[The following essay appeared in the November, 1992 issue of *SIAM News* and the March, 1993 issue of the *Bulletin of the Institute for Mathematics and Applications.*]

THE DEFINITION OF NUMERICAL ANALYSIS

Lloyd N. Trefethen
Dept. of Computer Science
Cornell Unviersity
LNT@cs.cornell.edu
1992

What is numerical analysis? I believe that this is more than a philosophical question. A certain wrong answer has taken hold among both outsiders to the field and insiders, distorting the image of a subject at the heart of the mathematical sciences.

Here is the wrong answer:

Numerical analysis is the study of rounding errors. (D1)

The reader will agree that it would be hard to devise a more uninviting description of a field. Rounding errors are inevitable, yes, but they are complicated and tedious and —*not fundamental.* If (D1) is a common perception, it is hardly surprising that numerical analysis is widely regarded as an unglamorous subject. In fact, mathematicians, physicists, and computer scientists have all tended to hold numerical analysis in low esteem for many years—a most unusual consensus.

Of course nobody believes or asserts (D1) quite as baldly as written. But consider the following opening chapter headings from some standard numerical analysis texts:

Isaacson & Keller (1966):	1. Norms, arithmetic, and well-posed computations.
Hamming (1971):	1. Roundoff and function evaluation.
Dahlquist & Björck (1974):	1. Some general principles of numerical calculation. 2. How to obtain and estimate accuracy....
Stoer & Bulirsch (1980):	1. Error analysis.
Conte & de Boor (1980):	1. Number systems and errors.
Atkinson (1987):	1. Error: its sources, propagation, and analysis.
Kahaner, Moler & Nash (1989):	1. Introduction. 2. Computer arithmetic and computational errors.

"Error" ... "roundoff" ... "computer arithmetic" — these are the words that keep reappearing. What impression does an inquisitive college student get upon opening such books? Or consider the definitions of numerical analysis in some dictionaries:

Webster's New Collegiate Dictionary (1973): "The study of quantitative approximations to the solutions of mathematical problems including consideration of the errors and bounds to the errors involved."

Chambers 20th Century Dictionary (1983): "The study of methods of approximation and their accuracy, etc."

The American Heritage Dictionary (1992): "The study of approximate solutions to mathematical problems, taking into account the extent of possible errors."

"Approximations" ... "accuracy" ... "errors" again. It seems to me that these definitions would serve most effectively to deter the curious from investigating further.

The singular value decomposition (SVD) affords another example of the perception of numerical analysis as the science of rounding errors. Although the roots of the SVD go back more than 100 years, it is mainly since the 1960s, through the work of Gene Golub and other numerical analysts, that it has achieved its present degree of prominence. The SVD is as fundamental an idea as the eigenvalue decomposition; it is the natural language for discussing all kinds of questions of norms and extrema involving nonsymmetric matrices or operators. Yet today, thirty years later, most mathematical scientists and even many applied mathematicians do not have a working knowledge of the SVD. Most of them have heard of it, but the impression seems to be widespread that the SVD is just a tool for combating rounding errors. A glance at a few numerical analysis textbooks suggests why. In one case after another, the SVD is buried deep in the book, typically in an advanced section on rank-deficient least-squares problems, and recommended mainly for its stability properties.

I am convinced that consciously or unconsciously, many people think that (D1) is at least half true. In actuality, it is a very small part of the truth. And although there are historical explanations for the influence of (D1) in the past, it is a less appropriate definition today and is destined to become still less appropriate in the future.

I propose the following alternative definition with which to enter the new century:

Numerical analysis is the study of algorithms
for the problems of continuous mathematics. (D2)

Boundaries between fields are always fuzzy; no definition can be perfect. But it seems to me that (D2) is as sharp a characterization as you could come up with for most disciplines.

The pivotal word is *algorithms.* Where was this word in those chapter headings and dictionary definitions? Hidden between the lines, at best, and yet surely this is the center of numerical analysis: devising and analyzing algorithms to solve a certain class of problems.

These are the problems of *continuous mathematics.* "Continuous" means that real or complex variables are involved; its opposite is "discrete." A dozen qualifications aside, numerical analysts are broadly concerned with continuous problems, while algorithms for discrete problems are the concern of other computer scientists.

Let us consider the implications of (D2). First of all it is clear that since real and complex numbers cannot be represented exactly on computers, (D2) implies that part of the business of numerical analysis must be to approximate them. This is where the rounding errors come in. Now for a certain set of problems, namely the ones that are solved by algorithms that take a finite number of steps, that is all there is to it. The premier example is Gaussian elimination for solving a linear system of equations $Ax = b$. To understand Gaussian elimination, you have to understand computer science issues such as operation counts and machine architectures, and you have to understand the propagation of rounding errors—stability. That's all you have to understand, and if somebody claims that (D2) is just a more polite restatement of (D1), you can't prove him or her wrong with the example of Gaussian elimination.

But most problems of continuous mathematics cannot be solved by finite algorithms! Unlike $Ax = b$, and unlike the discrete problems of computer science, most of the problems of numerical analysis could not be solved exactly even if we could work in exact arithmetic. Numerical analysts know this, and mention it along with a few words about Abel and Galois when they teach algorithms for computing matrix eigenvalues. Too often they forget to mention that the same conclusion extends to virtually any problem with a nonlinear term or a derivative in it—zerofinding, quadrature, differential equations, integral equations, optimization, you name it.

Even if rounding errors vanished, numerical analysis would remain. Approximating mere numbers, the task of floating-point arithmetic, is indeed a rather small topic and maybe even a tedious one. The deeper business of numerical analysis is approximating unknowns, not knowns. Rapid convergence of approximations is the aim, and the pride of our field is that, for many problems, we have invented algorithms that converge exceedingly fast.

These points are sometimes overlooked by enthusiasts of symbolic computing, especially recent converts, who are apt to think that the existence of Maple or Mathematica renders Matlab and Fortran obsolete. It is true that rounding errors can be made to vanish in the sense that in principle, any finite sequence of algebraic operations can be represented exactly on a computer by means of appropriate symbolic operations. Unless the problem being solved is a finite one, however, this only defers the inevitable approximations to the end of the calculation, by which point the quantities one is working with may have become extraordinarily cumbersome. Floating-point arithmetic is a name for numerical analysts' habit of doing their pruning at every step along the way of a calculation rather than in a single act at the end. Whichever way one proceeds, in floating-point or symbolically, the main problem of finding a rapidly convergent algorithm is the same.

In summary, it is a corollary of (D2) that numerical analysis is concerned with rounding errors and also with the deeper kinds of errors associated with convergence of approximations, which go by various names (truncation, discretization, iteration). Of course one could choose to make (D2) more explicit by adding words to describe these approximations and errors. But once words begin to be added it is hard to know where to stop, for (D2) also fails to mention some other important matters: that these algorithms are implemented on computers, whose architecture may be an important part of the problem; that reliability and efficiency are paramount goals; that some numerical analysts write programs and others prove theorems; and most important, that all of this work is *applied,* applied daily and successfully to thousands of applications on millions of computers around the world. "The problems of continuous mathematics" are the problems that science and engineering are built upon; without numerical methods, science and engineering as practiced today would come quickly to a halt. They are also the problems that preoccupied most mathematicians from the time of Newton to the twentieth century. As much as any pure mathematicians, numerical analysts are the heirs to the great tradition of Euler, Lagrange, Gauss and the rest. If Euler were alive today, he wouldn't be proving existence theorems.

* * *

Ten years ago, I would have stopped at this point. But the evolution of computing in the past decade has given the difference between (D1) and (D2) a new topicality.

Let us return to $Ax = b$. Much of numerical computation depends on linear algebra, and this highly developed subject has been the core of numerical analysis since the beginning. Numerical linear algebra served as the subject with respect to which the now standard concepts of stability, conditioning, and backward error analysis were defined and sharpened, and the central figure in these developments, from the 1950s to his death in 1986, was Jim Wilkinson.

I have mentioned that $Ax = b$ has the unusual feature that it can be solved in a finite sequence of operations. In fact, $Ax = b$ is more unusual than that, for the standard algorithm for solving it, Gaussian elimination, turns out to have extraordinarily complicated stability properties. Von Neumann wrote 180 pages of mathematics on this topic; Turing wrote one of his major papers; Wilkinson developed a theory that grew into two books and a career. Yet the fact remains that for certain $n \times n$ matrices, Gaussian elimination with partial pivoting amplifies rounding errors by a factor of order 2^n, making it a useless algorithm in the worst case. It seems that Gaussian elimination works in practice because the set of matrices with such behavior is vanishingly small, but to this day, nobody has a convincing explanation of why this should be so.

In manifold ways, then, Gaussian elimination is atypical. Few numerical algorithms have such subtle stability properties, and certainly no other was scrutinized in such depth by von Neumann, Turing, and Wilkinson. The effect? Gaussian elimination, which should have been a sideshow, lingered in the spotlight while our field was young and grew into the canonical algorithm of numerical analysis. Gaussian elimination set the agenda, Wilkinson set the tone, and the distressing result has been (D1).

Of course there is more than this to the history of how (D1) acquired currency. In the early years of computers, it was inevitable that arithmetic issues would receive concerted attention. Fixed-point computation required careful thought and novel hardware; floating-point computation arrived as a second revolution a few years later. Until these matters were well understood it was natural that arithmetic issues should be a central topic of numerical analysis, and, besides this, another force was at work. There is a general principle of computing that seems to have no name: *the faster the computer, the more important the speed of algorithms.* In the early years, with the early computers, the dangers of instability were nearly as great as they are today, and far less familiar. The gaps between fast and slow algorithms, however, were narrower.

A development has occurred in recent years that reflects how far we have come from that time. Instances have been accumulating in which, even though a finite algorithm exists for a problem, an infinite algorithm may be better. The distinction that seems absolute from a logical point of view turns out to have little importance in practice—and in fact, Abel and Galois notwithstanding, large-scale matrix eigenvalue problems are about as easy to solve in practice as linear systems of equations. For $Ax = b$, iterative methods are becoming more and more often the methods of choice as computers grow faster, matrices grow larger and less sparse (because of the advance from 2D to 3D simulations), and the $O(N^3)$ operation counts of the usual direct (= finite) algorithms become ever more painful. The name of the new game is *iteration with preconditioning.* Increasingly often it is not optimal to try to solve a problem exactly in one pass; instead, solve it approximately, then iterate. Multigrid methods, perhaps the most important development in numerical computation in the past twenty years, are based on a recursive application of this idea.

Even direct algorithms have been affected by the new manner of computing. Thanks to the work of Skeel and others, it has been noticed that the expense of making a direct method stable—say, of pivoting in Gaussian elimination—may in certain contexts be cost-ineffective. Instead, skip that step—solve the problem directly but unstably, then do one or two steps of iterative refinement. "Exact" Gaussian elimination becomes just another preconditioner!

Other problems besides $Ax = b$ have undergone analogous changes, and the famous example is linear programming. Linear programming problems are mathematically finite, and for decades, people solved them by a finite algorithm: the simplex method. Then Karmarkar announced in 1984 that iterative, infinite algorithms are sometimes better. The result has been controversy, intellectual excitement, and a perceptible shift of the entire field of linear programming away from the rather anomalous position it has traditionally occupied towards the mainstream of numerical computation.

I believe that the existence of finite algorithms for certain problems, together with other historical forces, has distracted us for decades from a balanced view of numerical analysis. Rounding errors and instability are important, and numerical analysts will always be the experts in these subjects and at pains to ensure that the unwary are not tripped up by them. But our central mission is to compute quantities that are typically uncomputable, from an analytical point of view, and to do it with lightning speed. For guidance to the future we should study not Gaussian elimination and its beguiling stability properties, but the diabolically fast conjugate gradient iteration—or Greengard and Rokhlin's $O(N)$ multipole algorithm for particle simulations—or the exponential convergence of spectral methods for solving certain PDEs—or the convergence in $O(1)$ iteration achieved by multigrid methods for many kinds of problems—or even Borwein and Borwein's magical AGM iteration for determining 1,000,000 digits of π in the blink of an eye. *That* is the heart of numerical analysis.

Notes

Many people, too numerous to name, provided comments on drafts of this essay. Their suggestions led me to many publications that I would otherwise not have found.

I do not claim that any of the ideas expressed here are entirely new. In fact, 30 years ago, in his *Elements of Numerical Analysis,* Peter Henrici defined numerical analysis as "the theory of constructive methods in mathematical analysis." Others have expressed similar views; Joseph Traub (*Communications of the ACM, 1972*), for example, defined numerical analysis as "the analysis of continuous algorithms." For that matter, both the Random House and the Oxford English dictionaries offer better definitions than the three quoted here.

And should the field be called "numerical analysis," "scientific computing," or something else entirely? ("mathematical engineering?"). That is another essay.