

Cambridge University Press

978-0-521-87727-5 - Numerical Solution of Hyperbolic Partial Differential Equations

John A. Trangenstein

Frontmatter

[More information](#)

NUMERICAL SOLUTION OF HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS

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

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

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

The solution of the Riemann problems for these applications is developed, so that the reader can use the theory to develop test problems for the methods, especially to measure errors for comparisons of accuracy and efficiency. The numerical methods involve a variety of important approaches, such as MUSCL and PPM, TVD, wave propagation, Lax–Friedrichs (aka central schemes), ENO and discontinuous Galerkin; all of these are discussed in one and multiple spatial dimensions. Since many of these methods depend on Riemann solvers, there is extensive discussion of the basic design principles of approximate Riemann solvers, and several computationally useful techniques. The final chapter contains a discussion of adaptive mesh refinement via structured grids.

The accompanying CD contains a hyperlinked version of the text which provides access 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 growth 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 of debugging.

John A. Trangenstein is Professor of Mathematics at Duke University, North Carolina

Cambridge University Press

978-0-521-87727-5 - Numerical Solution of Hyperbolic Partial Differential Equations

John A. Trangenstein

Frontmatter

[More information](#)

NUMERICAL SOLUTION OF
HYPERBOLIC PARTIAL
DIFFERENTIAL EQUATIONS

JOHN A. TRANGENSTEIN

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



Cambridge University Press
978-0-521-87727-5 - Numerical Solution of Hyperbolic Partial Differential Equations
John A. Trangenstein
Frontmatter
[More information](#)

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, São Paulo
Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK
Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521877275

© John A. Trangenstein 2007

This publication is in copyright. Subject to statutory exception
and to the provisions of relevant collective licensing agreements,
no reproduction of any part may take place without
the written permission of Cambridge University Press.

First published 2007

Printed in the United Kingdom at the University Press, Cambridge

A catalog record for this publication is available from the British Library

ISBN 978-0-521-87727-5 hardback

Cambridge University Press has no responsibility for the persistence or
accuracy of URLs for external or third-party internet websites referred to
in this publication, and does not guarantee that any content on such
websites is, or will remain, accurate or appropriate.

All material contained within the CD-ROM is protected by copyright and other intellectual property laws.
The customer acquires only the right to use the CD-ROM and does not acquire any other rights,
express or implied, unless these are stated explicitly in a separate licence.

To the extent permitted by applicable law, Cambridge University Press is not liable for direct damages
or loss of any kind resulting from the use of this product or from errors or faults contained in it,
and in every case Cambridge University Press's liability shall be limited to the amount
actually paid by the customer for the product.

Cambridge University Press

978-0-521-87727-5 - Numerical Solution of Hyperbolic Partial Differential Equations

John A. Trangenstein

Frontmatter

[More information](#)

To James A. Rowe

Contents

<i>Preface</i>	<i>page ix</i>
1 Introduction to Partial Differential Equations	1
2 Scalar Hyperbolic Conservation Laws	6
2.1 Linear Advection	6
2.1.1 Conservation Law on an Unbounded Domain	6
2.1.2 Integral Form of the Conservation Law	8
2.1.3 Advection–Diffusion Equation	9
2.1.4 Advection Equation on a Half-Line	10
2.1.5 Advection Equation on a Finite Interval	11
2.2 Linear Finite Difference Methods	12
2.2.1 Basics of Discretization	12
2.2.2 Explicit Upwind Differences	14
2.2.3 Programs for Explicit Upwind Differences	16
2.2.3.1 First Upwind Difference Program	16
2.2.3.2 Second Upwind Difference Program	17
2.2.3.3 Third Upwind Difference Program	18
2.2.3.4 Fourth Upwind Difference Program	20
2.2.3.5 Fifth Upwind Difference Program	21
2.2.4 Explicit Downwind Differences	23
2.2.5 Implicit Downwind Differences	24
2.2.6 Implicit Upwind Differences	25
2.2.7 Explicit Centered Differences	26
2.3 Modified Equation Analysis	30
2.3.1 Modified Equation Analysis for Explicit Upwind Differences	30

2.3.2	Modified Equation Analysis for Explicit Downwind Differences	31
2.3.3	Modified Equation Analysis for Explicit Centered Differences	32
2.3.4	Modified Equation Analysis Literature	33
2.4	Consistency, Stability and Convergence	35
2.5	Fourier Analysis of Finite Difference Schemes	38
2.5.1	Constant Coefficient Equations and Waves	39
2.5.2	Dimensionless Groups	40
2.5.3	Linear Finite Differences and Advection	41
2.5.4	Fourier Analysis of Individual Schemes	44
2.6	L^2 Stability for Linear Schemes	53
2.7	Lax Equivalence Theorem	55
2.8	Measuring Accuracy and Efficiency	69
3	Nonlinear Scalar Laws	81
3.1	Nonlinear Hyperbolic Conservation Laws	81
3.1.1	Nonlinear Equations on Unbounded Domains	81
3.1.2	Characteristics	82
3.1.3	Development of Singularities	84
3.1.4	Propagation of Discontinuities	85
3.1.5	Traveling Wave Profiles	89
3.1.6	Entropy Functions	92
3.1.7	Oleinik Chord Condition	95
3.1.8	Riemann Problems	97
3.1.9	Galilean Coordinate Transformations	99
3.2	Case Studies	102
3.2.1	Traffic Flow	102
3.2.2	Miscible Displacement Model	103
3.2.3	Buckley–Leverett Model	105
3.3	First-Order Finite Difference Methods	111
3.3.1	Explicit Upwind Differences	111
3.3.2	Lax–Friedrichs Scheme	112
3.3.3	Timestep Selection	117
3.3.4	Rusanov’s Scheme	118
3.3.5	Godunov’s Scheme	120
3.3.6	Comparison of Lax–Friedrichs, Godunov and Rusanov	124
3.4	Nonreflecting Boundary Conditions	125
3.5	Lax–Wendroff Process	129
3.6	Other Second Order Schemes	132

	<i>Contents</i>	ix
4 Nonlinear Hyperbolic Systems		135
4.1 Theory of Hyperbolic Systems		135
4.1.1 Hyperbolicity and Characteristics		135
4.1.2 Linear Systems		139
4.1.3 Frames of Reference		140
4.1.3.1 Useful Identities		141
4.1.3.2 Change of Frame of Reference for Conservation Laws		143
4.1.3.3 Change of Frame of Reference for Propagating Discontinuities		145
4.1.4 Rankine–Hugoniot Jump Condition		146
4.1.5 Lax Admissibility Conditions		150
4.1.6 Asymptotic Behavior of Hugoniot Loci		152
4.1.7 Centered Rarefactions		156
4.1.8 Riemann Problems		159
4.1.9 Riemann Problem for Linear Systems		159
4.1.10 Riemann Problem for Shallow Water		162
4.1.11 Entropy Functions		164
4.2 Upwind Schemes		176
4.2.1 Lax–Friedrichs Scheme		176
4.2.2 Rusanov Scheme		179
4.2.3 Godunov Scheme		179
4.3 Case Study: Maxwell’s Equations		183
4.3.1 Conservation Laws		183
4.3.2 Characteristic Analysis		184
4.4 Case Study: Gas Dynamics		186
4.4.1 Conservation Laws		187
4.4.2 Thermodynamics		187
4.4.3 Characteristic Analysis		188
4.4.4 Entropy Function		190
4.4.5 Centered Rarefaction Curves		192
4.4.6 Jump Conditions		194
4.4.7 Riemann Problem		200
4.4.8 Reflecting Walls		205
4.5 Case Study: Magnetohydrodynamics (MHD)		208
4.5.1 Conservation Laws		208
4.5.2 Characteristic Analysis		209
4.5.3 Entropy Function		218
4.5.4 Centered Rarefaction Curves		218
4.5.5 Jump Conditions		220

4.6	Case Study: Finite Deformation in Elastic Solids	221
4.6.1	Eulerian Formulation of Equations of Motion for Solids	221
4.6.2	Lagrangian Formulation of Equations of Motion for Solids	222
4.6.3	Constitutive Laws	223
4.6.4	Conservation Form of the Equations of Motion for Solids	225
4.6.5	Jump Conditions for Isothermal Solids	226
4.6.6	Characteristic Analysis for Solids	227
4.7	Case Study: Linear Elasticity	233
4.8	Case Study: Vibrating String	235
4.8.1	Conservation Laws	235
4.8.2	Characteristic Analysis	237
4.8.3	Jump Conditions	238
4.8.4	Lax Admissibility Conditions	240
4.8.5	Entropy Function	240
4.8.6	Wave Families for Concave Tension	241
4.8.7	Wave Family Intersections	245
4.8.8	Riemann Problem Solution	249
4.9	Case Study: Plasticity	255
4.9.1	Lagrangian Equations of Motion	255
4.9.2	Constitutive Laws	256
4.9.3	Centered Rarefactions	258
4.9.4	Hugoniot Loci	259
4.9.5	Entropy Function	261
4.9.6	Riemann Problem	261
4.10	Case Study: Polymer Model	267
4.10.1	Constitutive Laws	268
4.10.2	Characteristic Analysis	269
4.10.3	Jump Conditions	270
4.10.4	Riemann Problem Solution	271
4.11	Case Study: Three-Phase Buckley–Leverett Flow	274
4.11.1	Constitutive Models	274
4.11.2	Characteristic Analysis	276
4.11.3	Umbilic Point	277
4.11.4	Elliptic Regions	277
4.12	Case Study: Schaeffer–Schechter–Shearer System	278
4.13	Approximate Riemann Solvers	284
4.13.1	Design of Approximate Riemann Solvers	284
4.13.2	Artificial Diffusion	291
4.13.3	Rusanov Solver	293
4.13.4	Weak Wave Riemann Solver	294

<i>Contents</i>	xi
4.13.5 Colella–Glaz Riemann Solver	296
4.13.6 Osher–Solomon Riemann Solver	298
4.13.7 Bell–Colella–Trangenstein Approximate Riemann Problem Solver	299
4.13.8 Roe Riemann Solver	304
4.13.9 Harten–Hyman Modification of the Roe Solver	313
4.13.10 Harten–Lax–van Leer Scheme	315
4.13.11 HLL Solvers with Two Intermediate States	317
4.13.12 Approximate Riemann Solver Recommendations	320
5 Methods for Scalar Laws	326
5.1 Convergence	326
5.1.1 Consistency and Order	326
5.1.2 Linear Methods and Stability	328
5.1.3 Convergence of Linear Methods	330
5.2 Entropy Conditions and Difference Approximations	331
5.2.1 Bounded Convergence	331
5.2.2 Monotone Schemes	341
5.3 Nonlinear Stability	353
5.3.1 Total Variation	353
5.3.2 Total Variation Stability	354
5.3.3 Other Stability Notions	357
5.4 Propagation of Numerical Discontinuities	359
5.5 Monotonic Schemes	361
5.5.1 Smoothness Monitor	361
5.5.2 Monotonizations	362
5.5.3 MUSCL Scheme	364
5.6 Discrete Entropy Conditions	367
5.7 E-Schemes	368
5.8 Total Variation Diminishing Schemes	370
5.8.1 Sufficient Conditions for Diminishing Total Variation	370
5.8.2 Higher-Order TVD Schemes for Linear Advection	375
5.8.3 Extension to Nonlinear Scalar Conservation Laws	379
5.9 Slope-Limiter Schemes	383
5.9.1 Exact Integration for Constant Velocity	384
5.9.2 Piecewise Linear Reconstruction	386
5.9.3 Temporal Quadrature for Flux Integrals	388
5.9.4 Characteristic Tracing	389
5.9.5 Flux Evaluation	390
5.9.6 Non-Reflecting Boundaries with the MUSCL Scheme	391

xii	<i>Contents</i>	
5.10	Wave Propagation Slope Limiter Schemes	391
5.10.1	Cell-Centered Wave Propagation	391
5.10.2	Side-Centered Wave Propagation	394
5.11	Higher-Order Extensions of the Lax–Friedrichs Scheme	395
5.12	Piecewise Parabolic Method	402
5.13	Essentially Non-Oscillatory Schemes	408
5.14	Discontinuous Galerkin Methods	412
5.14.1	Weak Formulation	412
5.14.2	Basis Functions	413
5.14.3	Numerical Quadrature	414
5.14.4	Initial Data	415
5.14.5	Limiters	416
5.14.6	Timestep Selection	417
5.15	Case Studies	418
5.15.1	Case Study: Linear Advection	418
5.15.2	Case Study: Burgers’ Equation	422
5.15.3	Case Study: Traffic Flow	426
5.15.4	Case Study: Buckley–Leverett Model	427
6	Methods for Hyperbolic Systems	432
6.1	First-Order Schemes for Nonlinear Systems	432
6.1.1	Lax–Friedrichs Method	432
6.1.2	Random Choice Method	433
6.1.3	Godunov’s Method	433
6.1.3.1	Godunov’s Method with the Rusanov Flux	434
6.1.3.2	Godunov’s Method with the Harten–Lax–vanLeer (HLL) Solver	435
6.1.3.3	Godunov’s Method with the Harten–Hyman Fix for Roe’s Solver	436
6.2	Second-Order Schemes for Nonlinear Systems	438
6.2.1	Lax–Wendroff Method	438
6.2.2	MacCormack’s Method	439
6.2.3	Higher-Order Lax–Friedrichs Schemes	439
6.2.4	TVD Methods	443
6.2.5	MUSCL	447
6.2.6	Wave Propagation Methods	448
6.2.7	PPM	450
6.2.8	ENO	452
6.2.9	Discontinuous Galerkin Method	453

	<i>Contents</i>	xiii
6.3	Case Studies	456
6.3.1	Wave Equation	456
6.3.2	Shallow Water	456
6.3.3	Gas Dynamics	459
6.3.4	MHD	461
6.3.5	Nonlinear Elasticity	461
6.3.6	Cristescu's Vibrating String	461
6.3.7	Plasticity	464
6.3.8	Polymer Model	467
6.3.9	Schaeffer–Schechter–Shearer Model	470
7	Methods in Multiple Dimensions	474
7.1	Numerical Methods in Two Dimensions	474
7.1.1	Operator Splitting	474
7.1.2	Donor Cell Methods	476
7.1.2.1	Traditional Donor Cell Upwind Method	478
7.1.2.2	First-Order Corner Transport Upwind Method	479
7.1.2.3	Wave Propagation Form of First-Order Corner Transport Upwind	483
7.1.2.4	Second-Order Corner Transport Upwind Method	485
7.1.3	Wave Propagation	488
7.1.4	2D Lax–Friedrichs	489
7.1.4.1	First-Order Lax–Friedrichs	490
7.1.4.2	Second-Order Lax–Friedrichs	491
7.1.5	Multidimensional ENO	494
7.1.6	Discontinuous Galerkin Method on Rectangles	494
7.2	Riemann Problems in Two Dimensions	498
7.2.1	Burgers' Equation	498
7.2.2	Shallow Water	500
7.2.3	Gas Dynamics	503
7.3	Numerical Methods in Three Dimensions	506
7.3.1	Operator Splitting	506
7.3.2	Donor Cell Methods	508
7.3.3	Corner Transport Upwind Scheme	510
7.3.3.1	Linear Advection with Positive Velocity	513
7.3.3.2	Linear Advection with Arbitrary Velocity	517
7.3.3.3	General Nonlinear Problems	518
7.3.3.4	Second-Order Corner Transport Upwind	519
7.3.4	Wave Propagation	521

xiv	<i>Contents</i>	
7.4	Curvilinear Coordinates	521
7.4.1	Coordinate Transformations	522
7.4.2	Spherical Coordinates	523
7.4.2.1	Case Study: Eulerian Gas Dynamics in Spherical Coordinates	527
7.4.2.2	Case Study: Lagrangian Solid Mechanics in Spherical Coordinates	529
7.4.3	Cylindrical Coordinates	533
7.4.3.1	Case Study: Eulerian Gas Dynamics in Cylindrical Coordinates	537
7.4.3.2	Case Study: Lagrangian Solid Mechanics in Cylindrical Coordinates	539
7.5	Source Terms	542
7.6	Geometric Flexibility	542
8	Adaptive Mesh Refinement	544
8.1	Localized Phenomena	544
8.2	Basic Assumptions	546
8.3	Outline of the Algorithm	547
8.3.1	Timestep Selection	548
8.3.2	Advancing the Patches	549
8.3.2.1	Boundary Data	549
8.3.2.2	Flux Computation	550
8.3.2.3	Time Integration	552
8.3.3	Regridding	553
8.3.3.1	Proper Nesting	553
8.3.3.2	Tagging Cells for Refinement	556
8.3.3.3	Tag Buffering	559
8.3.3.4	Logically Rectangular Organization	559
8.3.3.5	Initializing Data after Regridding	559
8.3.4	Refluxing	560
8.3.5	Upscaling	560
8.3.6	Initialization	561
8.4	Object Oriented Programming	561
8.4.1	Programming Languages	562
8.4.2	AMR Classes	563
8.4.2.1	Geometric Indices	563
8.4.2.2	Boxes	567
8.4.2.3	Data Pointers	569
8.4.2.4	Lists	569

	<i>Contents</i>	xv
8.4.2.5	FlowVariables	570
8.4.2.6	Timesteps	571
8.4.2.7	TagBoxes	571
8.4.2.8	DataBoxes	571
8.4.2.9	EOSModels	572
8.4.2.10	Patch	572
8.4.2.11	Level	573
8.5	ScalarLaw Example	573
8.5.1	ScalarLaw Constructor	576
8.5.2	initialize	576
8.5.3	stableDt	577
8.5.4	stuffModelGhost	577
8.5.5	stuffBoxGhost	578
8.5.6	computeFluxes	578
8.5.7	conservativeDifference	579
8.5.8	findErrorCells	579
8.5.9	Numerical Example	579
8.6	Linear Elasticity Example	580
8.7	Gas Dynamics Examples	581
	<i>Bibliography</i>	584
	<i>Index</i>	593

Preface

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

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

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

I also tried to discuss a variety of numerical methods in this text, so that students could see a number of competing ideas. This book does not try to favor any one particular numerical scheme, and it does not serve as a user manual to a software package. It does have software available, to allow the reader to experiment with the various ideas. But the software is not designed for easy application to new problems. Instead, I hope that the readers will learn enough from this book to make intelligent decisions on which scheme is best for their problems, as well as how to implement that scheme efficiently.

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

Because this text develops analytical solutions to several problems, it is possible to measure the errors in the numerical methods on interesting test problem. This relates to a point I try to emphasize in teaching the course, that it is essential in numerical computation to perform mesh refinement studies in order to make sure that the method is performing properly. Another topic in this text is that numerical methods can be compared for accuracy (error for a given mesh size) and efficiency (error for a given amount of computational time). Sometimes people have an innate bias toward higher-order methods, but this may not be the most cost-effective approach for many problem. Efficiency is tricky to measure, because subtle programming issues can drive up computational time. I do not claim to have produced the most efficient version of any of the schemes in this text, so the efficiency comparisons should be taken "with a grain of salt."

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

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

As I gained experience with online text generation, I started to ask if it would be possible to develop an interactive text. First, I wanted students to be able to view the example programs while they were reading the text online. Next, I wanted students to be able to examine links to information available on the web. Then, I decided that it would be really nice if students could perform “what if” experiments within the text, by running numerical methods with different parameters and seeing the results immediately. Because I continue to think that only “real” programming languages (*i.e.*, C, C++ and Fortran) should be used for the material such as this, I rejected suggestions that I rewrite the programs in Matlab or Java. Eventually, our department systems programmer, Andrew Schretter, found a way to make things work for me, provided that I arrange for all parameter entry through graphical user interfaces. Our senior systems programmer, Yunliang Yu, did a lot of the development of the early form of the graphical user interface. One of my former graduate students, Wenjun Ying, programmed carefully the many cases for the marching cubes algorithm for visualizing level surfaces in three dimensions. I am 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 on a companion CD. The electronic form of the text contains links between equation or theorem references and the original statements. Similar links lead to bibliography citations or to occurrences of key words in the index. There are electronic links in the online text to source code and executables on the CD. This allows students to view computer implementations of the algorithms developed in the book, and to perform “what if” experiments with program and model parameters. However, since the text is the same for both versions of the book, this means that the paper text contains instructions to click on electronic links.

The graphical user interface (GUI) makes it easy for students to change parameters (and, in fact, to see all of the input parameters). The GUI also complicates the online programs. There is a danger that students may think that they have to program GUI's in order to solve these problems. That is not my intent. I have provided several example programs in the online version of chapter 2 to show students

how they can write simple programs (that produce data sets for post processing) or slightly more complex programs (that display numerical results during the computation to look like movies), or very sophisticated programs (that use GUI's for input parameters). I would be happy if all students could program successfully in the first style. After all, CLAWPACK is a very successful example of that simple and direct style of programming.

It is common that students in this class are taking it in order to learn programming in Fortran or C++, as much as they want to learn about the numerical methods. Both of these languages have advantages and disadvantages. Fortran is very good with arrays (subscripts can start at arbitrary values, which is useful for “ghost cells” in many methods) and has a very large set of intrinsic functions (for example, max and min with more than two arguments for slope limiters). Fortran is not very good with memory allocation, or with pointers in general. I use C++ to perform all memory allocation, and for all interactive graphics, including GUIs. When users select numerical methods through a GUI, then I set values for function pointers and pass those as arguments to Fortran routines. I do not recommend such practices for novice programmers. On the other hand, students who want to expand their programming skills can find several interesting techniques in the codes.

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

Unfortunately, mixing Fortran and C++ allows the possibility of truly bizarre programming errors. For example, declaring a Fortran subroutine to have a return value in a C++ `extern "C"` block can lead to stack corruption. I don't have a good defensive programming technique for that error.

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

Cambridge University Press

978-0-521-87727-5 - Numerical Solution of Hyperbolic Partial Differential Equations

John A. Trangenstein

Frontmatter

[More information](#)

Preface

xxi

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

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