in the US as prescribed by the Accreditation Board for Engineering and Technology (ABET). In this article, we examine the extent to which these tool packages so qualify: What kinds of computational experiences with them are appropriate for undergraduate students?

We're aware of the multiple goals that educational uses of computing technology must serve, as well as the challenge they present to a fair evaluation of computing software. Foremost in our minds as instructors experienced in the design of electronic instructional materials is the importance of appearance, simplicity, and user-interface functionality to the success of such materials. Yet, there are several types of user interfaces that connect users to different computing tasks according to different educational goals. This begs several questions: What are some major educational goals for science and engineering undergraduates? How are specific computing tasks related to those goals? How does each of the three productivity packages realize the required computations?

Undergraduates have a variety of learning styles and abilities, and they must simultaneously master material while learning how to learn. Ease of use in the packages? user interfaces as well as their adaptability to the variety of interactive mechanisms used in educational applications are key issues. Keep in mind, however, that the way and degree to which these are important depends on who the students are as well as the goals of the applications.

College and university instructors must be judicious in the type and intensity of development projects they undertake in creating educational materials, with respect to both the time and resources they dedicate. How well do these tool packages serve for materials-development work that faculty will likely perform alone? How efficient are they when fast response times are required for modifications? How expensive are they to purchase and, equally important, maintain?

Our approach to this review series is to describe the functions, features, and other elements these packages support and allow you to judge their value based on your values and objectives. To do this, we depart from making lists of features devoid of use-contexts, instead setting contexts in a variety of examples, both real and
idealized. We've principally drawn the real examples from each company's Web site, but each is implemented in only one of the three packages. By examining explicit application software, albeit developed for an educational purpose that might not match your own, we hope to present examples that help you envision how applications you create could work in each package.

From our perspective, these examples provide a concrete feature set to which we can refer when discussing how each package would implement an idealized example. We define each example as a paradigmatic application directed to one of the following educational roles for computational productivity packages in science and engineering undergraduate education: content tutorials, simulations, and computational programming.

**Proven Applications**
Maplesoft, Wolfram Research, and MathWorks all show evidence of wanting to maintain a position in the educational workplace. This is true in spite of the companies' self-histories in which only Maple claims to have had an academic birth and, as reported in the introductory article to this review series (CISE, Jan./Feb. 2005, pp. 8Â–16), has what most feels like an academic "personality." All three have an extensive number of exemplary educational applications available online:

- Maple
- Mathematica
- Matlab

These examples are resources for instructors to borrow as models or for students to use to supplement their learning opportunities on their own.

The significance of these examples goes beyond evidence of the companies' commitment to education and is broader than their use as classroom materials. They also illustrate a great deal about each package's range of computing power, and new developers can learn how to harness that power by examining the code. If you're already using one package, you might also find it valuable to "test drive" applications built with the others.

In evaluating the packages' look, feel, and capabilities through these examples, we didn't feel a need to select the same type of examples from each. Rather, we chose examples that demonstrated interesting features, strengths, peculiar characteristics, and so on. For simplicity, we present the examples in alphabetical order by company name.

**Maple**
We selected precalculus and calculus tutorial applications because they reflect Maple's mathematical character, and a rather extensive range of such lessons are available on the Maple site for free download. Many educators involved in the New Calculus movement have adopted Maple for implementing their material, which means that using it inducts you into a community defined by its novel approach to teaching calculus, gives you access to comprehensive materials, and provides an opportunity for a major overhaul of your mathematics curriculum.

The Precalculus Study Guide is an exercise-based electronic textbook developed by Maplesoft and available for sale on the company's Web site. The educational material in the lessons fall into a category that we'll call "tutorials." They don't stretch Maple's computational power, but they could conceivably be used for remedial work in colleges. Among the subjects this study guide covers are graphing of lines, polynomials, and rational functions; roots and rational powers; and transcendental and piecewise-defined functions. The free downloads are less elegant versions of the textbook lessons, but they nicely illustrate the standard Maple worksheet format. This permits interleaving comments and questions (displayed in developer-formatted text) with Maple expressions and functions (displayed in system-formatted text and shown, following command prompts [>], in red in Figure 1).
1. The Polar Form of Complex Numbers

   **Concept of Polar Form**

   Here is a complex number in standard “real part + imaginary part”
   component form.

   \[ z := 4 + 4 \sqrt{3} \cdot i; \]
   \[ \text{ComplexPlot}(z); \]

   When we look at the geometric representation, we might notice that
   this point forms a right triangle by looking at the real part of the number
   on the x-axis as one leg, and the diagonal line from the origin to the
   point as the hypotenuse. The angle that the hypotenuse makes with
   the x-axis is called the argument of the complex number.

   \[ \text{argument}(z); \]

   And the distance from the origin to the point is the absolute value of
   the complex number.

   \[ \text{abs}(z); \]

   These two numbers can identify the point completely. For example, this
   same number can be expressed in this form, which is called the polar
   or trigonometric form.

   \[ r \cdot \cos(\theta) + \text{Im}(z) \cdot \sin(\theta); \]

   **Figure 1.** Maple worksheet example from precalculus study guide. The line-by-line progression of the
   interactive lesson shows the interleaving of static information with interactive executables.

   The latest version of the Precalculus Study Guide includes 20 new tutorials, each using a GUI (Maplet) with
   buttons, input windows, graphical output windows, and so on. A GUI significantly alters the study guide's style,
   making it more congenial to exploration, but it lacks the more extensive text narrative possible with a worksheet
   because a GUI's advantages are diminished if the user must scroll to navigate it. Notice the complete absence of
   explanatory text in the window in Figure 2.

   ![Maplet GUI example from Maplesoft's Precalculus Study Guide.](image)

   **Figure 2.** Maplet GUI example from Maplesoft's Precalculus Study Guide. The tutorials use input
   boxes, control buttons, and display windows to let users input values and immediately see the
   results of the calculation using variables with those values. Notice the absence of explanatory text.

   Two of the educational issues we'll develop in this review are the costs and benefits of several material-delivery
   methods, including worksheets and GUIs. Distributing material through GUIs and worksheets is one way to take
   advantage of both. The Maple worksheet interface provides the user with immediate access to the Maple code
   and allows the user to change the code and explore additional options. The code's accessibility also provides a

http://www.computer.org/portal/site/cise/menuitem.f307eb76a9128778161489108bcd45f3... 4/26/2005
way to learn how to produce worksheets. However, the code for the Maplet tutorial lessons is one level removed from the user. Thus, the GUI protects the Maplet code from students’ casual tampering (although it’s fairly easy to access the code by moving to the Maple worksheet that generated the Maplet GUI).

The Calculus Study Guide, also available for purchase on the Maple site, is an electronic book to help those students taking their first course in calculus. The guide contains 31 Maple worksheets that give extensive coverage to five major subjects—limits, derivatives, application of derivatives, integrals, and applications of integrals—and 17 Maplet tutors that use the same GUI style as the Precalculus Study Guide. Other Maple course materials (including multivariable calculus, differential equations, partial differential equations, complex analysis, and matrix algebra) are also available for free download. Maplet tutors (for example, the Calculus 1 step-by-step differentiation problem solver) are useful as stand-alone “student practice modules” or for classroom demonstrations. An educational advantage of modularity in the packages is the ability for quick and responsive personalization of learning materials.

Mathematica
Mathematica has its own calculus tutor package, but we were prompted by the package's formalistic mathematical character and Wolfram's background as a physicist to select some physics applications. The company's Web site lists more than 750 Mathematica-related physics references (articles, books, demos, and courseware) available for download, so our application selection was somewhat arbitrary, but we did narrow the choices by eliminating examples written in earlier versions of Mathematica. In the end, we chose two examples from the quantum physics section: particle in a box and hydrogen atom.

These examples are part of a category of materials we call "visualizations." They don't use much of Â— Mathematica's computational power, but they demonstrate an important component of all the packages: rendering of results. Moreover, the contents of these examples are standard topics in most college physics and quantum mechanics courses. Figure 3 shows the GUIs for these two visualizations. Each example calculates and plots the respective quantum mechanical solutions to the infinite-square potential well problem and the hydrogen atom from user-specified values. The examples include text explanations using standard mathematical notation for the equations together with output-graphing windows.
Particle in a Box

A Particle in an Infinite Square Potential Well

The probability of finding the electron between $a$ and $a + \Delta a$.

![Graph of the probability distribution](image)

(a) Hydrogen Atom

This example visualizes the hydrogen atom as a quantum mechanical entity. Specifying integer values for the three quantum numbers, $n$, $l$, and $m$, gives a tool that displays the associated radial and angular wave functions as well as a probability density plot and allows the probability of finding the electron at a given radial distance. The plot is normalized to a length known as the Bohr radius, denoted by $a_B$.

Please specify a positive integer $n$, an integer between 3 and $n$, and an integer between 0 and 1.

![Graph of the radial probability distribution](image)

(b) Figure 3. Mathematica GUI visualization examples from the Wolfram Research Web site. Quantum mechanical probability density distribution for (a) a particle in a box and (b) the hydrogen atom.

The Mathematica site also includes other interesting visualization examples. The optics example, for instance, displays a detailed solution to Maxwell's equations for electromagnetic waves sustained in a homogeneous and isotropic dielectric medium, expressing this solution in both Cartesian and spherical coordinates. This example also shows the solution as an animated wave traveling through this medium, bringing out some subtle features of traveling waves that really require animation to be apparent to the novice. This capability thus has high value in lessons for first-time quantum learners.

These lessons incorporate the recently developed GUI with its Web-based method of delivery—WebMathematica. (We briefly discuss alternative delivery systems for all three packages, emphasizing Web delivery, at the end of this section.)

Matlab

We first looked for engineering-like applications, such as systems operations, to test Matlab, which has been widely adopted by the professional engineering community. However, during our initial Google search of existing Matlab examples, we found an even better choice in Erik Cheever's Visualizing Phasors (the Matlab file is available for download) directly from a Swarthmore College Web site.)
We selected this application for the Matlab example because it serves to mediate a particularly thorny learning task—understanding phasors—but also because our paradigmatic simulation example will be an exercise based on the behavior of AC electrical circuits analyzed using phasors.

Visualizing Phasors is a tool for depicting the relations between phasor and time graph representations of sinusoids. The user selects input parameters (voltage, impedance, and frequency) and then observes the corresponding time-dependent graphs of current and voltage (see Figure 4).

![Visualizing Phasors](image)

**Figure 4. GUI with results from the Matlab simulation example. The Visualizing Phasors application shows a phasor diagram’s relationship to its associated graphs of time behavior for the quantities represented by the phasors.**

The tool then calculates the current’s behavior through from the voltage applied across the impedance element by computing it from the AC generalization of Ohm’s law. We will call this category of educational application a “simulation.”

Given that Matlab’s great strength is in the engineering community, however, we needed an engineering example as well. A compelling reason for this strong engineering bent comes from Matlab’s numerical calculation capabilities. The syntax of its arithmetic representations is based on the matrix as the fundamental numerical construct (a real number is even represented as a 1 x 1 matrix, for example). Moreover, the development environment’s architecture is geared toward computational programming as it uses multiple windows, command histories, search paths, and so on, which all favor code development.

A lab exercise developed by Jeff Holmes, a biomedical engineer, for his students at Columbia presents a nice example in which the educational objective-numerical problem solving—neatly dovetails with executable code that illustrates how to program in Matlab. For the lab experiment, the instructor furnishes students with a Matlab (M-code) file as a program for simulating the experiment they will perform. Students write their own Matlab programs to reduce their experimental data, and then use the M-code simulation to test their own computational...
algorithms. This application thus serves the dual educational roles of simulation and computational programming. The M-code and lab instructions are available for download at www.computer.org/cise/techreviews/.

Application Delivery
Except for the Mathematica Optics example, users would normally need to have the application package to execute the cases we've described here. However, Mathematica's Web-based delivery system lets us use the applications without buying the package—albeit with restricted ability to explore the code and no ability to modify it. Maple and Matlab have their own Web-based delivery systems, which are similar to Mathematica's though different in functionality. The Matlab Web Server lets developers deploy Web-based Matlab applications. The user sends data from a Web browser to the Matlab application running on a server for computation, and the server returns results for display by the user's Web browser. Maple TA is a Web-based system designed for creating tests, assignments, and exercises; it automatically assesses student responses and performance—a useful feature for educational users.

Web-based delivery is a relatively new feature in all three packages. It's especially useful for educational applications, such as tutorials and simulations, because it favors instructional developers who value borrowing and swapping materials freely. Moreover, it gives a more consistent, although constrained, look to user interfaces across packages and obviates students' need to have the tool packages to operate the applications. We'll discuss some of the differences between—and implications of—these Web-based interfaces more fully in the next installment.

Paradigmatic Applications
Earlier, we singled out three roles for educational applications. To compare common application examples that compare how the packages work in educational settings, we'll invent one paradigm for each. Given our limited space, we'll develop the simulation role in considerably more depth than the tutorials and computations.

Simulation
The simulation example will be built around a lab-based scenario and will have features that spill over into the other two roles. It will serve as the baseline case.

We start with a definition of educational objectives:

- Enable students to collaborate in small groups as design and development teams.
- Serve as a medium for laboratory teaching assistants to interact with students in conducting "just in time" learning experiences.
- Provide properly motivated students with the opportunity for open-ended experimentation beyond the exercise's immediate requirements.

We then designed a paradigmatic exercise based on these goals. The flowchart in Figure 5 outlines such an Â—exercise based on our objectives, specifications, and assumptions.
Figures 5. Flowchart of the laboratory educational exercise specifying our "simulation" paradigm. Students complete the three branches to the exercise sequentially (moving from left to right) for the experiment.

This baseline case uses features that instructional applications generally need most. Our choices were animated visualizations, GUIs, text using standard mathematical notation, and interactive graphical output. With these features in mind, we chose to examine the behavior of AC electrical circuits, analyzed via phasors. The application should support the lab exercise by providing a prelab demonstration, in-lab design and simulation, and post-lab analysis and homework.

The application should visually display passive elements that students can arrange in circuits of their choice. The application should simulate the circuit’s current/voltage behavior under a choice of applied AC voltages. This is similar to the popular SPICE (simulation program with integrated circuit emphasis) technology, which encodes a circuit-performance simulation directly from circuits designed in the form of a schematic. In addition, the application should provide the capability for graphical comparison of the results from the computer’s simulation to data obtained by students from measurements on the physical circuits based on the design. Finally, it should have at least one Web-based implementation so that students can conduct it remotely from dorm rooms or public computer labs, as well as locally in the electricity lab.

We assume that, while in the lab, students will work collaboratively in groups and have access to a teaching assistant for help and advice. These assumptions favor graphics rather than text because graphical displays can, in the first instance, mediate group discussion more effectively. It is easier for several people to simultaneously look at and discuss graphical objects rather than text. In the second instance, graphical controls facilitate intervention by laboratory teaching assistants, who usually work over the shoulders of the group.

Tutorial
In the tutorial paradigm, we had a similar, but separate, set of educational objectives:

- Enable students to work individually.
- Serve as a medium for interactive learning.
- Provide properly motivated students with the opportunity for open-ended experimentation beyond the exercise's immediate requirements.

Here, individual activity replaces group work, and the interaction is between an individual learner and the organized material rather than between a group and a set of laboratory tools. In contrast to the baseline case, this situation favors easy readability and structural transparency of textual information. In consort with the baseline case, interactivity and open-endedness require embedded active objects, such as modifiable command statements and data containers.

With tutorials, our experience shows that the principal elements most needed are readable text using standard mathematical notation and interactive graphical output. In addition, sequential segments of the material should be organized by a clustering mechanism that lets the user alternatively "drill down" (open) and "reprise" (collapse) those segments, thus overlaying coarse-grained overviews on fine-grained details. The New User Guides for these three packages are good paradigms for tutorials. They conveniently furnish us with three realizations of the ideal and obviate the need for providing a set of detailed specifications. We can then simply evaluate the three package tutorials on how well they support our educational objectives.

**Computational Programming**
Proceeding to the last case, we developed the following educational objectives for the computational programming paradigm:

- Enable students to work individually or in groups.
- Serve as a medium for interactive learning of how to design, implement, and test computational algorithms.
- Provide properly motivated students with the opportunity for project-type experimentation—for example, collecting algorithms or adding appropriate input and output mechanisms—to produce comprehensive applications.

Here, we must accommodate both individual and group work with multiple interactions—among students, among students and programming tools, and among students and numerical analysis methods and algorithms. In contrast to the baseline case, this situation favors a congenial code development environment, a full and flexible programming "language," and the existence of and access to numerical analysis information and tools. This case also demands ease of transport for code segments and modules among students and with the broader user community.

In computational programming, a paradigm is actually a complete program development environment. In this case, we conveniently have three realizations of this ideal—namely, the development environments for Maple, Mathematica, and Matlab, themselves. Again, we can simply evaluate these package environments on how they support our educational objectives.

This review project has grown in size beyond our original estimate. Happily, this means that we've uncovered lots of interesting material. Unfortunately, it also means that we had to split the print edition into two installments. In the July/August issue of CISE, we'll continue this discussion by examining the inner workings of these tool packages. We'll describe the tools—flexibility, facility, and accessibility from the instructor-developer's standpoint. We'll also discuss the experiences of instructor-users, referring to specific parts of the
paradigmatic applications described here. This should provide a comparison of the effort involved and the outcomes achievable using each of the three packages for common end applications. We hope these observations and conclusions will carry some implications about the cost and benefits of using each package in a variety of educational contexts.

Researchers and developers, remember that we'll be continuing this series in September/October with a review of the packages from the perspective of scientists and engineers at work.

Also, keep in mind that the project of constructing this review series is an experiment in determining what is useful to you, our readers, for use in your own work. As we promised earlier, CISE will soon be soliciting your feedback on the new Tech Review format through an online "usability" evaluation—part of the magazine's renewed effort to evolve along with those whom we serve. We look forward to your cooperation in this effort.

**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. 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), the IEEE Computer Society, the American Society for Engineering Education (ASEE), and the American Association for the Advancement of Science (AAAS).

**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).

---

**Cite this article:**