# FINITE DIFFERENCE AND SPECTRAL METHODS FOR ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

Lloyd N. Trefethen
Cornell University

## Note

A trap that academics sometimes fall into is to begin a book and fail to finish it. This has happened to me. This book was intended to be published several years ago. Alas, two other projects jumped the queue (*Numerical Linear Algebra,* with David Bau, to be published in 1997 by SIAM, and *Spectra and Pseudospectra,* to be completed thereafter). At this point I do not know whether this numerical ODE/PDE book will ever be properly finished.

This is a shame—as people keep telling me. Electronically by email, and in person at conferences, I am often asked "when that PDE book is going to get done." Anyone who has been in such a situation knows how flattering such questions feel… at first.

The book is based on graduate courses taught to mathematicians and engineers since 1982 at the Courant Institute, MIT, and Cornell. Successively more complete versions have been used by me in these courses half a dozen times, and by a handful of colleagues at these universities and elsewhere. Much of the book is quite polished, especially the first half, and there are numerous exercises that have been tested in the classroom. But there are many gaps too, and undoubtedly a few errors. I have reproduced here the version of the text I used most recently in class, in the spring of 1994.

I am proud of Chapter 1, which I consider as clean an introduction as any to the classical theory of the numerical solution of ODE. The other main parts of the book are Chapter 2, which emphasizes the crucial relationship between smoothness of a function and decay of its Fourier transform; Chapters 3 and 4, which present the classical theory of numerical stability for finite difference approximations to linear PDE; and Chapters 7 and 8, which offer a hands-on introduction to Fourier and Chebyshev spectral methods.

This book is largely linear, and for that reason and others, there is much more to the numerical solution of differential equations than you will find here. On the other hand, Chapters 1–4 represent material that every numerical analyst should know. These chapters alone can be the basis of an appealing course at the graduate level.

 Copies can be obtained for \$22 or £15 each (including shipping) by contacting me directly. Professors who are using the book in class may contact me to obtain solutions to most of the problems. [This is no longer true – the book is freely available online at `http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html`.]

Nick Trefethen
July 1996

Department of Computer Science and Center for Applied Mathematics
Upson Hall, Cornell University, Ithaca, NY 14853, USA
(607) 255-4222, fax (607) 255-4428, LNT@cs.cornell.edu

# Contents

# Preface

Many books have been written on numerical methods for partial differential equations, ranging from the mathematical classic by Richtmyer and Morton to recent texts on computational fluid dynamics. But this is not an easy field to set forth in a book. It is too active and too large, touching a bewildering variety of issues in mathematics, physics, engineering, and computer science.

My goal has been to write an advanced textbook focused on basic principles. Two remarkably fruitful ideas pervade this field: numerical stability and Fourier analysis. I want the reader to think at length about these and other fundamentals, beginning with simple examples, and build in the process a solid foundation for subsequent work on more realistic problems. Numerical methods for partial differential equations are a subject of subtlety and beauty, but the appreciation of them may be lost if one views them too narrowly as tools to be hurried to application.

This is not a book of pure mathematics. A number of theorems are proved, but my purpose is to present a set of ideas rather than a sequence of theorems. The book should be accessible to mathematically inclined graduate students and practitioners in various fields of science and engineering, so long as they are comfortable with Fourier analysis, basic partial differential equations, some numerical methods, the elements of complex variables, and linear algebra including vector and matrix norms. At MIT and Cornell, successive drafts of this text have formed the basis of a large graduate course taught annually since 1985.

One unusual feature of the book is that I have attempted to discuss the numerical solution of ordinary and partial differential equations in parallel. There are numerous ideas in common here, well known to professionals, but usually not stressed in textbooks. In particular I have made extensive use of the elegant idea of *stability regions* in the complex plane.

Another unusual feature is that in a single volume, this book treats spectral as well as the more traditional finite difference methods for discretization of partial differential equations. The solution of discrete equations by multigrid iterations, the other most conspicuous development in this field in the past twenty years, is also introduced more briefly.

The unifying theme of this book is Fourier analysis. To be sure, it is well known that Fourier analysis is connected with numerical methods, but here we shall consider at least four of these connections in detail, and point out analogies between them: stability analysis, spectral methods, iterative solutions of elliptic equations, and multigrid methods. Indeed, a reasonably accurate title would have been "Fourier Analysis and Numerical Methods for Partial Differential Equations." Partly because of this emphasis, the ideas covered here are primarily linear; this is by no means a textbook on computational fluid dynamics. But it should prepare the reader to read such books at high speed.

Besides these larger themes, a number of smaller topics appear here that are not so often found outside research papers, including:

- *Order stars* for stability and accuracy analysis,
- *Modified equations* for estimating discretization errors,
- Stability in $\ell^p$ *norms,*
- *Absorbing boundary conditions* for wave calculations,
- Stability analysis of non-normal discretizations via *pseudospectra,*
- *Group velocity* effects and their connection with the stability of boundary conditions.

Like every author, I have inevitably given greatest emphasis to those topics I know through my own work.

The reader will come to know my biases soon enough; I hope he or she shares them. One is that exercises are essential in any textbook, even if the treatment is advanced. (The exercises here range from cookbook to philosophical; those marked with solid triangles require computer programming.) Another is that almost every idea can and should be illustrated graphically. A third is that numerical methods should be studied at all levels, from the most formal to the most practical. This book emphasizes the formal aspects, saying relatively little about practical details, but the reader should forget at his or her peril that a vast amount of practical experience in numerical computation has accumulated over the years. This experience is magnificently represented in a wealth of high-quality mathematical software available around the world, such as EISPACK, LINPACK, LAPACK, ODEPACK, and the IMSL and NAG Boeing Fortran libraries, not to mention such interactive systems as Matlab and Mathematica.* Half of every applied problem can and should be handled by off-the-shelf programs nowadays. This book will tell you something about what those programs are doing, and help you to think productively about the other half.

*For on-line acquisition of numerical software, every reader of this book should be sure to be familiar with the "Netlib" facility maintained at Bell Laboratories and Oak Ridge National Laboratory. For information, send the e-mail message "help" to netlib@research.att.com or netlib@ornl.gov.

# Notation

| | |
|---|---|
| $\mathbb{R}, \mathbb{C}, \mathbb{Z}$ | real numbers, complex numbers, integers |
| $\star$ | marks a number contaminated by rounding errors |
| $O, o, \Omega, \Theta$ | order relations analogous to $\leq$, $<$, $\geq$, $=$ |
| $x, y, t$ | space and time variables |
| $\xi, \eta, \omega$ | $x$ wave number, $y$ wave number, frequency |
| $T$ | initial-value problem is posed for $t \in [0, T]$ |
| $N$ | dimension of system of equations |
| $d$ | number of space dimensions |
| $u$ | dependent variable (scalar or $N$-vector) |
| $h, k$ | space and time step sizes $h = \Delta x$, $k = \Delta t$ |
| $i, j, n$ | space and time indices |
| $\ell, r, s$ | integers defining size of finite difference stencil |
| $v_{ij}^n$ | numerical approximation to $u(ih, jh, nk)$ |
| $\hat{u}, \hat{v}$ | Fourier transforms |
| $\lambda, \sigma$ | mesh ratios $\lambda = k/h$, $\sigma = k^2/h$ |
| $K, Z$ | space and time shift operators $K\colon v_j \mapsto v_{j+1}$, $Z\colon v^n \mapsto v^{n+1}$ |
| $\kappa, z$ | space and time amplification factors $\kappa = e^{i\xi h}$, $z = e^{i\omega k}$ |
| $\Delta, \nabla$ | forward and backward difference operators $\Delta = K - 1$, $\nabla = 1 - K^{-1}$ |
| $\rho(z), \sigma(z)$ | characteristic polynomials for a linear multistep formula |
| $c, c_g$ | phase and group velocities $c = -\omega/\xi$, $c_g = -d\omega/d\xi$ |
| $\Lambda(A), \Lambda_\epsilon(A)$ | spectrum and $\epsilon$-pseudospectrum of $A$ |

The following little sections $\alpha$ and $\beta$ contain essential background material for the remainder of the text. They are not chapters, just tidbits. Think of them as appetizers.

# $\alpha$. Big-O and related symbols

How fast is an algorithm? How accurate are the results it produces? Answering questions like these requires estimating operation counts and errors, and we need a convenient notation for doing so. This book will use the six symbols $o$, $O$, $\Omega$, $\Theta$, $\sim$, and $\approx$. Four of these are standard throughout the mathematical sciences, while $\Omega$ and $\Theta$ have recently become standard at least in theoretical computer science.*

Here are the first four definitions. They are written for functions $f(x)$ and $g(x)$ as $x \to \infty$, but analogous definitions apply to other limits such as $x \to 0$.

$O$ $f(x) = O(g(x))$ as $x \to \infty$ if there exist constants $C > 0$ and $x_0 > 0$ such that $|f(x)| \le Cg(x)$ for all $x \ge x_0$. If $g(x) \ne 0$ this means $|f(x)|/g(x)$ is bounded from above for all sufficiently large $x$. In words: "$f$ is O of $g$."

$o$ $f(x) = o(g(x))$ as $x \to \infty$ if for *any* constant $C > 0$ there exists a constant $x_0 > 0$ such that $|f(x)| \le Cg(x)$ for all $x \ge x_0$. If $g(x) \ne 0$ this means $|f(x)|/g(x)$ approaches 0 as $x \to \infty$. In words: "$f$ is little-O of $g$."

$\Omega$ $f(x) = \Omega(g(x))$ as $x \to \infty$ if there exist constants $C > 0$ and $x_0 > 0$ such that $f(x) \ge Cg(x)$ for all $x \ge x_0$. If $g(x) \ne 0$ this means $f(x)/g(x)$ is bounded from below for all sufficiently large $x$. In words: "$f$ is omega of $g$."

$\Theta$ $f(x) = \Theta(g(x))$ as $x \to \infty$ if there exist constants $C, C' > 0$ and $x_0 > 0$ such that $Cg(x) \le f(x) \le C'g(x)$ for all $x \ge x_0$. If $g(x) \ne 0$ this means $f(x)/g(x)$ is bounded from above and below. In words: "$f$ is theta of $g$."

The use of the symbols $O$, $o$, $\Omega$, and $\Theta$ is a bit illogical, for properly speaking, $O(g(x))$ for example is the *set* of all functions $f(x)$ with a certain property; but why then don't we write $f(x) \in O(g(x))$? The answer is simply that the traditional usage, illogical or not, is well established and very convenient.

The final two definitions are not so irregular:

* Thanks to the delightful note "Big omicron and big omega and big theta" by D. E. Knuth, *ACM SIGACT News 8*, 1976.

- $\sim$ $f(x) \sim g(x)$ as $x \to \infty$ if $f(x) - g(x) = o(|g(x)|)$. If $g(x) \neq 0$ this is equivalent to $\lim_{x\to\infty} f(x)/g(x) = 1$. In words: "$f$ is asymptotic to $g$."

- $\approx$ This symbol is easy: it has no precise definition! The statement $f(x) \approx g(x)$ is qualitative only, and the argument $x$ plays no role. Often in this book, but not always, we shall use $\approx$ to indicate that a real number has been rounded or chopped to a finite number of digits, e.g., $\pi \approx 3$ or $\pi \approx 3.14159265358979323846$.

**EXAMPLE $\alpha$.1.** True statements as $x \to \infty$: $\sin x = O(1)$, $\log x = o(x^{1/100})$, $e^x = \Omega(x^{100})$, $2 + \sin^2 x = \Theta(1)$, $\cosh x = e^x + o(1) \sim e^x$. False statements as $x \to \infty$: $x = o(100x)$, $\sin x = \Omega(1)$, $e^{-x} \sim 0$.

The relationship between these definitions can be summarized by the Venn diagram of Figure $\alpha$.1. Let $X$ denote the set of all pairs of functions $\{f, g\}$. The diagram illustrates, for example, that the subset of $X$ consisting of pairs $\{f, g\}$ with $f(x) = o(g(x))$ is contained in the subset of pairs with $f(x) = O(g(x))$. In other words, $f(x) = o(g(x))$ implies $f(x) = O(g(x))$.

**Figure $\alpha$.1.** Venn diagram showing the relationship between $o$, $O$, $\Omega$, and $\Theta$.

## EXERCISES

▷ *$\alpha$.1. True or False?*

*(a)* $x^2 \sin(1/x) \sim x^2$ as $x \to 0$.

*(b)* $x^2 \sin(1/x) \sim x$ as $x \to \infty$.

*(c)* $(\sin x)^2 \sim \sin(x^2)$ as $x \to 0$.

*(d)* $(\log x)^2 = o(\log(x^2))$ as $x \to \infty$.

*(e)* $n! = \Theta(n/e)^n$ as $n \to \infty$.

▷ *$\alpha$.2.* Suppose $g(x) \sim 1$ as $x \to \infty$. Find interesting examples of functions $f \neq g$ occupying all five distinct positions in the Venn diagram of Figure $\alpha$.1.

▷ *$\alpha$.3. True or False?*

*(a)* $A = \Theta(V^{2/3})$ as $V \to \infty$, where $A$ and $V$ are the surface area and volume of a sphere measured in square miles and cubic microns, respectively.

*(b)* $\lim_{n\to\infty} a_n = b \iff a_n = b + o(1)$ as $n \to \infty$.

*(c)* A set of $N$ words can be sorted into alphabetical order by an algorithm that makes $O(N \log N)$ pairwise comparisons.

*(d)* Solving an $N \times N$ matrix problem $Ax = b$ by Gaussian elimination on a serial computer takes $\Omega(N^3)$ operations.

# $\beta$. Rounding and truncation errors

Throughout this book we shall be concerned with computed, inexact approximations to ideal quantities. Two things keep them from being exact: rounding errors and truncation errors.

**Rounding errors** are the errors introduced by computers in simulating real or complex arithmetic by floating-point arithmetic.* For most purposes they can be treated as ubiquitous and random, and it is possible to be quite precise about how they behave. Given a particular machine with a particular system of floating-point arithmetic, let **machine epsilon** be defined as the smallest positive floating-point number $\epsilon$ such that $1 \oplus \epsilon > 1$. Here $\oplus$ denotes floating-point addition of floating-point numbers; analogous notations apply for $-$, $\times$, and $\div$. Then on well-designed computers, if $a$ and $b$ are arbitrary floating-point numbers, the following identities hold:

$$\left.\begin{aligned} a \oplus b &= (a+b)(1+\delta) \\ a \ominus b &= (a-b)(1+\delta) \\ a \otimes b &= (a\times b)(1+\delta) \\ a \oslash b &= (a\div b)(1+\delta) \end{aligned}\right\} \text{ for some } \delta \text{ with } |\delta| < \epsilon; \qquad (\beta.1)$$

the quantity $\delta$ is of course different from one appearance to the next. In other words, each floating-point operation introduces a *relative error* of magnitude less than $\epsilon$.†

**Truncation errors**, or **discretization errors**, are the errors introduced by algorithms in replacing an infinite or infinitesimal quantity by something finite; they have nothing to do with floating-point arithmetic. For example, the error introduced in truncating an infinite series at some finite term $N$ is a truncation error, and so is the error introduced in replacing a derivative $du/dt$ by the finite difference $[u(t+\Delta t)-u(t-\Delta t)]/2\Delta t$. Truncation errors do not appear in all areas of numerical computation; they are absent, for example, in solving a linear system of equations by Gaussian elimination. They are usually unavoidable, however, in the numerical solution of differential equations. Unlike rounding errors, truncation errors are generally not at all random, but may have a great deal of regularity.

Rounding errors would vanish if computers were infinitely accurate. Truncation errors would vanish if computers were infinitely fast and had infinitely large memories, so that there were no need to settle for finite approximations. Further discussion of this distinction can

---

***Floating-point numbers** are real numbers of the form $x = \pm f \times \beta^e$, where $\beta$ is the **base** (usually 2, 10, or 16), $f < 1$ is the **fraction** represented to $t$ digits in base $\beta$, and $e$ is an adjustable **exponent**. **Floating-point arithmetic** refers to the use of floating-point numbers together with approximate arithmetic operations defined upon them. Early computers sometimes used "fixed-point arithmetic" instead, but this is unheard-of nowadays except in special-purpose machines. For an introduction to floating-point arithmetic see G. B. Forsythe, M. A. Malcolm & C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, 1977, and for a table of the floating-point characteristics of existing computers as of 1988, see W. J. Cody, *ACM Trans. Math. Softw. 14* (1988), 303–311.

†This description ignores the possibility of underflow or overflow.

be found in many books on numerical analysis, such as P. Henrici, *Essentials of Numerical Analysis*, Wiley, 19??.

In this book we shall frequently ask how a particular computation may be affected by rounding or truncation errors. Which source of error dominates? Most often, rounding errors are negligible and the accuracy is determined by truncation errors. When an instability is present, however, the situation is sometimes reversed. We shall pay careful attention to such matters, for there is no other way to understand mathematical computations fully.

Truncation errors depend only on the algorithm, but rounding errors depend also on the machine. We shall use the following notation throughout the book: a number preceded by a star $^\star$ is significantly contaminated by rounding errors and hence machine-dependent.

---

**EXAMPLE $\beta$.1.** For example, suppose we approximate $e^x$ by the 101-term series

$$f(x) = 1+x+\frac{x^2}{2}+\cdots+\frac{x^{100}}{100!},$$

evaluated on the computer from left to right as written. On a Sun Workstation in double precision with $\epsilon = 2^{-52} \approx 2.22\times 10^{-16}$, we obtain

$$e^{20} \approx 4.8517\times 10^{8}, \qquad f(20) \approx 4.8517\times 10^{8},$$

$$e^{100} \approx 2.6881\times 10^{43}, \qquad f(100) \approx 1.4155\times 10^{43},$$

$$e^{-20} \approx 2.0612\times 10^{-9}, \qquad f(-20) \approx {}^\star 5.6219\times 10^{-9}.$$

Thus $f(20)$ is accurate, but $f(100)$ is inaccurate because of truncation errors, and $f(-20)$ is inaccurate because of rounding errors. (Why? See Exercise $\beta$.2.) This straightforward summation of the Taylor series for $e^x$ is a classic example of a bad algorithm: it is sometimes unstable and always inefficient, unless $|x|$ is small.

---

Some people consider the study of rounding errors a tedious business, and there are those who assume that all of numerical analysis must be tedious, for isn't it mainly the study of rounding errors? Here are three reasons why these views are unjustified.

First, thanks to the simple formulas ($\beta$.1), the modeling of rounding errors is easy and even elegant. Rounding errors are by no means capricious or unpredictable, and one can understand them well without worrying about machine-dependent engineering details.

Second, the existence of inexact arithmetic should be blamed not on computers, but on mathematics itself. It has been known for a long time that many well-defined quantities cannot be calculated exactly by any finite sequence of elementary operations; the classic example is the "unsolvability" of the roots of polynomials of degree $\geq 5$. Thus approximations are unavoidable even in principle.*

---

* Mathematics also rests on approximations at a more fundamental level. The set of real numbers is uncountably infinite, but as was first pointed out by Turing in his famous paper on "Computable

Finally and most important, numerical analysis is not just the study of rounding errors! Here is a better definition:

*Numerical analysis is the study of algorithms*
*for mathematical problems involving continuous variables.*

Thus it is a field devoted to *algorithms,* in which the basic questions take the form, "What is the best algorithm for computing such-and-such?" Insensitivity to rounding errors is only a part of what makes a numerical algorithm good; speed, for example, is certainly as important.

**EXERCISES**

▶ *$\beta$.1.* Figure out machine epsilon exactly on your calculator or computer. What are the base $\beta$ and the precision $t$? Explain your reasoning carefully.

▷ *$\beta$.2.* *(Continuation of Example $\beta$.1.)*

*(a)* Explain the mechanism by which rounding errors affected the computation of $f(-20)$.
*(b)* If we compute $f(-100)$ in the same way, will there be a star in front of the result?
*(c)* Suggest an algorithm for calculating $e^x$ more quickly and accurately in floating-point arithmetic.

---

Numbers" in 1937, only a countable subset of them can ever be specified by any finite description. The vast majority of real numbers can only be described in effect by listing their infinite decimal expansions, which requires an infinite amount of time. Thus most real numbers are not merely unrepresentable on computers; they are unmentionable even by pure mathematicians, and exist only in a most shadowy sense.