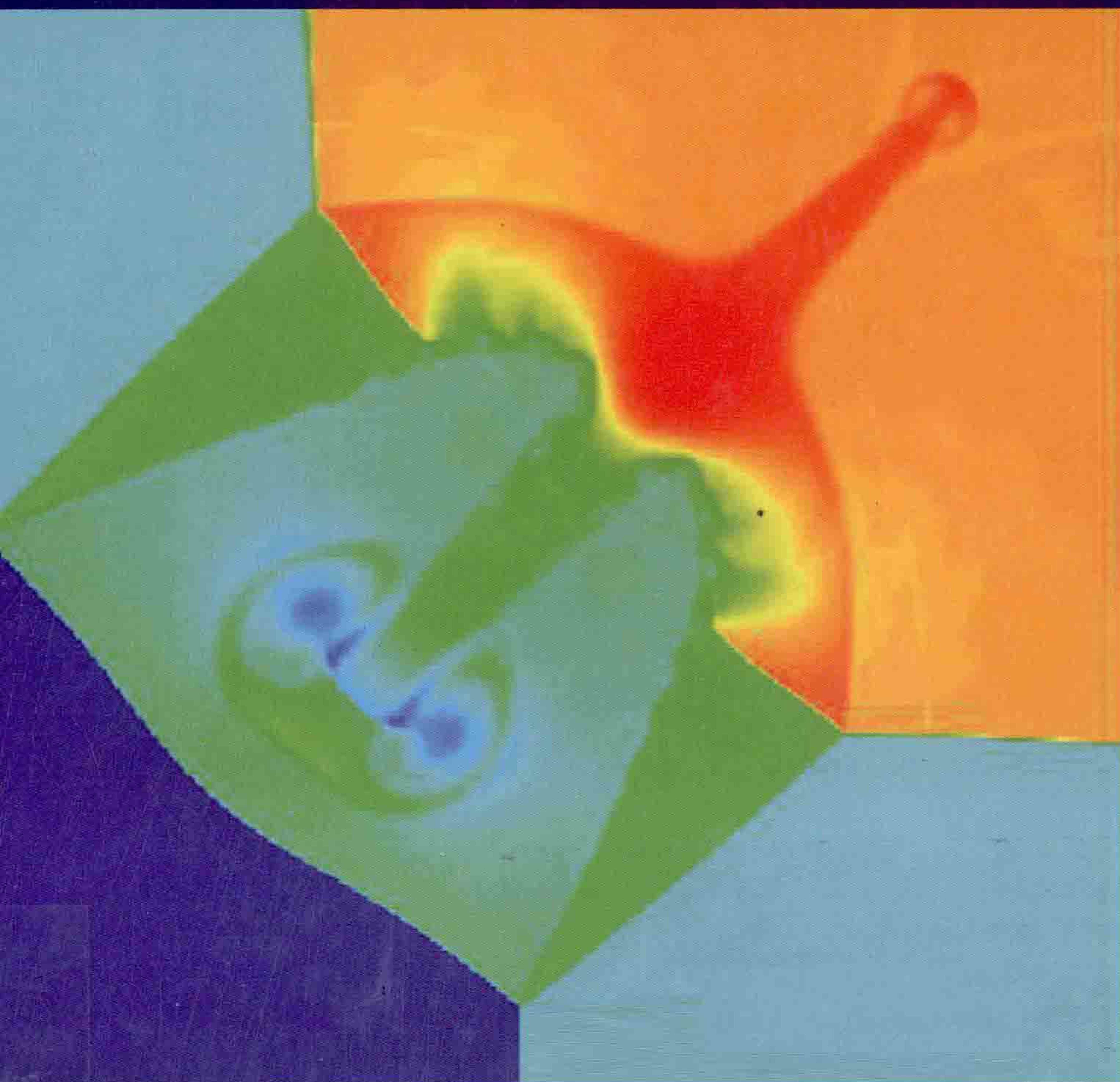
Numerical Solution of Hyperbolic Partial Differential Equations



John A. Trangenstein

CAMBRIDGE

NUMERICAL SOLUTION OF HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS

JOHN A. TRANGENSTEIN

Department of Mathematics, Duke University
Durham, NC 27708-0320

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Pre

www.cambridge.org
Information on this title: www.cambridge.org/978052187727

First published 2009

Printed in the United Kingdom at the University Press, Cambr

A catalog record for this publication is available from the British

ISBN 978-0-521-87727-5 Hardback

NUMERICAL SOLUTION OF HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS

This is a new type of graduate textbook, with both print and interactive electronic cor ponents (on CD). It is a comprehensive presentation of modern shock-capturing methoc including both finite volume and finite element methods, covering the theory of hyperbol conservation laws and the theory of the numerical methods.

Classical techniques for judging the qualitative performance of the schemes, such modified equation analysis and Fourier analysis, are used to motivate the development classical higher-order methods (the Lax–Wendroff process) and to prove results such as tl Lax Equivalence Theorem.

The range of applications (shallow water, compressible gas dynamics, magnetohydr dynamics, finite deformation in solids, plasticity, polymer flooding and water/gas injecti in oil recovery) is broad enough to engage most engineering disciplines and many areas applied mathematics.

The solution of the Riemann problems for these applications is developed, so that tl reader can use the theory to develop test problems for the methods, especially to me sure errors for comparisons of accuracy and efficiency. The numerical methods invol a variety of important approaches, such as MUSCL and PPM, TVD, wave propagatio Lax–Friedrichs (aka central schemes), ENO and discontinuous Galerkin; all of these a discussed in o ...d multiple spatial dimensions. Since many of these methods depend Riemann solvers, there is extensive discussion of the basic design principles of approxima Riemann solvers, and several computationally useful techniques. The final chapter contai a discussion of adaj ve mesh refinement via structured grids.

The accompanying CD contains a hyperlinked version of the text which provides acce to computer codes for all of the text figures. Through this electronic text students can:

- See the codes and run them, choosing their own input parameters interactively
- View the online numerical results as movies
- Gain an appreciation for both the dynamics of the problem application, and the grow of numerical errors
- Download and modify the code for use with other applications
- Study the code to learn how to structure their programs for modularity and ease debugging

JOHN A. TRANGENSTEIN is Professor of Mathematics at Duke University, North Carolin

To James A. Rowe

Preface

Hyperbolic conservation laws describe a number of interesting physical probler in diverse areas such as fluid dynamics, solid mechanics, and astrophysics. O emphasis in this book is on nonlinearities in these problems, especially those tl lead to the development of propagating discontinuities. These propagating discon nuities can appear as the familiar shock waves in gases (the "boom" from explosio or super-sonic airplanes), but share many mathematical properties with other wav that do not appear to be so "shocking" (such as steep changes in oil saturations petroleum reservoirs). These nonlinearities require special treatment, usually l methods that are themselves nonlinear. Of course, the numerical methods in tl book can be used to solve linear hyperbolic conservation laws, but our methods w not be as fast or accurate as possible for these problems. If you are only interest in *linear* hyperbolic conservation laws, you should read about spectral methods a multipole expansions.

This book grew out of a one-semester course I have taught at Duke Universi over the past decade. Quite frankly, it has taken me at least 10 years to develop t material into a form that I like. I may tinker with the material more in the futu because I expect that I will never be fully satisfied.

I have designed this book to describe both numerical methods and their applic tions. As a result, I have included substantial discussion about the analytical soluti of hyperbolic conservation laws, as well as discussion about numerical methods. this course, I have tried to cover the applications in such a way that the engineeri students can see the mathematical structure that is common to all of these proble areas. With this information, I hope that they will be able to adapt new numeric methods developed for other problem areas to their own applications. I try to get t mathematics students to adopt one of the physical models for their computatio during the semester, so that the numerical experiments can help them to devel physical intuition.

I also tried to discuss a variety of numerical methods in this text, so th could see a number of competing ideas. This book does not try to fav particular numerical scheme, and it does not serve as a user manual to package. It does have software available, to allow the reader to experi the various ideas. But the software is not designed for easy applicati problems. Instead, I hope that the readers will learn enough from this bo intelligent decisions on which scheme is best for their problems, as wel implement that scheme efficiently.

There are a number of very good books on related topics. LeVeq *Volume Methods for Hyperbolic Problems* [97] is one that covers the m well, describes several important numerical methods, but emphasize propagation scheme over all. Other books are specialized for particul areas, such as Hirsch's *Numerical Computation of Internal and Exte* [73], Peyret and Taylor's *Computational Methods for Fluid Flow* [131] *Computational Fluid Dynamics* [137] and Toro's *Riemann Solvers and Methods for Fluid Dynamics* [159]. These books contain very interesting that are particular for fluid dynamics, and should not be ignored.

Because this text develops analytical solutions to several problems, it to measure the errors in the numerical methods on interesting test pro relates to a point I try to emphasize in teaching the course, that it is e numerical computation to perform mesh refinement studies in order to that the method is performing properly. Another topic in this text is that methods can be compared for accuracy (error for a given mesh size) and (error for a given amount of computational time). Sometimes people hav bias toward higher-order methods, but this may not be the most co approach for many problems. Efficiency is tricky to measure, because gramming issues can drive up computational time. I do not claim to hav the most efficient version of any of the schemes in this text, so the effici parisons should be taken "with a grain of salt."

The numerical comparisons produced some surprises for me. For was surprised that approximate Riemann problem solvers often proc numerical results in Godunov methods than "exact" Riemann solver surprise is that there is no clear best scheme or worst scheme in this tex I have omitted discussions of schemes that have fallen out of favor in th for good reasons). There are some schemes that generally work better and some that often are less efficient than most, but all schemes have in which they perform well. The journal literature, of course, is full o of the latter behavior, since the authors get to choose computational exa benefit their method.

During the past ten years, I have watched numerical methods evolve, compute gain amazing speed, and students struggle harder with programming. The evolı tion of the methods led me to develop the course material into a form that studen could access online. In that way, I could insert additional text for ready access b the students. The speed of current desktop machines allows us to make some re sonably interesting computations during the semester, seeing in a few minutes wh used to require overnight runs on supercomputers. During that time, however, th new operating systems have separated the students ever farther from programmin details.

As I gained experience with online text generation, I started to ask if it woul be possible to develop an interactive text. First, I wanted students to be able t view the example programs while they were reading the text online. Next, I wante students to be able to examine links to information available on the web. Then, decided that it would be really nice if students could perform "what if" experimen within the text, by running numerical methods with different parameters and seein the results immediately. Because I continue to think that only "real" programmin languages (*i.e.*, C, C^{++} and Fortran) should be used for the material such as this, rejected suggestions that I rewrite the programs in Matlab or Java. Eventually, o department systems programmer, Andrew Schretter, found a way to make thin work for me, provided that I arrange for all parameter entry through graphic user interfaces. Our senior systems programmer, Yunliang Yu, did a lot of th development of the early form of the graphical user interface. One of my form graduate students, Wenjun Ying, programmed carefully the many cases for th marching cubes algorithm for visualizing level surfaces in three dimensions. I a greatly indebted to Andrew, Wenjun and Yunliang for their help.

This text is being published in two forms: traditional paper copy and a PDF file c a companion CD. The electronic form of the text contains links between equation theorem references and the original statements. Similar links lead to bibliograpl citations or to occurrences of key words in the index. There are electronic linl in the online text to source code and executables on the CD. This allows studen to view computer implementations of the algorithms developed in the book, ar to perform "what if" experiments with program and model parameters. Howeve since the text is the same for both versions of the book, this means that the pap text contains instructions to click on electronic links.

The graphical user interface (GUI) makes it easy for students to change parar eters (and, in fact, to see all of the input parameters). The GUI also complicat the online programs. There is a danger that students may think that they have program GUIs in order to solve these problems. That is not my intent. I have pr vided several example programs in the online version of chapter 2 to show studen how they can write simple programs (that produce data sets for post p or slightly more complex programs (that display numerical results durin putation to look like movies), or very sophisticated programs (that use input parameters). I would be happy if all students could program succ the first style. After all, CLAWPACK is a very successful example of t and direct style of programming.

It is common that students in this class are taking it in order to learn pro in Fortran or C^{++}, as much as they want to learn about the numerical met of these languages have advantages and disadvantages. Fortran is very arrays (subscripts can start at arbitrary values, which is useful for "gho many methods) and has a very large set of intrinsic functions (for exa and min with more than two arguments for slope limiters). Fortran i good with memory allocation, or with pointers in general. I use C^{++} to memory allocation, and for all interactive graphics, including GUIs. W select numerical methods through a GUI, then I set values for function p pass those as arguments to Fortran routines. I do not recommend sucl for novice programmers. On the other hand, students who want to ex programming skills can find several interesting techniques in the codes.

I do try to emphasize **defensive programming** when I teach courses tl scientific computing. By this term, I mean the use of programming pra make it easier to prevent or identify programming errors. It is often catch the use of uninitialized variables, the access of memory out of memory leaks. The mixed-language programs all use the following defer First, floating-point traps are enabled in unoptimized code. Second, flo array values are initialized to IEEE infinity. Third, a memory debugger memory allocation by overloading `operator new` in C^{++}. When th makes an allocation request, the memory debugger gets even more spac heap, and puts special bit patterns into the space before and after the user m a result, the programmer can ask the memory debugger to check individu or all pointers for writes out of bounds. This memory debugger is very fas not add significantly to the overall memory requirements. The memory also informs the programmer about memory leaks, providing informa where the unfreed pointer was allocated.

Unfortunately, mixing Fortran and C^{++} allows the possibility of tr programming errors. For example, declaring a Fortran subroutine to ha value in a C^{++} `extern "C"` block can lead to stack corruption. I d good defensive programming technique for that error.

But this book is really about numerical methods, not programming interested in hyperbolic conservation laws well after graduate school indebted to several people for helping me to develop that interest. Joh

Gregory Shubin were particularly helpful when we worked together at Exxo Production Research. At Lawrence Livermore National Laboratory, I learnt muc about Godunov methods from both John Bell and Phil Colella, and about objec oriented programming from Bill Crutchfield and Mike Welcome. I want to than all of them for their kind assistance during our years together.

Finally, emotional support throughout a project of this sort is essential. I wai to thank my wife, Becky, for all her love and understanding throughout our yeai together. I could not have written this book without her.

Contents