Using *Mathematica* in a Graduate Numerical Methods Course

Shirley Pomeranz

Department of Mathematical and Computer Sciences
The University of Tulsa

1. Introduction

This paper is a continuation of observations on the use of the Computer Algebra System (CAS), *Mathematica* (Wolfram Research, Inc.), as the software of choice for a graduate numerical methods course[1].

Currently, there are many software tools that can be used for numerical methods. In the "old days" FORTRAN was the primary tool. Today there are procedural, functional, and/or rule based programming languages. There are computer mathematics systems: *Axiom, Derive, Macsyma, Magma, Maple, Mathcad, Mathematica, Mathview, Matlab, Milo, Reduce*, etc. Spreadsheets, especially *Microsoft Excel*, are very popular with engineering students. This bewildering array of tools makes the choice of software very difficult.

The course, Numerical Methods for Engineers and Scientists (MA7273), taught at The University of Tulsa, and offered through the Department of Mathematical and Computer Sciences, deals with numerical methods for solving partial differential equations. The students are beginning graduate students from diverse engineering and math/computer science disciplines. The students enrolled in the course in the 1998 fall semester represented the Departments of Mathematical and Computer Sciences (five students), Mechanical Engineering (four students), Petroleum Engineering (two students), and Chemical Engineering (one student). My experiences and observations, as well as student comments, are presented in the following sections of this paper.

2. Mathematica version of the course

The new features of the course this 1998 fall semester, included the use of a text that specifically chooses *Mathematica* as its associated software package. The text is Numerical Solutions for Partial Differential Equations, Victor Ganzha and Evgenii Vorozhtsov, CRC Press, 1996. The authors provide a disk with *Mathematica* notebooks that is integrated with the text. The text's use of *Mathematica* is not restricted to numerically solving and graphically representing the finite difference and finite element solutions of problems. The symbolic capabilities of *Mathematica*

are used in order to perform theoretical analyses: consistency and stability (i.e., convergence) analyses of the various time-marching finite difference schemes, dispersion and dissipation analyses, etc.

The course was conducted by first allowing several weeks of *Mathematica* lessons (with a computer projection system in the classroom to demonstrate the *Mathematica* lessons). *Mathematica* tutorials (consisting of 17 *Mathematica* notebooks) were written and made available via the TU computer network, by TU Professor, Dr. Dale Doty. For the first several weeks, the graduate mathematical modeling course taught by Dale and the graduate numerical methods course that I was teaching met together to cover the *Mathematica* material, which was presented by Dale. This created a community of TU graduate students “immersed" in *Mathematica*.

The course topics are primarily numerical methods for partial differential equations, and include finite difference methods, method of characteristics, and the finite element method. The course focuses on the descriptions, derivations, uses, comparisons, and analyses of methods. Theoretical analysis and numerical work are involved. For example, we numerically solve boundary-value and initial-boundary-value problems; perform consistency, stability, and convergence analyses for finite-difference time-marching methods for parabolic and hyperbolic problems; study efficient linear solvers for elliptic problems; and have an introduction to error analysis for the finite element method. *Mathematica* is used in all of these endeavors.

The rationale for offering a *Mathematica*-based course is based on the following benefits. The numerical capabilities of *Mathematica* alleviate tedious hand calculations. The line-by-line interpretation of code permits students to discover many types of errors and fix them as they arise. The graphics capabilities are an aid in the debugging process, and, more importantly, in interpreting and understanding the numerical solution to a problem. Being able to conveniently graphically interpret a solution is crucial for students’ understanding of the correctness of a solution[2]. Some of the many uses of the symbolic capabilities have already been mentioned (also see Section 3). If necessary, *Mathematica* can be used as a front-end to run code that is already written in FORTRAN or C, via the MathLink utility. MathLink for Microsoft Word and MathLink for Excel permit access to *Mathematica* from within Word or Excel, respectively.

Mathematica Version 3.0 has improved typesetting, user-friendly help, and palettes (which are similar to menus or toolbars). *Mathematica* is easier to program now that there are palettes that can create typeset expressions and carry out basic operations. The notation is the same as that of standard mathematics. Thus, at this level, there is no need to get involved with specific syntax rules[3].

On the other hand, in order to use *Mathematica*'s powerful programming (and other) capabilities, students must become familiar with the other aspects of *Mathematica*. And this does take time. *Mathematica* has a high and long learning curve[4]. However, students are able to use many of the simpler applications of *Mathematica* right away.

Furthermore, over time, as students work with *Mathematica* they learn, in a very natural way, about its more advanced features.

3. *Mathematica* projects

Mathematica's symbolic computation capabilities were utilized to derive finite-difference formulas (see Figure 1). For example, the Lax-Wendroff finite-difference scheme for 1-D transport equations was derived by constructing a quadratic Lagrange interpolating polynomial. *Mathematica* solved a linear system to obtain the polynomial coefficients; then, replacement rules were used to substitute these coefficients into the general expression for the polynomial (alternatively, *Mathematica* could have been used to directly perform the interpolation via its **Interpolation** or **InterpolatingPolynomial** commands). Finally, the polynomial was evaluated at an appropriate point (also determined by *Mathematica*). The resulting expression was the Lax-Wendroff scheme.

Mathematica notebooks that were provided on the disk were discussed in the text. The programs in these notebooks were used as templates. The students could run them directly or go into the code and modify the programs.

For example, one notebook performed stability analyses of time-marching finite-difference methods for parabolic and hyperbolic problems. This notebook used finite-difference schemes (as input) and produced plots (as output). Graphs of planar stability regions for various values of a given parameter (Courant number) were plotted as a function of the Fourier mode (multiplied by the spatial step-size). This permitted students to run numerical experiments and graphically predict the stability region for a given scheme. Then, more precise analytical arguments in support of the observed numerical results could be constructed.

Notebooks on the text's disk created graphics (which can be animated) that demonstrated wave propagation and which could be used to compare dissipative and dispersive effects of various finite-difference schemes (see Figure 2).

Students wrote their own *Mathematica* programs to solve an explicit finite-difference implementation of the one-dimensional wave equation. Their results included graphics and error analysis. Numerical experiments were performed that involved changing one line of the code and replacing a first-order approximation for the first time-line with a more accurate, second-order approximation. Students compared the resulting graphics, numerical results, and error estimates from both approximations. The numerical calculations (also performed with *Mathematica*) confirmed the accuracy of each of the two approximations.

A simple two-dimensional elliptic finite element problem using piecewise-linear shape functions was assigned to students as part of their take-home final exam. This (interactive) finite element *Mathematica* notebook was produced by this author at an NSF workshop[5]. Students solved a small finite element linear system and then interpreted their finite element solution using the *Mathematica* notebook. Students

interpreted their results graphically to produce 3-D surface graphics of the solution (temperature), contour plots, and plots of two-dimensional vector field (heat flux) components (see Figure 3).

4. Student comments

The following list gives some students' comments on their experiences using *Mathematica* in this course. Some of the comments are more directed to my teaching of the course than to the explicit choice of *Mathematica*. However, the reader will be able to distinguish between these two cases.

- I feel that using *Mathematica* in the classroom is a good idea, especially at the graduate level. ... I have found many other uses for *Mathematica* in my research and other courses.

- Use of *Mathematica* simplifies the problem to a great extent, especially the calculation. ... It really reduces a lot of manual work.

- I feel that ... a different program is more beneficial for more complex linear algebra problems ... MATLAB, FORTRAN, etc. Also ... need a *Mathematica* prerequisite to this class, ... so that the first three weeks don't need to be used learning *Mathematica*.

- Since class time is limited, less time should be spent on teaching students fundamentals of *Mathematica*. I believe that most students have the ability to teach that stuff to themselves. ... need more programming using *Mathematica*. Students should be encouraged to write *Mathematica* programs of their own, not just to use/modify available *Mathematica* notebooks.

- Overall, I think that *Mathematica* is a fairly useful tool. For example, using the program to visualize stability was helpful. *Mathematica* seems to be quite user-friendly in that the Help Menu is actually quite useful.

- I have found *Mathematica* a very useful tool in many senses. For example, I can use it to evaluate any algorithm before coding it in another programming language. *Mathematica* helps me to check the logic in a fast and easy way. I have used it in other courses for homework and the quality of the results has improved substantially.

- My first recommendation for someone who is taking a similar class is to buy a student version of *Mathematica*. I think that the more accessible the software is to you, the more likely you are to be able to successfully integrate whatever discipline you're studying with *Mathematica*. I also like the graphical capabilities of *Mathematica* for Numerical Methods because they allow you to see (and double check) the solution that you are producing. It seems that for more realistic problems *Mathematica* may be a little slow, but for learning I think that it is a great tool.

5. Summary

The stage must be set in order to ensure that this choice of software is well received.

- Equipment and software must be available and accessible.
- At least one individual needs to be actively involved in guaranteeing that help, computers, and *Mathematica* courses are available.
- The faculty needs clear-cut reasons to learn *Mathematica*.
- Enough time must be allowed for synergism to take place. Students should be given enough time to reach the stage at which they view *Mathematica* as a tool for enhancing their productivity.

There will always be tradeoffs in the choice of tools for teaching numerical methods. The "richer working environment[6]" and versatility of *Mathematica* make it a strong candidate for the software of choice in a graduate numerical methods course. *Mathematica* is a software package that will be useful to students, long after they cease to be students.

Bibliography

1. Shirley Pomeranz. Using CAS in a graduate numerical methods course. *1996 ASEE Annual Conference Proceedings*. Washington, DC.
2. David Epstein & Silvio Levy. Experimentation and proof in mathematics. *Notices of the AMS*, v. 42. 670-674. (June 1995).
3. Steven Wilkinson. Software Reviews: *Mathematica*. *The College Mathematics Journal*, v. 29, n. 4. 323-329. (September 1998).
4. Sandy Balkin. Taking calculus with *Mathematica*. *The Mathematica Journal*, v. 4, n. 2. 52-53. (Spring 1994).
5. Robert Lopez & Mark Yoder. NSF Workshop: *Revitalizing the Engineering, Mathematics and Science Curriculum via Symbolic Algebra*. Rose-Hulman Institute of Technology. Terre Haute, IN. (July 10-15, 1995).
6. Allan Hayes. *Mathematica* as a tool for teaching elementary numerical analysis. http://www.bham.ac.uk/ctimath/reviews/. (May 1995).

Mathematica file:
from "Numerical Solutions for Partial Differential Equations"
Victor Ganzha and Evgenii Vorozhtsov, CRC Press, 1996

```
uxx = a* u[x-h] + b* u[x] + c* u[x+h]
s = Normal Series[uxx, {h, 0, 2}]
s1 = s - u''[x]
derlist = Prepend[
  Union[Cases[{s1}, Derivative[__][u][x], Infinity]], u[x]]
coefh = DeleteCases[Flatten[CoefficientList[s1, derlist]], 0]
ss = Solve[coefh == 0, {a, b, c}]
OutputForm[ss >> taylr.m]
First[ss]
uxx
uxx = First[uxx /. ss]
```

$$-\frac{2u[x]}{h^2} + \frac{u[x-h]}{h^2} + \frac{u[x+h]}{h^2}$$

```
OutputForm[uxx >> taylr]
```

Figure 1. *Mathematica* code for construction of the finite-difference approximation of the second derivative

Mathematica graphics:
from Numerical Solutions for Partial Differential Equations, Victor Ganzha and Evgenii Vorozhtsov, CRC Press, 1996

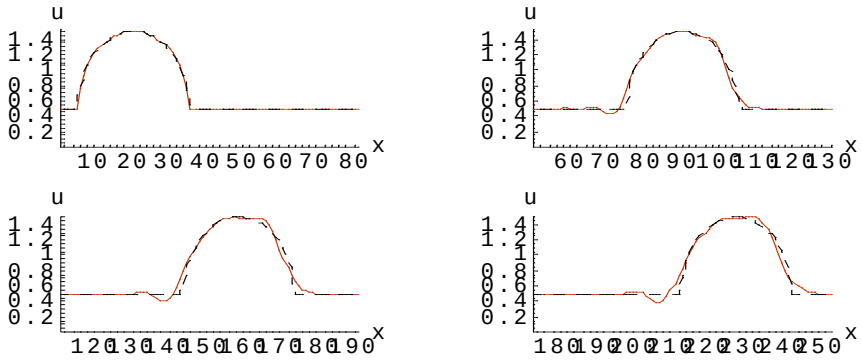
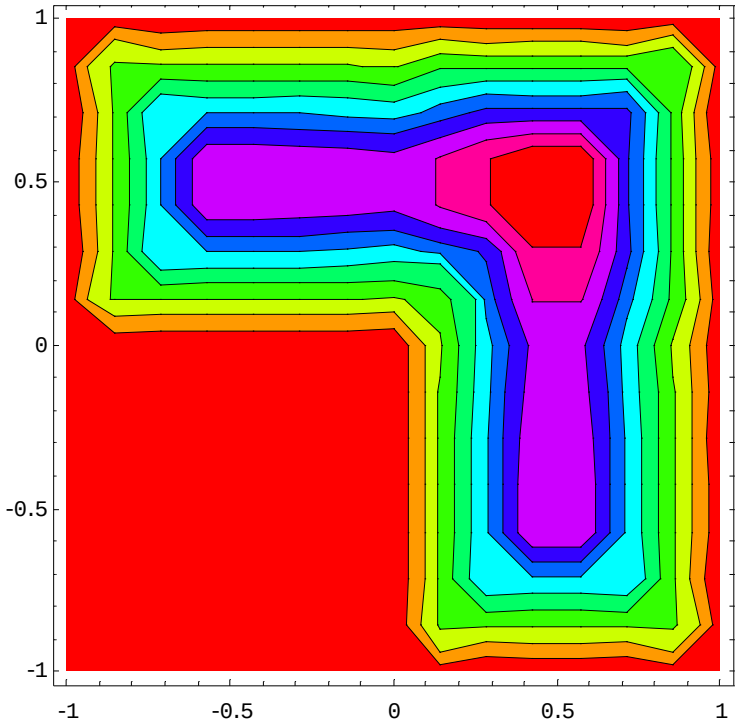


Figure 2. *Mathematica* graphics for the Law-Wendroff solution of a one-dimensional advection

Problem from
The Numerical Solution of Ordinary and Partial Differential Equations,
Granville Sewell, Academic Press, 1988.

Use the finite element method with linear triangular elements to solve Poisson's equation,
$u_{xx} + u_{yy} = -1$, on an L-shaped region in the plane.
Homogeneous Dirichlet boundary conditions are specified.

FEM Solution: Contour Plot



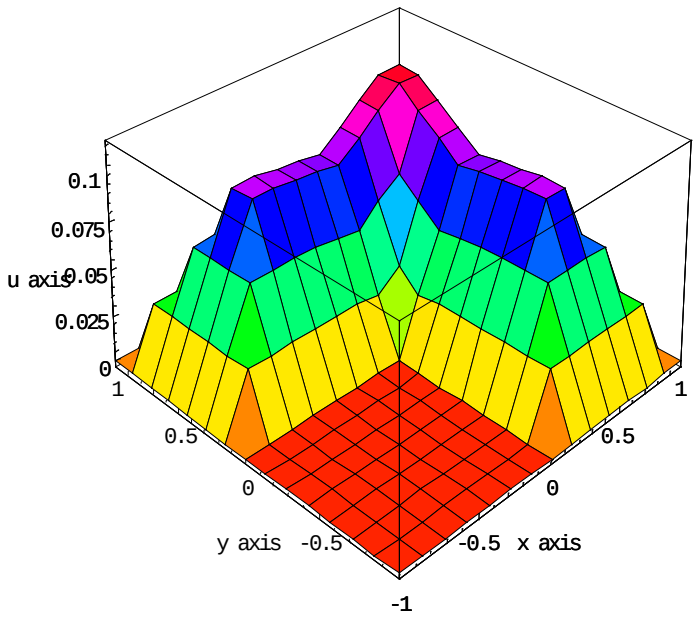


Figure 3. *Mathematica* graphics for visualization of a finite element solution

SHIRLEY POMERANZ
Shirley Pomeranz is an Associate Professor of Mathematics in the Department of Mathematical and Computer Sciences at The University of Tulsa. She is the 1999 ASEE Mathematics Division Chair and is a member of the Editorial Advisory Board for *The International Journal of Engineering Education*. Her interests include support of women in mathematics and research involving the finite element method.