TEACHING NUMERICAL ANALYSIS

IN A SMALL COLLEGE ENVIRONMENT

Daniel S. Yates
Randolph-Macon College

The purpose of this paper is to share my experience teaching a one-semester, undergraduate course in numerical analysis at a small, four-year private college, Randolph-Macon College, in Ashland, Virginia. I should say at the outset that I am not an expert at teaching numerical analysis; to the contrary, I am a novice. I brought several handicaps which I describe below. Yet I am encouraged by my first experience. The person I hope to reach with my commentary is that person who anticipates teaching a numerical analysis course for the first time and who might avoid some of the mistakes I made and be reassured by some of my successes.

Randolph-Macon College has an enrollment of approximately 1000 students and a mathematics faculty of seven. In the past half dozen years, there has been an average of about 10 math majors each year. Numerical analysis is not required for the mathematics major but may be elected by mathematics and computer science majors to satisfy their requirement for hours in the major field.

The Instructor I joined the R-MC faculty between terms during the 1987-88 school year when there was an unexpected need to identify an instructor for the numerical analysis course and several introductory statistics sections. For fifteen years previously, I had been a mathematics and computer resource teacher for public schools in the Richmond, Virginia area. My previous college teaching experience consisted of three years as an instructor at Virginia Tech in the nineteen sixties. I did have considerable experience in recent years teaching programming in BASIC and Logo at the public school and college levels.

My training in mathematics, twenty years ago, consisted entirely of pure mathematics, with two terms of statistics and no other experience in any area of applied mathematics. I had neither taught nor even taken a course on numerical analysis prior to the spring 1988 term, although I have, in recent years, explored some numerical techniques, such as writing a program to approximate the sum of a series and using the bisection method to obtain roots of an equation on a spreadsheet. I wanted to teach this course for my own professional development and because of my interest in using the computer to enhance the teaching of mathematics and to make learning more efficient (and more appealing to students).

The Course I consulted with several numerical analysis instructors at nearby universities to get different perspectives on the most prominent texts. I selected the Burden and Faires text (Prindle-Webber) because it is a respected text, but also because the authors take what seemed to be a sensible, middle of the road position on the role of the computer in such a course. That is,

they provide algorithms (without program listings) that are easily converted into computer code.

For the one-semester course, taught during the spring term, I arranged to move the class into the college's Computer Literacy Lab in order to have access to a demonstration computer with LCD overhead projection device and multiple computers and printers (all aging IBM's).

At Randolph-Macon the fall and spring semesters are short, lasting only 13 weeks, with a four-week January term in the middle. Because of the relatively short spring term, the topics to be covered in the course had to be chosen carefully. The course sequence reflected my personal preference:

Chapter 1	Mathematical Preliminaries
	Review of Calculus
	Round-off errors and computer arithmetic
	Algorithms and convergence
Chapter 2	Solutions of Equations in One Variable
	The bisection algorithm
	Fixed point iteration
	The Newton-Raphson Method
	Error analysis for iterative methods
Hour Test 1	
Chapter 2	Accelerating convergence
	Zeros of real polynomials
Chapter 3	Interpolation and Polynomial Approximation
	The Taylor polynomials
	Interpolation and the Lagrange polynomial
	Iterated interpolation.
Hour Test 2	
Chapter 6	Direct Methods for Solving Linear Equations
	Linear systems of equations
	Gaussian elimination and backward substitution
	Linear algebra and matrix inversion
	The determinant of a matrix
	Pivoting strategies
	Special types of matrices
	Direct factorization of matrices
Final Exam	(3 hours) comprehensive, but emphasized Chapter 6 material

Programming The course description in the catalog states that a prerequisite for the course is a "working knowledge of a computer programming language." I polled the students to determine their language of choice and was surprised that all 14 enrolled students indicated BASIC, although basic computer science courses at the college provide instruction only in Pascal. My conclusion was that the students were most comfortable with BASIC from their high school experience. I indicated that a continuing requirement in the course would be to translate the provided algorithms into programs that executed, and to use these programs to solve selected problems that included tedious computations.

Things that went well The use of the computer definitely motivated study of the numerical methods and added an extra dimension by eliminating much of the tedious computations by hand. In fact, I can't imagine teaching this course in any practical way without having access to the computer, both for in-class demonstrations and for students to code and use on non-routine problems and applications. I am convinced that the students perceived the computer as an indispensable tool and one that freed them to concentrate on the various approximating methods.

The demonstration computer, in particular, was a most important asset for it allowed me to follow theoretic discussions of approximating techniques and algorithms with quick, efficient solutions to sample problems. Being able to quickly display program listings, or to graph a complex function made the instructional process more efficient and demonstrated, over and over, the special symbiosis between man and computer.

Although I had some reservations about endorsing the use of BASIC as a programming language in the course (because of the negative press BASIC has received recently and the emerging role of Pascal in college computing curricula), I found that the algorithms translate directly into BASIC without the opportunity for "spaghetti programming" that detractors say BASIC permits. Non-structured programming by students was not a factor because of the nature of the algorithms presented in the text.

I elected to use a function graphing utility on several occasions when the objective was to approximate roots to specified degrees of accuracy. Most numerical routines to do this require that one begin with an interval that brackets a root. Yet we encountered occasional equations that were sufficiently complex that the general location of the graph was not easily determined. In these instances, I pointed out to the students that one could make use of the built-in mathematical functions in BASIC to quickly get a general feel for the characteristics of a function and its graph. The two steps are to enter the function into computer memory with a one-line program such as:

```
10 DEF FNF(X) = 16*X^4 - 40*X^3 + 5*X^2 + 20*X + 6
```

and then type, in immediate execution mode:

```
FOR X = -10 TO 10 : PRINT X, FNF(X) : NEXT
```

to produce a table of ordered pairs, (x,f(x)). In most cases, one can determine by inspection where the function is increasing or decreasing and the general location of the roots (although not for this example--here, there are two roots between 1 and 2). Of course, the interval and increment can be modified to either check another interval or to obtain a refinement within the previous interval. The students picked up on this technique immediately and made good use of it thereafter.

Things that did not work well I made the mistake of not providing a review of BASIC programming syntax prior to the first programming

assignment, and found that about one-third of the students had programming skills that were weak at best. Subsequently, I departed from the text to provide an overview of those techniques that they would need most often. My second mistake was in not providing practice programming exercises before returning to the text. Those students with weak skills continued to have trouble with their coding throughout the course. Next time, I will definitely devote more time early in the course to insuring that students can quickly translate the algorithms into code.

Although the move to the computer lab did provide for a demonstration computer and screen, the other computers in the room and the fact that each student was facing a computer that is kept running continuously provided yet another problem. Students, facing the front of the room, could surreptitiously develop their own agenda while giving the impression that they were paying attention to the instruction at the front of the room. Suffice it to say that I will avoid this situation in the future.

I also found that the majority of my students had a somewhat less than satisfactory recall of important results from calculus (consistent with the "Nothing transfers" hypothesis). But I suspect that this will always be a source of irritation for teachers regardless of the role of the computer.

Conclusions and recommendations Having considered the above, I have made several decisions for the next time I teach this course (spring 1989), and offer them for your consideration.

1) Retain the text; it is rigorous but accessible. The algorithms provided strike a good balance between the mathematics of the continuous processes that are the principle object of numerical analysis and the aspects of computers and codes that enable numerical solutions.

2) Classroom environment: restrict the equipment to a single, demonstration computer with projection capability or linked to large screen monitor(s).

3) Provide a review of programming language syntax prior to the first coding assignment.

4) Emphasize to faculty advisors the prerequisite of a working knowledge of a programming language.

Postscript I would be happy to share my tests and examination and/or a 5 1/4" disk of programs in BASIC for the IBM with any prospective numerical analysis instructor who would be interested. Write to Dr. Dan Yates, Mathematics Department, Randolph-Macon College, Ashland, VA 23005.