Mathematica brings new understanding to teaching and research

I. Teaching with *Mathematica*

Two debates in the mathematics community today center on training new mathematicians or users of mathematics, and the relationship between research and teaching. The common thread is how best to impart mathematical knowledge and a way of doing mathematics. Nothing is more essential than an enthusiastic and well-organized teacher. Every teacher, however, appreciates tools that are flexible, reliable, and stimulating. Traditional math classes, as taught with chalk, mimeographs, and textbooks in expository lectures face several limitations:

- Chalk works well in the hands of a skilled board-artist but is limited to the rough, nonreproducible figures that can be sketched in front of a class.
- Texts and mimeographs can present materials efficiently, but in their traditional form must balance expository motivation or clarity against depth.
- There is tension within course syllabi: should we cover examples more carefully to motivate the theory, or must we race through the definition-lemma-theorem-proof-corollary cycle? In frequently taught "service courses," such as calculus and differential equations, the syllabus has devolved to a "grab bag" of unmotivated topics—formal techniques like partial fraction expansions or series solutions to an ordinary differential equation—coupled with artificially neat exercises.
- There is discontent among both students and teachers with the lecture model where the math instructor "reads" lecture notes to a class. Students' responses are largely limited to homework exercises and exam questions.

To invigorate the study of mathematics, here are several ways that students and faculty can use *Mathematica*®—a powerful tool for teaching and learning.

Mathematica as a calculator

Why not just use a calculator? Students discover that *Mathematica* on NeXT™ computers is more flexible than a handheld symbolic calculator. Aside from its arbitrary precision arithmetic (exact arithmetic over the rationals), rich set of calculus and linear algebra operators, special functions, and revolutionary integration of sophisticated graphics and animation, *Mathematica* is a programming language designed to read like standard mathematics. This brings the computer within reach of a student or teacher who expects to focus on the mathematics—not the machine.

At the Rose-Hulman Institute of Technology, Mark Yoder, assistant professor of electrical and computer engineering, now teaches his image processing course in a laboratory of 25 NeXTstation™ computers. Yoder felt that advantages of using NeXT computers and *Mathematica* significantly outweighed the risk involved with introducing a new computing environment to his students.

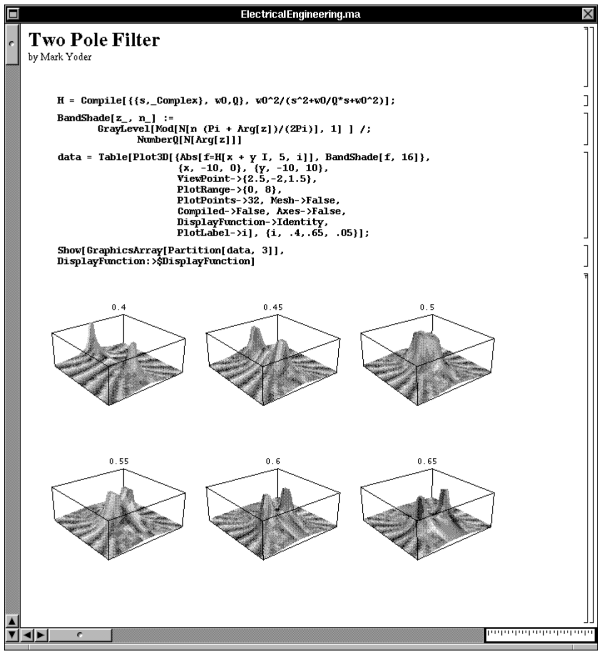
"I used to teach the image processing course in a traditional lecture setting, and the major disappointment I had with it was we couldn't easily implement a given algorithm to see what it did to an image," Yoder comments. "With *Mathematica,* I frequently introduce a new algorithm during the first half of a period and have the students implement the algorithm during the second half." Because *Mathematica* can quickly evaluate complex image processing algorithms and display the results, students can experiment and gain a better understanding of image processing.

Multimedia authoring with *Mathematica*

The superiority of *Mathematica* on NeXT computers over special purpose hardware or software becomes clear when students and faculty begin creating *Mathematica* Notebooks—electronic documents that integrate live algebraic or analytic calculations, text, graphics, animation, and sound into an easy-to-use environment. Students can interact with Notebooks by changing variables, defining functions, even writing their own animations or sound compositions; they can carry Notebooks on a disk or exchange them with other students via electronic mail. Teachers find Notebooks a flexible medium to write up labs and lecture notes for class.

Students can interact with Notebooks by changing variables, defining functions, even writing their own animations or sound compositions.

At the Georgia Institute of Technology, Brian Evans, a graduate research assistant, teaches a course in signal processing. His students write up their work in *Mathematica* Notebooks. In the Analog Filter Design unit, for example, students use animation to visualize the changes in the filter



Two Pole Filter
by Mark Yoder

```
H = Compile[{{s,_Complex}, w0,Q}, w0^2/(s^2+w0/Q*s+w0^2)];

BandShade[z_, n_] :=
       GrayLevel[Mod[N[n (Pi + Arg[z])/(2Pi)], 1] ] /;
              NumberQ[N[Arg[z]]]

data = Table[Plot3D[{Abs[f=H[x + y I, 5, i]], BandShade[f, 16]},
                     {x, -10, 0}, {y, -10, 10},
                     ViewPoint->{2.5,-2,1.5},
                     PlotRange->{0, 8},
                     PlotPoints->32, Mesh->False,
                     Compiled->False, Axes->False,
                     DisplayFunction->Identity,
                     PlotLabel->i], {i, .4,.65, .05}];

Show[GraphicsArray[Partition[data, 3]],
DisplayFunction:>$DisplayFunction]
```

Plots for a two-pole filter $\dfrac{w_0^2}{s^2+\dfrac{w_0 s}{Q}+w_0^2}$ *with Q varying from .4 to .6*

as they vary the parameters; later they can redesign the filter and observe the results.

Relating theory with application

Mathematica enables students to encapsulate knowledge in a usable form, whereas in a traditional course this encapsulation takes the form of theorems and sometimes, memorized proofs. In Brad Osgood's core calculus course at Stanford University, for example, students are guided into reinventing the bisection and Newton's root-finding methods, then introduced to *Mathematica*'s built-in root-finder function. Armed with this *Mathematica* function, plus some knowledge of the theory behind its operation, students can apply it later in the course. Similarly, they can build an Euler method for solving ordinary differential equations before being introduced to *Mathematica*'s sophisticated adaptive Runge-Kutta method. They compare these methods by error analysis, reinforcing the differential calculus learned before, and apply these encapsulated tools and theorems to a variety of physical problems that would have been beyond the scope of a traditional calculus course.

Mathematica as a math microworld

Also at Stanford, David Stork is pioneering a course in Adaptive Pattern Recognition and Neural Networks using *Mathematica*. Since *Mathematica* supports both procedural and functional styles of programming, students who have some programming experience can implement their projects in C or Pascal-like style, though they can transfer the neural net methods that depend on vector calculus, linear algebra and some statistics directly into *Mathematica* Notebooks. It is rewarding, especially in this visually oriented discipline, for students to see the results in graphical form.

Jon Barwise, founding co-director of the Center for the Study of Language and Information at Stanford (now at Indiana University), comments in an article on the gulf between research mathematics and what mathematicians teach: "There are several computer programs designed to teach standard materials, that relieve faculty of the chore. The worst are basically automated page turners with on-line grading to help the student get through some standard text. Such programs only exacerbate the problem, increasing the distance between what the students learn and what mathematics is really like.

"There are many alternatives, however. For one, we can use the computer as a tool for experimentation. By making mathematics come alive, it can serve as a powerful source of intuition and inspiration. A program like *Mathematica*. . . provides tools with which even freshmen can explore uncharted waters, looking for patterns and then trying to understand them. Using such programs made me realize what toy examples I have always used in teaching calculus. We have in calculus a most sophisticated tool for exploration, but we seldom encourage students to use it except in a goldfish bowl fished over by countless generations of earlier students. . . .

". . . By allowing the student to carry out huge calculations no one could perform with pencil and paper, students can apply what they learn to original and significant problems. And the graphic capabilities of the modern computer allow us to visualize functions and other patterns that would have been completely inaccessible when I was a student. . . . In rethinking our curriculum. . . we have an exciting opportunity to introduce students to the joys of mathematical discovery that motivate us as mathematicians." (From "Notices of the AMS," v. 37 no. 8, 10/1990, p. 1017.)

Integrating *Mathematica* with NeXTstep applications

What's wrong with using special purpose math applications? Nothing, as long as they can be integrated into an instructor's needs. *Mathematica* is easily extensible in the NeXTstep® environment, providing a narrative structure that can be reshaped at anytime by the instructor. At the simplest level, one can execute C routines from within *Mathematica*. One of Stork's Notebooks, for example, uses a Voronoi tesselation routine that was compiled from public domain research code and called from within *Mathematica* like a standard *Mathematica* function.

There are instances when it is better to mask the full power and complexity of *Mathematica,* especially when dealing with a physical or geometric simulation. One can use NeXTstep to write highly interactive and robust custom applications backed by *Mathematica's* computation engine. With Interface Builder,™ one can create and redesign these custom front ends easily, leaving the courseware author free to think about content and learning.

Mathematica is easily extensible in the NeXTstep environment, providing a narrative structure that can be reshaped at anytime by the instructor.

For certain simulations, custom front ends can bring mathematical models to life in a way that could only have been done, more laboriously, with physical demonstrations. Best of all, students can carry these virtual labs around in a backpack and even write their own experiments. Advantages of custom front ends include capability masking, refined input methods, and refined output.

- For example: Students work with a simpler, friendlier custom interface so they don't have to learn as much *Mathematica* to study a particular problem. A student of knot theory, for example, might want to graphically assemble or dissect a knot while seeing the associated polynomial invariants constructed in parallel, without worrying about *Mathematica* syntax.
- With complete control over input, custom interfaces can be more interactive than the standard Notebook front end, for example, with sliders dynamically attached to *Mathematica* functions or graphic objects, as in the knot theory example.
- Output can be refined. In place of the Notebook's top-to-bottom outline format, a custom front end can present more sophisticated forms of animation, perhaps of multiple objects with transparent overlays for a pharmaco-kinetic whole-body box model.

For more information on custom front ends, see "Creating custom front ends to *Mathematica,*" on pages 22-23 of this issue, and "Integrated curriculum encourages students to become problem solvers," and "PhaseScope™ energizes the study of mathematics," from *NeXT on Campus* Winter 1991.

II. Mathematical Research with *Mathematica*

Twentieth-century mathematicians study a universe of objects—logics, prime numbers, dynamical systems, groups, knots, and spacetime singularities, to name a few—but since Gauss' day three centuries ago, they have moved away from a constructivist, "bare-handed" calculation approach. Why? Partly because of a shift in style and because the objects couldn't be computed. Theorems like "black holes radiate and small black holes radiate faster than big ones" were not discovered or proved by calculation in the common sense of the word. Until now, computers offered mathematicians essentially no help in either exploring their universe or proving their conjectures, because computer programs did not speak mathematical language; it took too much effort to teach them even the most basic mathematical knowledge and forms of reasoning. Computers and especially computer software have matured to the point where mathematicians can use computers in addition to their beloved chalk and blackboard.

Though mathematicians may entertain sharply different notions of calculation, a symbolic manipulation program (SMP) like *Mathematica* certainly can do small algebraic calculations that form part of the underbrush of mathematical research. Moreover, despite the injunction against relying on pictures for proofs, mathematicians often sketch ideas as figures. These sketches, whether algebraic or geometric, are guided by heuristics of experience. *Mathematica's* computable graphics, which are bound to some mathematical structure as opposed to merely graphic structure, provide an intelligent blackboard for this kind of exploration. In *Mathematica,* one works directly with mathematical types—graphs, differential equations, quadratic forms, tensors, and so on—rather than data structures or control structures that have little to do with the object of study.

Mathematica's principal strengths include: a rule-based pattern-matching, functional language; integrated manipulation of graphics, algebra, numerics, and sound; uniform kernel behavior across platforms; and, on the NeXT platform, a sophisticated document front-end called the Notebook.

Algebraic computation

For his doctoral research at Princeton University, John Sullivan (now at the Geometry Computing Project in Minnesota) used *Mathematica* for algebraic calculations that would have been tedious to perform by hand, and might have been "swept under the rug" in the final write up. In a paper on a crystalline approximation theorem for hypersurfaces, Sullivan calculated a bound on the number of unit balls that can be placed with centers on an
r-sphere in dimension *n*. Sparing readers the nonessential algebra, he encodes the calculation as a few lines of *Mathematica* using Gegenbauer polynomials, which can be independently verified or studied.

Using *Mathematica* as a scriptable algebraic calculator can be useful. Mathematicians have written packages to perform algebraic computations in topology (the Kauffman-Jones polynomials, and braid groups) and elliptic curves. But *Mathematica* is also a programming language, which unlike all traditional languages and most SMPs, provides a concise representation of algebraic expressions in fairly standard mathematical syntax.

Dana Scott, at Carnegie Mellon University, wrote an entire course in projective geometry, partly to explore how feasible it was to implement mathematical ideas on a personal computer.

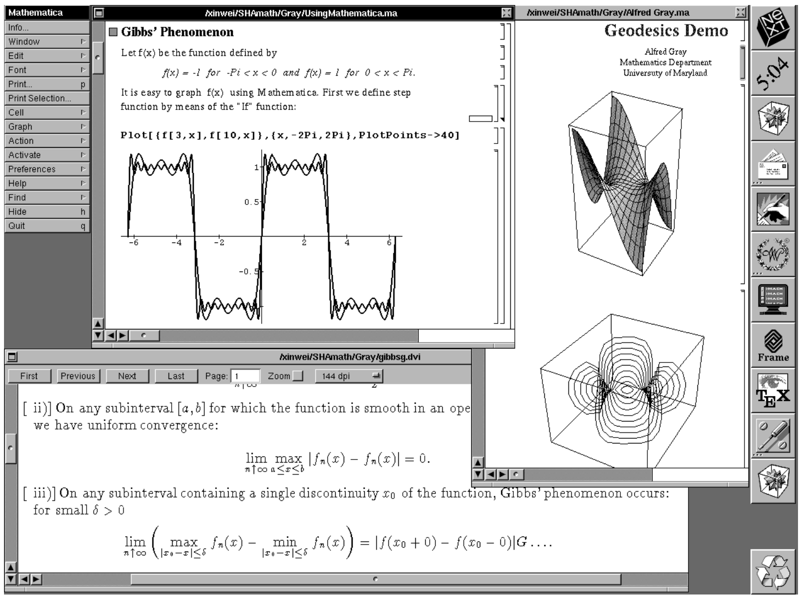
Scott says, "Mathematically, the most interesting outcome of the experiment of teaching this particular course was to find how easy it is to use the ideas of linear space theory, exterior products, and partial differential operators in presenting geometry, both in computations and in proofs aided by the powers of symbolic computation in *Mathematica*. The implementations of these ideas were done in a natural way—programming did not detract from the mathematics.

"The beauty of *Mathematica* is that we can not only do numerical computations such as:

```
PD[3 x + 4 y + z, 7 u + v - 25 w]
```

(which evaluates to zero showing that the point with coordinates {7,1,-25} lies on the line with coordinates {3,4,1}), but we can also allow indeterminants in the formulae that

Mathematica integrates seamlessly with other NeXTstep applications. Formulas can be exported in T_EX™ *form and viewed in a* T_EX *previewer. Graphs, formulae, sounds, and text can be cut and pasted between Mathematica Notebooks and any number of applications.*



Gibbs' Phenomenon

Let f(x) be the function defined by

$f(x) = -1$ for $-Pi < x < 0$ and $f(x) = 1$ for $0 < x < Pi$.

It is easy to graph f(x) using Mathematica. First we define step function by means of the "If" function:

```
Plot[{f[3,x],f[10,x]},{x,-2Pi,2Pi},PlotPoints->40]
```

Geodesics Demo

Alfred Gray
Mathematics Department
University of Maryland

[ii)] On any subinterval $[a,b]$ for which the function is smooth in an ope
we have uniform convergence:

$$\lim_{n\uparrow\infty} \max_{a\le x\le b} |f_n(x) - f_n(x)| = 0.$$

[iii)] On any subinterval containing a single discontinuity x_0 of the function, Gibbs' phenomenon occurs:
for small $\delta > 0$

$$\lim_{n\uparrow\infty} \left(\max_{|x_0-x|\le\delta} f_n(x) - \min_{|x_0-x|\le\delta} f_n(x) \right) = |f(x_0+0) - f(x_0-0)|G\ldots.$$

give us conditions for something to happen."

To define certain functions on polynomials with indeterminate coefficients, a Degree function, for example, Scott taught *Mathematica* a few definitions (called "transformation rules" in *Mathematica*).

"The virtues to be emphasized here are that these rules work and are easy to write. It's important to be able to prototype ideas quickly, and my experience is that *Mathematica* allows that. Once the idea is made coherent and firmly demonstrated, then attention can be directed to more efficient implementations." His implementation in *Mathematica* of a generalized exterior product is essentially the standard definition:

```
PE[p___, 0, q___]      := 0;
PE[p___, l_ + m_, q___]:= PE[p,l,q] + PE[p,m,q];
PE[p___, n_ l_, q___]  :=
    n PE[p,l,q]/; DegP[n]== 0;
PE[p___, l_,l_, q___]  := 0;
PE[p___, l_,m_, q___]  :=
    -PE[p,m,l,q] /; Order[l,m] == -1;
PE[x,y] = w; PE[x,z] = -v; PE[y,z] = u;
PE[u,v] = z; PE[u,w] = -y; PE[v,w] = x;
```

Furthermore, working out explicit formulas allows the student or teacher to draw the figures corresponding to theorems using the extensive graphical capabilities of *Mathematica*.

Geometric computation

One of *Mathematica*'s chief strengths is its ability to produce powerful and mathematically useful graphics. At the University of Maryland, Alfred Gray has used *Mathematica* to study Gibbs' phenomenon. "The graphics interface is useful for pointing the way to new results. I discovered a new type of Gibbs' phenomenon that exists for Bessel functions, but not ordinary trigonometric functions. It arose when I did some graphs of Fourier Bessel series. Although Bessel functions have been studied for more than one hundred years, few graphs were made because of the computational complexity. The graphs were comparatively easy to do with *Mathematica*, and I have written an article with Mark Pinsky (Northwestern University) that gives a theoretical basis for the graphs."

Gray has also written *Mathematica* packages to compute geodesics—curves of shortest length—on two-dimensional surfaces. In his original setting, Gray collaborated with George Francis (University of Illinois, Urbana), using a Silicon Graphics computer to render the equations that were computed by *Mathematica*. In version 2.0 of *Mathematica,* using MathLink™ in the NeXT environment, he hopes to drive rendering software directly from *Mathematica* on a NeXT computer. "I feel *Mathematica*'s ability to generate graphics is its most useful feature."

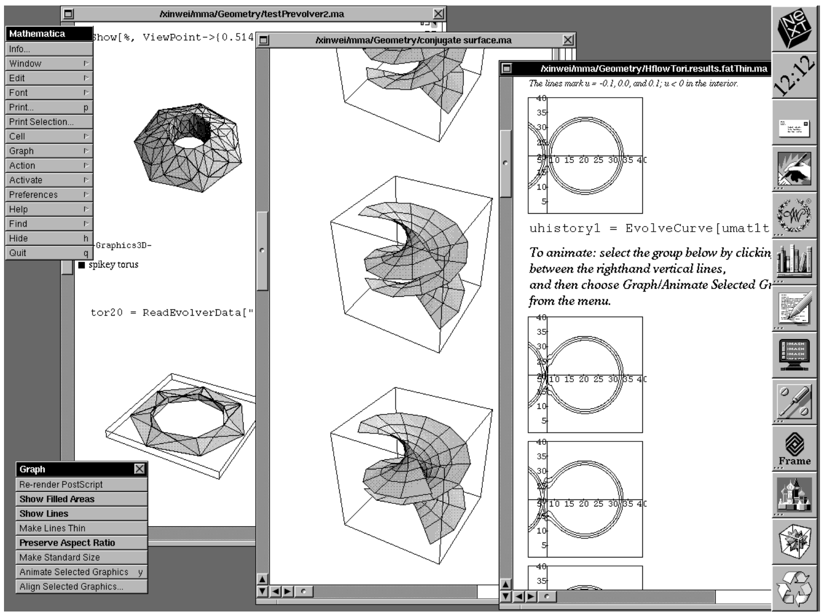
At Stanford University, graduate student Sha Xin Wei has been experimenting with *Mathematica* as a tool for studying curvature-dependent evolutions of surfaces. The study of geometric variational problems traditionally has used methods from partial differential equations, geometric measure theory, and differential geometry. Until recently, the use of numerical simulations has been restricted because software has not been sophisticated enough to directly visualize the differential equations.

"With *Mathematica*," says Sha, "one doesn't need an expert programmer or a numerical analyst to quickly and flexibly explore the qualitative behavior of geometric nonlinear partial differential equations. Animation and color graphics provide a natural way to study all sorts of parametrized evolutions.

"It's important to be able to prototype ideas quickly, and my experience is that Mathematica allows that."

"One powerful application from the Geometry Computing Project is Ken Brakke's Surface Evolver program, which was written in C but used rather cumbersome data files to describe the topology and geometry of the initial surface. As soon as I got Brakke's Evolver, I wrote a *Mathematica* script that allowed me to prepare initial surfaces and view Evolver output using *Mathematica*'s powerful 3D plot functions, saving me from dealing with hordes of vertex, edge, facet lists. This was the easiest and quickest way to prepare initial data and to see the results in a geometrically meaningful fashion.

"The difference between *Mathematica* and all other graphics programs is that graphic objects can be directly linked to their geometric/algebraic structure. An example I like to use



```
Show[%, ViewPoint->{0.514
tor20 = ReadEvolverData["
```

The lines mark u = -0.1, 0.0, and 0.1; u < 0 in the interior.

```
uhistory1 = EvolveCurve[umat1t
```

To animate: select the group below by clickin between the righthand vertical lines, and then choose Graph/Animate Selected Gr from the menu.

Mathematica can be used to present evolutions of surfaces or parametrized families of surfaces as animation sequences. Shown are (left to right): the output of Brakke's Surface Evolver, written in the C language; an isometric family of minimal surfaces, and evolution of a torus by mean curvature.

is the classical Weierstrass representation of minimal surfaces. Such a surface can be described via a complex vector-valued integral involving its Gauss map and another complex function. By simply inserting a complex rotation $e^{i\phi}$ into the Weierstrass representation, you can have *Mathematica* draw an entire smooth family of minimal surfaces out of one—it's a lovely mix of complex analysis, geometry, and animation."

Integrating *Mathematica* with other applications

Mathematicians have started to use *Mathematica* in concert with other applications. The Display PostScript® environment supports a $\TeX$ previewer, and with *Mathematica's* psfig utility, it is simple for Gray to paste *Mathematica* graphics into his $\TeX$ articles. Applications like FrameMaker® make it easy to integrate research into book-length monographs.

Gray says: "A NeXT workstation is the most convenient place to use *Mathematica*. First, the ability to combine text and graphics in one file makes working with *Mathematica* on a NeXT much easier than on other workstations. Even if the other workstations have windows, they do not have Notebooks. Secondly, editing is much easier. I can work three-times faster on a NeXT than on another workstation. Thirdly, it is easy to transfer information between *Mathematica* and word processors. When I write a research article that requires graphics, I work with $\TeX$ and *Mathematica* simultaneously. I generate the graphics with *Mathematica,* write the article with $\TeX$, and put the graphics in the $\TeX$ document."

Sha Xin Wei intends to use Interface Builder to construct front ends for other geometry applications and *Mathematica*. "I expect to use applications in concert with *Mathematica,* some running on other machines on a network, exchanging geometric and algebraic results with each other. We'll be able to write mathematical hyperdocuments with $\TeX$-like displays that have directly manipulable algebra along the lines of FrameMath™ as well as embedded geometric simulations. My goal is to be able to open an article where, say, a surface is described as some minimal solution to a variational problem. I'd like to be able to select a patch on the surface using a mouse or perhaps a glove, define my own perturbation by writing a few equations with a pen, and watch it deform, all within one window—an intelligent blackboard, working in conjunction with the mathematician."