Wave propagation software, computational science, and reproducible research

Randall J. LeVeque*

Dedicated to Germund Dahlquist (1925–2005) and Joseph Oliger (1941–2005), two of the influential mentors who have shaped my career. They each inspired generations of students with their interest in the connections between mathematics and computation.

Abstract. Wave propagation algorithms are a class of high-resolution finite volume methods for solving hyperbolic partial differential equations arising in diverse applications. The development and use of the CLAWPACK software implementing these methods serves as a case study for a more general discussion of mathematical aspects of software development and the need for more reproducibility in computational research. Sample applications discussed include medical applications of shock waves and geophysical fluid dynamics modeling volcanoes and tsunamis.

Mathematics Subject Classification (2000). Primary 65Y15; Secondary 74S10.

Keywords. Hyperbolic partial differential equations, software, numerical analysis, reproducible research, scientific computing, CLAWPACK.

1. Introduction

I will ultimately describe a class of numerical methods for solving hyperbolic partial differential equations, software that implements these methods, and some scientific applications. However, for the broad audience that I am honored to address in these proceedings, I would like to first make some more general comments on the topic of software development and its relation to mathematics, and on computational science and reproducible research.

I begin with a quote from a 1995 paper by J. B. Buckheit and D. L. Donoho [13] about wavelet analysis and a software package they developed to aid in studying and applying their methods:

> An article about computational science in a scientific publication is *not* the scholarship itself, it is merely *advertising* of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

*This research was supported in part by several NSF and DOE grants over the past decade, including NSF grants DMS-9803442, DMS-0106511, and CMS-0245206 and DOE grants DE-FG03-96ER25292, DE-FG03-00ER25292, and DE-FC02-01ER25474.

Proceedings of the International Congress of Mathematicians, Madrid, Spain, 2006

They present this as a slogan to distill the insights of Jon Claerbout, an exploration geophysicist who has been a pioneer in this direction since the early 90s (e.g., [59]), and they give many compelling examples to illustrate its truth. I first ran across this quote on the webpage [11] of the book [12], which provides complete codes in several languages for the solutions to each of the 100-digit challenge problems proposed by Trefethen [63]. (This is set of ten computational problems, each easy to state and with a single number as the answer. The challenge was to compute at least 10 digits of each number.) In spite of some progress in the direction of reproducible research, many of the complaints of Buckheit and Donoho still ring true today, as discussed further in the recent paper by Donoho and Huo [24].

Much of my work over the past 10 years has been devoted to trying to make it easier for myself, my students, and other researchers to perform computational scholarship in the field of numerical methods for hyperbolic PDEs, and also I hope in a variety of applications areas in science and engineering where these methods are used. This work has resulted in the CLAWPACK software [40]. This software is apparently being fairly widely used, both in teaching and research. More than 5 000 people have registered to download the code, mentioning all sorts of interesting problems they plan to tackle with it. However, I am not convinced that it is being used to the extent possible in advancing scholarship of the type described above. One goal of this paper is to encourage researchers (myself included) to work harder towards this end, in my field and more generally in computational science.

I will make a distinction between software and computer programs. In my notation, *software* means a package of computer tools that are designed to be applied with some generality to a class of problems that may arise in many different applications. A *computer program* is a code that solves one particular problem. A program may be written entirely from scratch or it may employ one or more software packages as tools. This is an important distinction to make since one has different goals in developing software than in writing a specific computer program. Software is intended to be used by others and to be as general as practically possible. A program is written to solve a problem and often the author does not intend for it to be seen or used by anyone else.

Pure mathematicians search for abstract structures that transcend particular examples and that often unify disparate fields. They produce theorems that apply as generally as possible and that can be used as solid and dependable building blocks for future work. In addition to developing new algorithms, some computational mathematicians also produce theorems, rigorous results guaranteeing that a particular algorithm converges or bounds on the magnitude of the error, for example. Such theorems give us the confidence to apply the algorithm to real world problems.

Other computational mathematicians focus on the development of software that implements an algorithm in a dependable manner. This is perhaps an under-appreciated art in the mathematical world, compared to proving theorems, but I believe it is an analogous mathematical activity in many ways. In both cases the goal is to distill the essence of a set of particular examples into a general result, something that applies as broadly as possible while giving an interesting and nontrivial result that

can be built upon and used as a "subroutine" in future work. In both cases the result is an encapsulation of a set of knowledge that has well defined inputs and outputs and is believed to be proved correct, and that applies in many situations.

The process of developing a novel algorithm and writing software to implement it is also in some ways similar to the process involved in proving a theorem. One needs some mathematical insight to get started, but then working through many technical details is typically required to make everything fit together in a manner that produces the desired result. This part is not very glamorous but is a crucial part of the scholarship. Often frustrations arise when one little detail does not work out quite the way one hoped. Sometimes algorithms, like partially completed proofs, must be shelved for years until someone else makes a breakthrough in a related area that suddenly makes everything come together.

And everything does have to fit together just right; having a nice idea that seems like it should work is not enough. Glossing over the details is not allowed, and is particularly hard to pull off in a computer program. While it may be possible to slip things by the referees in the description of an algorithm in a paper (as also sometimes happens in a shoddy proof), computers will not parse the command "and then a miracle occurs". We are forced to be explicit in every step.

Of course even once a program does work, in the sense of compiling without errors and producing results that seem reasonable, we are faced with the thorny issue of "proving" that it is in fact correct. In computer science there is a whole field devoted to developing methodologies for formally proving the correctness of computer programs. In computational science the programs are often so complex and the problem it is designed to solve so ill-defined that formal correctness proofs are generally impossible. Instead the buzzwords are *Verification and Validation* (V&V for short). These can be summarized by the following mnemonic:

Verification: Are we solving the problem right?
Validation: Are we solving the right problem?

For a physical experiment modeled by partial differential equations, for example, we must verify that the computer program solves the PDEs accurately. A computational scientist must also validate the code by comparing it against experiments to ensure that the PDEs discretized are actually a sufficient model of reality for the situation being modeled. The Euler equations of gas dynamics are sufficient in some situations, but completely inadequate in other cases where viscosity plays a significant role.

Researchers in numerical analysis and scientific computing (as defined in the next section) are generally most concerned with verification, while scientists, engineers, and applied mathematicians focusing on mathematical modeling must also be concerned with validation. Even the relatively simple task of verifying that a code solves the given equations properly can be a real challenge, however, and is often as much an art as a science. A good test problem that captures the essence of some potential difficulty while having a solution that can be checked is often hard to come by, and developing test suites for different classes of algorithms is valuable scholarship in

itself. Numerous papers have been written on the subject of how best to test computer programs or software for scientific computing; see [17], [29], [33], [38], [53], or [56] for just a few approaches.

Elegance is valued in algorithm and software design as it is in other mathematical endeavors. Often the first attempt at an algorithm is not very clean; it is a brute force approach that gets the job done. Later work is often devoted to cleaning things up, perhaps in fundamental ways that greatly reduce the computational complexity, but also often in more subtle ways that simply make it more "elegant", a hard to define property that we recognize when we see.

Perhaps I am straying too far from the topic of reproducible scientific computing, but to make progress in this direction I think it is important to recognize software development as a valid and challenging mathematical activity. It takes a slightly different type of mathematician to develop the necessary intuition and skills to excel at this than is required to prove theorems, just as doing algebra vs. analysis takes different mindsets and these are rarely done equally well by the same mathematician. But no one doubts that algebraists and analysts are both mathematicians, even if they cannot get beyond the first page of each others' papers. Knuth [35] did an interesting study on the connections between algorithmic and mathematical thinking, a topic that he also touched on in an earlier paper [34] on "computer science and its relation to mathematics". This paper was written in 1974, at a time when many computer science departments were just being established, often by mathematicians. It makes interesting reading today, along with similar papers of the same vintage, such as [9], [25]. Software development is a logical conclusion of algorithmic thinking, and the development of software for mathematical algorithms naturally belongs in a mathematics department.

The reason I care about this topic is not for my own mathematical ego. I have been lucky to be at an institution where my work in this direction has been encouraged, or at least tolerated. It may have helped that I did not put much effort into software development until well after I was tenured.[1] The main value of the tenure system is that established people do not need to worry what our colleagues think of our activities or how they choose to label them. But the future depends on bright young people. I think computational science affords a wonderful opportunity to get students involved in a host of mathematical challenges, and making significant progress on these requires computational mathematicians with solid training in a broad range of mathematical

[1] However, many of my attitudes towards software development were shaped by my experiences as a graduate student in the Computer Science Department at Stanford, where students in the numerical analysis group were responsible for maintaining the library of numerical routines [10] available to physicists at the Stanford Linear Accelerator Center (SLAC) and acting as consultants, activities that were encouraged by Gene Golub and Joe Oliger. There I had the pleasure of working directly with an outstanding set of fellow students, most of whom have gone on to make software contributions of their own, including Marsha Berger, Petter Bjorstad, Dan Boley, Tony Chan, Bill Coughran, Bill Gropp, Eric Grosse, Mike Heath, Frank Luk, Stephen Nash, Michael Overton, and Lloyd Trefethen. Many of us were also shaped by Cleve Moler's course on numerical linear algebra, where he tried out his new MATLAB program on us as a front end to the LINPACK and EISPACK routines that were already setting the standard for mathematical software [23], [27], [61]. I think the Computer Science students were more impressed with MATLAB and much more influenced by this experience than Cleve takes credit for in [48].

tools and the ability to apply mathematical abstraction to common problems arising in multiple fields. While there are many talented computational scientists working on specific challenging problems in virtually every science and engineering department, a computational mathematician, centered in a mathematics (or applied mathematics) department, has the best chance of appreciating the common mathematical structure of the problems and producing algorithms and software that are broadly applicable. Doing so not only avoids a lot of wasted effort by scientists whose time is better spent on the peculiarities of their specialty, it also leads to the introduction of techniques into fields where they might not otherwise be invented and the discovery of new connections between existing algorithms and applications.

I once heard Jim Glimm remark that "applied mathematicians are the honey bees of the mathematical world, whose job is to cross-pollinate applications." In addition to providing a service to those in other disciplines, the process of collecting nectar can result in some sweet rewards back in our own hive. This is equally true in algorithm and software development as it is in more classical and theoretical aspects of applied mathematics.

But young mathematicians will feel free to pursue such activities, and to also do the less rewarding but crucial aspects of the scholarship such as documenting their codes and making them presentable to the rest of the world, only if it is accepted as valid mathematical scholarship. If it is seen as non-mathematics, it will only be a waste of time that is best avoided by anyone seeking tenure in a mathematics department.

Applied mathematics in general is becoming much more acceptable in mathematics departments than it once was, at least in the United States. However, I doubt that the careful development of software or computer programs, or the work required to turn research codes into publishable scholarship, has the same level of acceptance.

2. Numerical analysis, scientific computing, and computational science & engineering

One can argue at length about the meaning of the terms in this section title. To me, "numerical analysis" has a double meaning: the analysis and solution of real-world problems using numerical methods, and the invention and analysis of the methods themselves using the techniques of mathematics.

When used in the latter sense, numerical analysis belongs firmly in a mathematics (or applied mathematics) department. As just one example, analyzing the stability and convergence properties of finite difference or finite element methods is no less difficult (often more difficult) than analyzing the underlying differential equations, and relies on similar tools of analysis. Specialists in this type of numerical analysis may or may not do much computing themselves, and may be far removed from computational science.

Numerical analysis in the sense of using numerical methods to solve problems, or developing software for general use, is often called "scientific computing" or "com-

putational science & engineering" these days. One can make a further distinction between these two terms: Scientific computing is often used to refer to the development of computational tools useful in science and engineering. This is the main thrust of the *SIAM Journal on Scientific Computing*, for example, which contains few theorems relative to the more theoretical *SIAM Journal on Numerical Analysis*, but still focuses on mathematical and algorithmic developments. Computational science & engineering refers more specifically to the use of computational tools to do real science or engineering in some other field, as a complement to experimental or theoretical science and engineering.

Not everyone would agree with my definitions of these terms. In particular, it can be argued that "computational science" refers to the science of doing computation and "computational engineering" to the implementation of this science in the form of software development, but for my purposes I will lump these two activities under "scientific computing". It is important to be aware of this lack of consistency in nomenclature since, for example, many recently developed academic programs in Computational Science & Engineering stress aspects of scientific computing as well.

I have been arguing that "scientific computing", in the sense just described, is a branch of mathematics (as well as being a branch of other disciplines, such as computer science), and that other mathematicians should be more aware of the intellectual challenges and demands of this field, including the need to document and distribute code. Not only are the activities of many practitioners of scientific computing essentially mathematical, but they (and their students) benefit greatly from frequent contact with more theoretical numerical analysts and mathematicians working in related areas. Other mathematicians may also benefit from having computational experts in the department, particularly as more fields of pure mathematics develop computational sides and realize the benefits of experimental mathematics – there is even a journal (see expmath.org) now devoted to this approach.

3. Reproducible research

Within the world of science, computation is now rightly seen as a third vertex of a triangle complementing experiment and theory. However, as it is now often practiced, one can make a good case that computing is the last refuge of the scientific scoundrel. Of course not all computational scientists are scoundrels, any more than all patriots are, but those inclined to be sloppy in their work currently find themselves too much at home in the computational sciences. Buckheit and Donoho [13] refer to the situation in the field of wavelets as "a scandal". The same can be said of many other fields, and I include some of my own work in the category of scandalous.

Where else in science can one get away with publishing observations that are claimed to prove a theory or illustrate the success of a technique without having to give a careful description of the methods used, in sufficient detail that others can attempt to repeat the experiment? In other branches of science it is not only expected

that publications contain such details, it is also standard practice for other labs to attempt to repeat important experiments soon after they are published. Even though this may not lead to significant new publications, it is viewed as a valuable piece of scholarship and a necessary aspect of the scientific method.

Scientific and mathematical journals are filled with pretty pictures these days of computational experiments that the reader has no hope of repeating. Even brilliant and well intentioned computational scientists often do a poor job of presenting their work in a reproducible manner. The methods are often very vaguely defined, and even if they are carefully defined they would normally have to be implemented from scratch by the reader in order to test them. Most modern algorithms are so complicated that there is little hope of doing this properly. Many computer codes have evolved over time to the point where even the person running them and publishing the results knows little about some of the choices made in the implementation. And such poor records are typically kept of exactly what version of the code was used and the parameter values chosen that even the author of a paper often finds it impossible to reproduce the published results at a later time.

The idea of "reproducible research" in scientific computing is to archive and make publicly available all of the codes used to create the figures or tables in a paper in such a way that the reader can download the codes and run them to reproduce the results. The program can then be examined to see exactly what has been done.

The development of very high level programming languages has made it easier to share codes and generate reproducible research. Historically, many papers and text books contained pseudo-code, a high level description of an algorithm that is intended to clearly explain how it works, but that would not run directly on a computer. These days many algorithms can be written in languages such as MATLAB in a way that is both easy for the reader to comprehend and also executable, with all details intact. For example, we make heavy use of this in the recent paper [15]. We present various grid mappings that define logically rectangular grids in smooth domains without corners, such as those shown later in Figure 1, and all of the MATLAB codes needed to describe the various mappings are short enough to fit naturally in the paper. The associated webpage contains the longer CLAWPACK codes used to solve various hyperbolic test problems on these grids.

Trefethen's book on spectral methods [62] is a good example of a textbook along these lines, in which each figure is generated by a 1-page MATLAB program. These are all included in the book and nicely complement the mathematical description the methods discussed. Trefethen makes a plea for more attention to short and elegant computer programs in his recent essay [64].

However, for larger scale computer programs used in scientific publications, there are many possible objections to making them available in the form required to reproduce the research. I will discuss two of these, perhaps the primary stumbling blocks.

One natural objection is that it is a lot of work to clean up a code to the point where someone else can even use it, let alone read it. It certainly is, but it is often well

worth doing, not only in the interest of good science but also for the personal reason of being able to figure out later what you did and perhaps build on it.

Those of us in academia should get in the habit of teaching good programming, documentation, and record keeping practices to our students, and then demand it of them. We owe it to them to teach this set of computational science skills, ones that I hope will be increasingly necessary in academic research environments and that are also highly valued in industrial and government labs. It will also improve the chances that we will be able to build on the work they have done once they graduate, and that future students will be able to make use of it rather than starting from scratch as is too often the case today.

While ideally all published programs would be nicely structured and easily readable with ample comments, as a first step it would be valuable simply to provide and archive the working code that produced the results in a paper. Even this takes more effort than one might think. It is important to begin expecting this as a natural part of the process so that people will feel less like they have to make a choice between finishing off one project properly or going on to another where they can more rapidly produce additional publications. The current system strongly encourages the latter.

As Buckheit and Donoho [13] point out, the scientific method and style of presenting experiments in publications that is currently taken for granted in the experimental sciences was unheard of before the mid-1800s. Now it is a required aspect of respectable research and experimentalists are expected to spend a fair amount of time keeping careful lab books, fully documenting each experiment, and writing their papers to include the details needed to repeat the experiments. A paradigm shift of the same nature may be needed in the computational sciences.

Requiring it of our students may be a good place to start, provided we recognize how much time and effort it takes. Perhaps we should be more willing to accept an elegant and well documented computer program as a substantial part of a thesis, for example.

A second objection to publishing computer code is that a working program for solving a scientific or engineering problem is a valuable piece of intellectual property and there is no way to control its use by others once it is made publicly available. Of course if the research goal is to develop general software then it is desirable to have as many people using it as possible. However, for a scientist or mathematician who is primarily interested in studying some specific class of problems and has developed a computer program as a tool for that purpose, there is little incentive to give this tool away free to other researchers. This is particularly true if the program has taken years to develop and provides a competitive edge that could potentially lead to several additional publications in the future. By making the program globally available once the first publication appears, other researchers can potentially skip years of work and start applying the program to other problems immediately. In this sense providing a program is fundamentally different than carefully describing the materials and techniques of an experiment; it is more like inviting every scientist in the world to come use your carefully constructed lab apparatus free of charge.

This argument undoubtedly has considerable merit in some situations, but on the whole I think it is overblown. It is notoriously difficult to take someone else's code and apply it to a slightly different problem. This is true even when people are trying to collaborate and willing to provide hands-on assistance with the code (though of course this type of collaboration does frequently occur). It is often true even when the author of the code claims it is general software that is easy to adapt to new problems. It is particularly true if the code is obscurely written with few comments and the author is not willing to help out, as would probably be true of many of the research codes people feel the strongest attachment to.

Moreover, my own experience in computational science is that virtually every computational experiment leads to more questions than answers. There is such a wealth of interesting phenomena that can be explored computationally these days that any worthwhile code can probably lead to more publications than its author can possibly produce. If other researchers are able to take the code and apply it in some direction that would not otherwise be pursued, that should be seen as a positive development, both for science and for its original author, provided of course that s/he gets some credit in the process. This is particularly true for computational mathematicians, whose goals are often the development of a new algorithm rather than the solution of specific scientific problems. Even for those not interested in software development *per se*, anything we can do to make it easier for others to use the methods we invent should be viewed as beneficial to our own careers.

Perhaps what is needed is some sort of recognized patent process for scientific codes, so that programs could be made available for inspection and independent execution to verify results, but with the understanding that they cannot be modified and used in new publications without the express permission of the author for some period of years. Permission could be granted in return for co-authorship, for example. In fact such a system already works quite well informally, and greater emphasis on reproducible research would make it function even better. It would be quite easy to determine when people are violating this code of ethics if everyone were expected to "publish" their code along with any paper. If the code is an unauthorized modification of someone else's, this would be hard to hide.

4. Wave propagation algorithms and CLAWPACK

As a case study in software development, and its relation to mathematics, scientific computing, and reproducible research, I will briefly review some of the history behind my own work on CLAWPACK (Conservation LAWs PACKage), software for solving hyperbolic systems of partial differential equations.

This software development project began in 1994. I had just taught a graduate course on numerical methods for conservation laws and had distributed some sample computer programs to the students as a starting point for a class project. In the fall I went on sabbatical and decided to spend a few weeks cleaning up the program and

redesigning it as a software package, in large part because I was also planning to spend much of the year revising my lecture notes [42] from a course I taught at ETH-Zürich in 1989 into a longer book, and I wanted to complement the text with programs the students could easily use.

I seriously misjudged the effort involved – I spent most of that year and considerable time since developing software, which grew into something much more that I originally intended. The book [44] took several more years of work and iterations of teaching the course and did not appear until 2002.

Virtually all of the figures in this book are reproducible, in the sense that the programs that generated them can each be downloaded from a website and easily run by the student. Most figure captions contain a link to the corresponding website, in the form [claw/book/chap23/advection/polar], for example, from the caption of Figure 23.3, which is easily translated into the appropriate web address. (Start at http://www.amath.washington.edu/~claw/book.html to browse through all these webpages). Each webpage contains the computer code and many also contain additional material not in the book, for example movies of the solution evolving in time.

All of these examples are based on the CLAWPACK software. This software is described briefly in the book and more completely in the User Guide [41] available on the web. Once the basic software is installed, the problem-specific code for each example in the book is quite small and easy to comprehend and modify. The reader is encouraged to experiment with the programs and observe how changes in parameters or methods affect the results. These programs, along with others on the CLAWPACK website [40], can also form the basis for developing programs to solve similar problems.

This software implements a class of methods I call "wave propagation algorithms" for solving linear or nonlinear hyperbolic problems. Hyperbolic partial differential equations are a broad class of equations that typically model wave propagation or advective transport phenomena. The classic example is the second-order wave equation $p_{tt} = c^2 p_{xx}$ for linear acoustics, modeling the propagation of pressure disturbances in a medium with sound speed c. The CLAWPACK software, however, is set up to solve a different form of hyperbolic equations: systems that involve only first order derivatives in space and time.

In the linear case, a first-order system of PDEs (in one space dimension and time) has the form $q_t + Aq_x = 0$, where $q(x, t)$ is a vector of some m conserved quantities, A is an $m \times m$ matrix, and subscripts denote partial derivatives. In the nonlinear case, a system of m conservation laws takes the form $q_t + f(q)_x = 0$, where $f(q)$ is the flux function (in the linear case, $f(q) = Aq$). This system is called *hyperbolic* if the flux Jacobian matrix $f'(q)$ is diagonalizable with real eigenvalues. The system of Euler equations for inviscid compressible gas dynamics has this form, for example, where mass, momentum, and energy are the conserved quantities. The full nonlinear equations can develop shock wave solutions in which these quantities are discontinuous, one of the primary challenges in numerical modeling. Linearizing this system gives the linear acoustics equations in the form of a first-order system of

equations for pressure and velocity. Cross differentiating this system allows one to eliminate velocity and obtain the single second-order wave equation for p mentioned above, but the first-order formulation allows the modeling of a much broader range of phenomena.

First-order hyperbolic systems arise naturally in a multitude of applications, including for example elastodynamics (linear and nonlinear), electromagnetic wave propagation (including nonlinear optics), shallow water equations (important in oceanography and atmospheric modeling), and magnetohydrodynamic and relativistic flow problems in astrophysics.

The wave-propagation algorithms are based on two key ideas: Riemann solvers and limiters. The *Riemann problem* consists of the hyperbolic equation under study with special initial conditions at some time $\bar{t}$: piecewise constant data with left state q_ℓ and right state q_r and a jump discontinuity in each conserved quantity at a single point in space, say $\bar{x}$. The solution to this Riemann problem for $t > \bar{t}$ is a similarity solution, a function of $(x-\bar{x})/(t-\bar{t})$ alone that consists of a set of waves propagating at constant speeds away from the initial discontinuity. The definition of hyperbolicity guarantees this, and the eigenvalues of the flux Jacobian are related to the wave speeds. In the linear case the eigenvalues of the matrix A are exactly the wave speeds. In the nonlinear case the eigenvalues vary with q. The Riemann solution may then contain shock waves and rarefaction waves, but even in the nonlinear case this special problem has a similarity solution with constant wave speeds.

In 1959, Sergei Godunov [30] proposed a numerical method for solving general shock wave problems in gas dynamics by using the Riemann problem as a building block. If the physical domain is decomposed into a finite number of grid cells and the solution approximated by a piecewise constant function that is constant in each grid cell, then at the start of a time step the initial data consists of a set of Riemann problems, one at each cell interface. By solving each of these Riemann problems the solution can be evolved forward in time by a small increment. The resulting solution is averaged over each grid cell to obtain a new piecewise constant approximation to the solution at the end of the time step. This procedure is repeated in the next time step. This idea of basing the numerical method on Riemann solutions turned out to be a key idea in extending the "method of characteristics" from linear hyperbolic systems to important nonlinear shock propagation problems. This also leads naturally to a software framework: the particular hyperbolic equation being solved is determined by providing a Riemann Solver. This is a subroutine that, given any two states q_ℓ and q_r, returns the wave speeds of the resulting waves in the similarity solution, along with the corresponding waves themselves (i.e., the jump in q across each wave). The updating formulas for the cell averages based on these waves are very simple and independent of the particular system being solved.

Godunov's method turned out to be very robust and capable of solving problems involving strong shocks where other methods failed. However, it is only first-order accurate on smooth solutions to the PDEs. This means that the error goes to zero only as the first power of the discretization steps Δx and Δt. Moreover, although

complicated solutions involving strong shocks and their interactions could be robustly approximated without the code crashing, the resulting approximations of shock waves are typically smeared out. The process of averaging the solution over grid cells each time step introduces a large amount of "numerical dissipation" or "numerical viscosity".

During the 1970s and 1980s, a tremendous amount of effort was devoted to developing more accurate versions of Godunov's method that better approximated smooth solutions and also captured shock waves more sharply. These methods often go by the general name of "high-resolution shock capturing methods". A wide variety of methods of this type have been proposed and effectively used. Many of these, including the wave-propagation algorithms of CLAWPACK, have the relatively modest goal of achieving something close to second-order accuracy on smooth solutions coupled with sharp resolution of discontinuities. Other approaches have been used that can achieve much better accuracy in certain situations, though for general nonlinear problems involving complicated shock structures, particularly in more than one dimension, it seems hard to improve very much beyond what is obtained using second-order methods.

One standard second-order method for a linear hyperbolic system is the Lax–Wendroff method, first proposed in 1960, which is based on approximating the first few terms of a Taylor series expansion of the solution at time $t + \Delta t$ about the solution at time t. This method does not work at all well for problems with discontinuities, however, as it is highly dispersive and nonphysical oscillations arise that destroy all accuracy. In the nonlinear case these oscillations around shock waves can also lead to nonlinear instabilities.

The key feature in many high-resolution methods is to apply a *limiter function* in some manner to suppress these oscillations. In the wave-propagation algorithms this is done in the following way. The Lax–Wendroff method can be rewritten as Godunov's method plus a correction term that again can be expressed solely in terms of the waves and wave speeds in the Riemann solutions arising at each cell interface. Where the solution is smooth, adding in these correction terms improves the accuracy. Where the solution is not smooth, for example near a shock, the Taylor series expansion is not valid and these "correction terms", which approximate higher derivatives in the Taylor expansions, do more harm than good. We can determine how smooth the solution is by comparing the magnitude of a wave with the magnitude of the corresponding wave at the neighboring cell interfaces. If these differ greatly then the solution is not behaving smoothly and the correction terms should be "limited" in some manner.

Many variants of this idea have been used. In some cases it is the Lax–Wendroff expression for the flux at the interface between cells that is limited (in so-called "flux limiter" methods). Another approach is to view the Lax–Wendroff method as a generalization of Godunov's method in which a piecewise linear function in each grid cell is defined and the values at the edges of each cell then used to define Riemann problems. In this case the slope chosen in each cell is based on the averages in nearby cells, with some "slope limiter" applied in regions where it appears the solution is not behaving smoothly. For the special case of a scalar conservation law, a very

nice mathematical theory was developed to guide the choice of limiter functions. The true solution to a scalar problem has its total variation non-increasing over time. By requiring that a numerical method be "total variation diminishing" (TVD) it was possible to derive methods that were essentially second order accurate, or higher, but that could be proved to not introduce spurious oscillations.

The 1980s were an exciting time in this field, as robust high-resolution methods were developed for a variety of challenging applications and as computers became powerful enough that extensions of these methods and the related mathematical theory to more than one space dimension became possible and necessary.

I played a modest role in some of these developments, but I think my main contribution has been in providing a formulation of these methods that lends itself well to software that is very broadly applicable, and in leading the effort to write this software. The wave-propagation formulation that I favor has the advantage that once the Riemann problem has been solved, the limiters and high-resolution correction terms are applied in a general manner that is independent of the equation being solved. Moreover a framework for doing this in two dimensions was proposed in [43] that retains this modularity, separating the process of solving a one-dimensional Riemann problem at each cell interface, along with a related "transverse Riemann problem" in the orthogonal direction, from the process of propagating these waves and correction terms with appropriate limiters. While it had long been recognized that the methods being developed were in principle broadly applicable to all hyperbolic problems, for systems of equations most methods were developed or at least presented in a form that was specific to one particular problem (often the Euler equations of gas dynamics) and a certain amount of work was required to translate them to other problems. Computer programs that I was aware of were all problem-specific, and generally not publicly available.

My original motivation for developing this framework was not software development, but rather the need to teach graduate students, and the desire to write a book that explained how to apply these powerful high-resolution methods to a variety of problems. I had students coming from the Mathematics and Applied Mathematics departments, many of whom knew little about fluid dynamics, and students from several science and engineering departments who had specific interests in very diverse directions.

My subsequent motivation for software development was partly educational, but also partly because I wanted to better publicize the wave-propagation framework I had developed and make it easier for others to use methods in this form. I recognized that these methods, while quite general, were also sufficiently complicated that few would bother to implement them from my descriptions in journal articles. For this research to have any impact beyond a few publications it seemed necessary to provide more than pseudo-code descriptions of the algorithms.

5. CLAWPACK as an environment for developing and testing methods

I have provided a few basic sample programs within the CLAWPACK package itself, and some additional test problems in a directory tree labeled applications available at the website. Complete programs also exist for each of the figures in my book. These each consist of a tarred Unix directory containing a small set of subroutines to be used together with the main software. These subroutines specify the specific problem, including the Riemann solver and the initial and boundary conditions. Many standard boundary conditions are already available in the default boundary condition routine, and a variety of Riemann solvers are also provided for many different systems of equations. As a result it is often very easy to adapt one of these examples to a new problem, particularly for some of the standard test problems that often appear in publications.

Although the development of the CLAWPACK software was originally motivated by the desire to make a set of existing methods more broadly accessible, the availability of this software has also encouraged me to pursue new algorithmic advances that I otherwise might not have. I hope that the software will also prove useful to others as a programming environment for developing and testing new algorithms, and for comparing different methods on the same problems. Since the source code is available and the basic CLAWPACK routines are reasonably simple and well documented, it should be easy for users to modify them and try out new ideas. I encourage such use, and I certainly use it this way myself.

As one example, there are many approaches to developing the "approximate Riemann solvers" that are typically used for nonlinear problems. It often is not cost effective, and may not even be possible, to solve the Riemann problem exactly at every cell interface each time step. Different versions of the Riemann solver can easily be swapped in and tested on a set of problems. See Figures 15.5 and 15.6 in [44] and the associated programs for an example of this sort of comparison. The CLAWPACK software also comes with a set of standard limiter functions, and an input parameter specifies which one will be used. The subroutine where these are implemented can be easily modified to test out new approaches.

More extensive modifications could be made to the software as well, for example by replacing the wave-propagation algorithms currently implemented by a different approach. If a new method is formulated so that it depends on a Riemann solver and boundary conditions that are specified in the form already used in CLAWPACK, then it should be easy to test out the new method on all the test problems and applications already developed for CLAWPACK. This would facilitate comparison of a new method with existing methods.

Careful direct comparisons of different methods on the same test problems are too seldom performed in this field, as in many computational fields. One reason for this is the difficulty of implementing other peoples' methods, so the typical paper contains only results obtained with the authors' method. Sometimes (not always) the method has been tested on standard test problems and the results can be compared with

others in the literature, with some work on the reader's part, and assuming the reader is content with comparisons in the "eyeball norm" since many papers only contain contour plots of the computed solution and no quantitative results. Of course there are many exceptions to this, including some papers devoted to careful comparisons of different methods (e.g., [31], [46]), but these papers are still a minority. I hope that CLAWPACK might facilitate this process more in the future, and that new algorithms of this same general type might be provided in a CLAWPACK implementation that allows direct use and comparison by others.

6. CLAWPACK extensions and infrastructure

This software effort began about 10 years ago, as recounted above, and has continued in unexpected ways as the software grew beyond my initial intentions in several directions. While originally I viewed it primarily as an educational tool, it was based on my own research codes and I realized that it could perhaps be valuable for other researchers as well, perhaps even as a tool for solving real problems and not just academic test problems. To make it more useful as a general tool required several enhancements.

I originally wrote subroutines for solving systems in one and two space dimensions, but started collaborating with Jan Olav Langseth that same year on the development of three-dimensional generalizations [37]. He took my course as a visiting graduate student from the University of Oslo and went on to write a thesis partly on this topic [36], and wrote the three-dimensional subroutines in CLAWPACK.

I had also been collaborating with Marsha Berger on research related to using cut-cell Cartesian grids to solve the Euler equations in non-rectangular geometries (e.g., [5], [6]) and was familiar with her implementation of adaptive mesh refinement on rectangular grids, using an approach pioneered in her work with Joe Oliger and Phil Colella [4], [8], [7]. While still on sabbatical, in 1995 we started working together to modify her adaptive mesh refinement (AMR) code and make it more generally applicable to any hyperbolic system that can be solved in CLAWPACK. In this approach the rectangular grid is refined by introducing finer grids on rectangular patches. Several levels of refinement are allowed, by an arbitrary refinement factor at each level. The program automatically refines and de-refines as the computation proceeds, based on a standard default or user-specified error criterion, and so fine grids are used only where they are needed. This approach is particularly valuable for problems involving shock waves propagating in three space dimensions. To capture the shock waves as sharp discontinuities, even with high-resolution methods, requires a fairly fine grid around the shock, much finer than is needed in regions where the solution is smooth. Without AMR many three-dimensional problems would be impractical to solve except on the largest supercomputers.

This AMRCLAW software is now a standard part of CLAWPACK, and was extended from two to three space dimensions with programming help from Dave McQueen and

Donna Calhoun. The goal of the AMRCLAW code is primarily to facilitate research and practical problem solving, rather than to teach adaptive refinement techniques. The core library consists of about 9 000 lines of Fortran 77 code that is quite convoluted and not structured or fully documented in a way that others can easily understand. This stems in large part from its history as a merging together of two different codes, with different notation and conventions that have largely been preserved and worked around, and with many new features added over the years as afterthoughts that were not originally designed for. It may not be particularly elegant, but it has proved valuable in solving practical problems and by now has been tested to the point where it is fairly robust and reliable. All of the source code is available for others to inspect and modify, at their own peril perhaps.

The CLAWPACK framework can also be used with other AMR packages. Sorin Mitran is developing BEARCLAW [47], a Fortran 90 version with similar capabilities to AMRCLAW but designed with these capabilities in mind from the beginning. This code is still being developed and has not been tested as extensively as AMRCLAW, but it has been successfully used in astrophysical applications [51], [52]. Ralf Deiterding has developed a general purpose adaptive refinement code AMROC [18], [19] in C++ that also allows the use of the CLAWPACK solvers with adaptive refinement. This is a primary computational tool in the Virtual Shock Physics Test Facility at the Caltech ASC Center for Simulation of Dynamic Response of Materials [1]. Recently Donna Calhoun has written a CHOMBO-CLAW interface [14] between the CLAWPACK solvers and the C++ CHOMBO software package developed by Colella's group at Lawrence Berkeley Lab [16]. These newer packages have various advantages over the AMRCLAW software in CLAWPACK. They are written in more advanced languages that are more suitable for the data structures and dynamic memory allocation needed in AMR codes. They also have more capabilities such as parallel implementations, the ability to handle implicit methods, and/or coupling with elliptic solvers as required in some applications.

Other extensions of CLAWPACK have also been developed, such as the CLAWMAN software [2] that solves hyperbolic systems on curved two-dimensional manifolds. This has been used to solve geophysical flow problems on the sphere and relativistic flow problems in curved space-time near a black hole [3], [54], [55].

The original software was designed for purely Cartesian grids in rectangular regions of space. Some practical problems have this form, but most are posed in more complicated physical domains, e.g., for flows around or through a physical object. There are many approaches to handling complex geometries. The cut-cell Cartesian grid approach has already been mentioned above. At the other extreme lie unstructured grids, typically composed of triangular cells in two dimensions or tetrahedra in three dimensions. These can conform to very general boundaries, but grid generation then becomes a challenging problem in itself, and implementations must deal with special data structures to keep track of what cells are adjacent to one another.

For fairly simple domains, a good compromise is often possible in which the grid is logically rectangular in computational space, but is mapped to a nonrectagular

physical domain. In two dimensions, this means that each grid cell is a quadrilateral and simple (i, j) indexing can be used to denote the cells, with neighboring cells having indices $(i \pm 1, j \pm 1)$. In three dimensions the grid cells are hexahedral but still logically rectangular. It is quite easy to apply CLAWPACK in such situations (as described in Chapter 23 of [44]), with a standard set of additional subroutines used to specify the mapping function. One nice feature of this approach is that the AMR routines work perfectly well on mapped grids – the patches of refinement are still rectangular in computational space. The wave-propagation algorithms turn out to work quite robustly on quadrilateral grids, even if the grid is nonorthogonal and far from smooth. Rather than incorporating the grid mapping directly into the differential equations as "metric terms" that involve derivatives of the mapping function, in the wave-propagation approach one solves one-dimensional Riemann problems orthogonal to each grid interface and transverse Riemann problems based on the adjacent cell interfaces.

Figure 1 shows two grids from some recent work with Calhoun [15] on the use of logically rectangular grids for solving problems in domains with smooth boundaries. The figure on the left shows a quadrilateral grid for a circle, while that on the right is a logically rectangular grid on the sphere. Each grid is simply a rectangle in computational space. Of course polar coordinates also give a logically rectangular grid in a circle, but grid lines coalesce at the center where cells have much smaller area than those near the perimeter. This presents a problem when using explicit

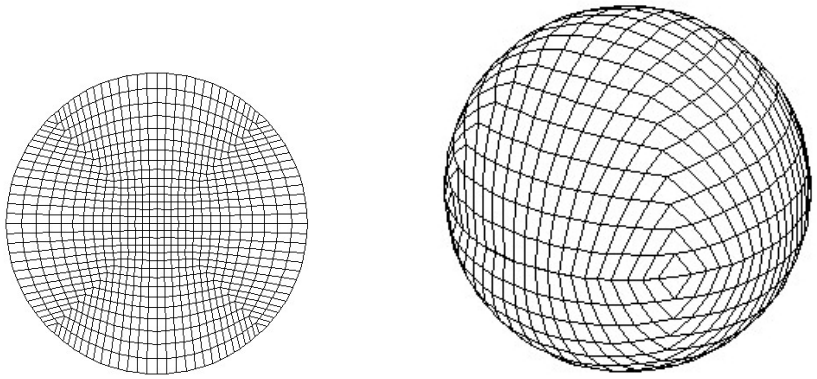


Figure 1. Quadrilateral grids on the circle and the sphere. In each case the computational domain is a rectangular grid. The MATLAB code that generates these grids is available on the webpage [39].

methods for hyperbolic problems: the disparity in cell sizes leads to an undesirable restriction on the time step. The grids in Figure 1 are far from smooth or orthogonal; in fact the images of the two orthogonal cell edges at each corner of the rectangular computational domain are nearly collinear in the physical domain. But the cell areas are nearly uniform, differing by at most a factor of 2. Standard finite difference

methods would presumably not work well on these grids, but applying CLAWPACK yields very nice results, as demonstrated in [15].

Writing new methods in "CLAWPACK form" makes it possible to take advantage of the infrastructure developed for this software, for example to apply the method on a general quadrilateral grid. It may also be possible to apply adaptive mesh refinement quite easily with the new method by taking advantage of one of the AMR wrappers described above. Since implementing AMR effectively is far from trivial, and often has little to do with the particular numerical method used on each grid patch, making use of existing implementations could be worthwhile even if it takes a bit of work to formulate the method in an appropriate form. Extensive graphics routines in MATLAB have been written to plot the results computed by CLAWPACK. These routines (written mostly by Calhoun and myself) deal with adaptive mesh refinement data in two or three space dimensions. This is a not easily done directly with most graphics packages, and again this infrastructure may be useful to developers of new methods. (In fact these graphics routines can take AMR data produced by any program, not just CLAWPACK, provided it is stored in the appropriate form.)

One current research project, with graduate student David Ketcheson, is to implement higher order numerical methods in CLAWPACK, specifically the weighted essentially non-oscillatory (WENO) methods that are based on higher order interpolation in space coupled with Runge–Kutta time stepping (e.g., [31], [60]). Doing so requires reformulating these methods slightly, and in particular we are developing a version that works for linear hyperbolic systems that are not in conservation form. Systems of this form arise in many applications, such as the propagation in heterogeneous media of acoustic, elastic, or electromagnetic waves. In many linear applications the solution is smooth but highly oscillatory, and in this case higher-order methods may be beneficial.

7. Applications in computational science

In addition to continuing to work on algorithm development, in the past few years I have become more directly involved in applications of computational science, driven in part by the existence of this software as a starting point. Problems that can be solved easily with the existing software have little interest to me; as a mathematician I consider them solved, though there may be plenty of interesting science to be done by judicious use of the software as a tool. This is best done, however, by scientists who are experts in a particular domain.

A practical problem where CLAWPACK fails to perform well, or where some substantial work is required to apply it, is much more interesting to me. Although the methods implemented in CLAWPACK work well on many classical hyperbolic problems, there are a wealth of more challenging problems that are yet to be solved, and I hope that CLAWPACK might form the basis for approaching some of these problems without the need to rewrite much of the basic infrastructure.

In the remainder of this section I will briefly describe a few topics that are currently occupying me and my students. More details on these and other problems, along with movies, papers, and sometimes computer code, can be found by clicking on the "Research interests" link from my webpage.

Shock wave therapy and lithotripsy. Focused shock waves are used in several medical procedures. Extracorporeal shock wave lithotripsy (ESWL) is a standard clinical procedure for pulverizing kidney stones noninvasively. There are several different lithotripter designs. In one model, a spark plug immersed in water generates a cavitating bubble that launches a spherical shock wave. This wave reflects from an ellipsoidal shaped reflector and refocuses at the distal focus of the ellipsoid, where the kidney stone is centered. The shock wave pulse has a jump in pressure of roughly 50 MPa (500 atmospheres) over a few nanoseconds, followed by a more slowly decaying decrease in pressure passing below atmospheric pressure before relaxing to ambient. This tensile portion of the wave is particularly important in kidney stone comminution since stones are composed of brittle material that does not withstand tensile stress well. Typically thousands of pulses are applied clinically (at a rate of 1 to 4 per second). The breakup process is not well understood and better understanding might allow clinical treatment with fewer pulses and less damage to the surrounding kidney.

Together with Kirsten Fagnan and Brian MacConaghy, two graduate students in Applied Mathematics, I have recently been collaborating with researchers at the Applied Physics Laboratory and the medical school at the University of Washington to develop a computational model of nonlinear elastic wave propagation in heterogeneous media that can be used to aid in the study of this process. Preliminary computations have been performed using linear elasticity in two-dimensional axisymmetric configurations in which a cylindrical test stone is aligned with the axis of the lithotripter ellipsoid. We are currently extending these computations to an appropriate nonlinear elasticity model, and also to three dimensional calculations for non-axisymmetric configurations.

We also hope to perform simulations useful in the study of extracorporeal shock wave therapy (ESWT), a relatively new application of lithotripter shock waves to treat medical conditions other than kidney stones, in which the goal is to stimulate tissue or bone without destroying it. For example, several recent clinical studies have shown that treating nonunions (broken bones that fail to heal) with ESWT can lead to rapid healing of the bone [26], [57], perhaps because it stimulates the growth of new vascular structure in regions where there is insufficient blood flow. Conditions such as tennis elbow, plantar fasciitis, and tendinitis have also been successfully treated with ESWL; see for example [32], [58].

In ESWL applications the shock wave is often focusing in a region where there is a complicated mix of bones and tissue. The interfaces between these materials cause significant reflection of wave energy. It is often crucial to insure that the shocks do not accidentally refocus in undesirable locations, such as nearby organs or nerves, which could potentially cause extensive collateral damage. Ideally one would like to be able

to use MRI data from a patient to set material parameters and run 3d simulations of the shock wave propagation in order to adjust the angle of the beam to achieve maximal impact in the desired region with minimal focusing elsewhere. Our current work is a first step in this direction.

Volcanic flows. My recent student Marica Pelanti developed a dusty gas model in which the Euler equations for atmospheric gas are coupled to another set of conservation laws for the mass, momentum, and energy of a dispersed dust phase [49], [50]. The two sets of equations are coupled together by source terms modeling viscous drag and heat transfer between the phases. The dust is assumed pressureless and requires special Riemann solvers, based on [45]. This model has been used to study the jets arising from high-velocity volcanic eruptions.

The speed of sound in a dusty gas is considerably less than the sound speed in the atmosphere. As a result, volcanic jets can easily be supersonic relative to the dusty gas sound speed, leading to interesting shock wave structures within an eruption column. Pelanti explored this for 2D axi-symmetric jets for both flat topography and for topography where the jet expands through an idealized conical crater. Similar work has been performed by Augusto Neri and Tomaso Esposti Ongaro in the Earth Sciences Department at the University of Pisa and we are now collaborating with them on some comparisons of our results.

We have also interacted extensively with researchers at the USGS Cascade Volcano Observatory (CVO) near Mount St. Helens (MSH), who study many aspects of volcanic flows and are charged with hazard assessment for MSH. After a long quiescent period, MSH became quite active again in October, 2004, and we worked with these researchers to try to produce a full three-dimensional model of pyroclastic flows over the topography of MSH in order to predict the possible impact of an eruption of various magnitudes. Extension of the dusty gas model to the full three-dimensional topography of MSH is underway, although it turns out there are many numerical and modeling issues still to be tackled.

Considerable data is available from the 1980 eruption that can be used to validate a code. In particular, there is a set of photographs and maps that show the direction in which trees were blown down when the initial pyroclastic blast from the eruption passed over the surrounding ridges. The blown-down trees created a snapshot of the velocity vectors in the leading edge of the flow. These exhibit complex flow patterns, such as recirculation zones on the lee side of ridges where the trees were blown down in the direction pointing towards MSH instead of away. We hope to eventually compare computed velocities from our simulations with these observations.

Richard Iverson and Roger Denlinger at CVO have also done extensive work on modeling debris flows, such as those that arise when water from melting glaciers on a volcano mixes with trees, boulders, and other debris, creating highly damaging and life-threatening flows [20], [21]. Their numerical work is based on equations similar to the shallow water equations but enhanced with more physics within the flow. They use Riemann solver techniques based in part on the wave-propagation framework, with the

additional solution of an elasticity problem within each Riemann solver to compute the local stress tensor within the debris. They encounter difficulties at the edge of the flow similar to the dry-cell problems that arise in tsunami modeling (discussed below) and we have collaborated with them on solving these problems. Denlinger has also recently modeled the great Missoula floods using similar techniques [22].

Tsunami modeling. David George and I have been developing a version of AMRCLAW capable of modeling tsunamis, including both their global propagation over large expanses of ocean and the inundation and run-up around small scale features at the level of individual beaches or harbors. We solve the shallow water equations on rectangular grids, in which each grid cell has an elevation value for the earth surface (which is called topography if it is above sea level, or bathymetry when underwater). The components of q are the fluid depth h and momenta hu and hv. Grid cells above sea level are dry ($h = 0$) and cells can become wet or dry dynamically as waves move along the shore. This approach avoids the need to model the shoreline as a separate interface, but developing a robust code based on this approach requires a Riemann solver that can deal well with both wet and dry states. The bathymetry comes in as a source term in the conservation laws. Away from shore the bathymetry is varying on a scale of several kilometers (the Indian Ocean is about 4 km deep, for example) whereas a tsunami propagating over the ocean is a few meters high at most. This difference in scales leads to difficulties that I will not describe here, and requires the use of some sort of "well-balanced" scheme in which the source term is incorporated into the Riemann solver. Though of small magnitude in the ocean, a tsunami may have a wavelength of more than 100 km and so the shallow water equations are an appropriate model. The propagation velocity is $\sqrt{gh}$, where g is the gravitational constant, and as they approach shore h decreases and the wave magnitude increases as the wavelength shortens, the same phenomenon observed in breaking waves on a beach. But in a tsunami the entire water column is set in motion by an uplifting of the ocean floor, whereas in wind-driven surface waves only the water very near the surface is moving. The enormous energy tsunamis carry gives them great destructive potential.

Adaptive mesh refinement is essential for this problem. We wish to propagate waves over the ocean, where grid cells several kilometers on a side can be used, and simultaneously predict the run-up around local features, where cell sizes of a few meters are desirable. Developing a well-balanced dry-state Riemann solver that works well in the context of AMR proved to be quite challenging and many difficulties appeared at the boundaries between grids at different levels. These could only be solved by some substantial reworking of the AMRCLAW code. The result is a special-purpose program that incorporates these algorithmic modifications and can now be applied to many tsunami scenarios. It is currently being tested by comparing predictions with measurements made at various places around the Indian Ocean in the wake of the 26 December 2004 Sumatra earthquake.

For this application we are working very closely with tsunami scientists. Our involvement in tsunami modeling arose out of a joint NSF grant with Harry Yeh in

civil engineering at Oregon State University and Joe Hamack at Penn State, who were doing wave tank experiments and related mathematical modeling that they wished to complement with numerical simulations. Since the Sumatra event, our focus has shifted to the larger scale, and the contacts we had already established in the tsunami modeling community proved invaluable. Many Tsunami Survey Teams traveled to the Indian Ocean and surveyed different parts of the coastline, measuring the run-up and inundation observed. Yeh was on a team that mapped the region near Chennai (Madras), India, and our initial validation work is focused on comparing predictions with his observations in this area. Unfortunately fine-scale bathymetry data is hard to come by and we have resorted to digitizing navigational charts to obtain some of the necessary data.

Figure 2 shows part of a simulation of the Indian Ocean tsunami, as described further in the caption. See the webpage [39] for color versions and movies, along with the computer program.

The program we have developed is a research code for this particular problem, but we intend to further improve it and ultimately make it available to the community. There is far more data available than we can compare against ourselves and we hope that other researchers will be able to use it for some of this work and publish the results. If our code does not work well or does not agree well with observations in some cases then we may need to revisit it, or perhaps others will make further improvements to it.

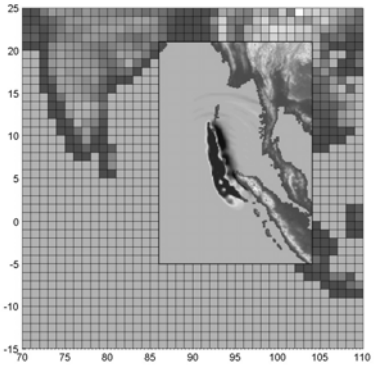
We hope that this code may ultimately be useful as a real-time tsunami prediction tool, and we are working with Vasily Titov and other scientists of the NOAA National Tsunami Hazard Mitigation Program in Seattle to compare our code with theirs and see how we can best complement their efforts (some of which are described in a recent Scientific American article [28]).

8. Conclusions

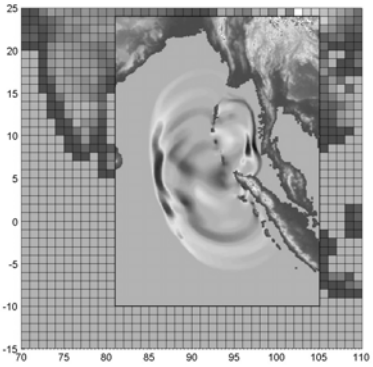
I have made a case that many aspects of scientific computing and software development should be viewed as inherently mathematical, and that mathematicians can play a very important role in computational science. I have also encouraged researchers in this area to produce reproducible research, in particular by making computer programs, not just software, available to others. I have presented some of my own research activities as a case study, though I do not claim it is the best example to follow.

I do hope, however, that the software we have produced will find wider use as one tool in this direction, both as a development environment for testing new methods and as a building block for solving problems in science and engineering. My hope is that others who develop methods or applications using this package will make their full code available on the web, particularly if it has been used to compute results that appear in publications.

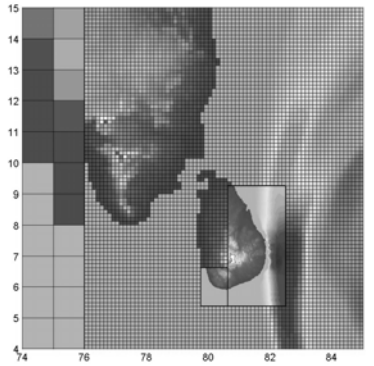
(a) Time 01:06:45 (525 seconds)



(b) Time 02:10:55 (4375 seconds)



(c) Time 02:43:00 (6300 seconds)



(d) Time 3:15:05 (8225 seconds)

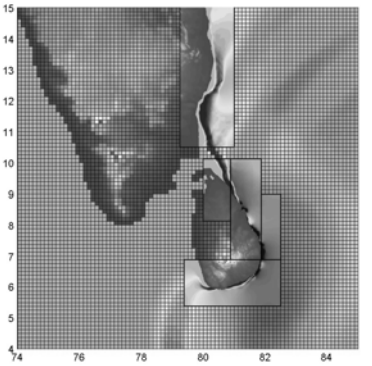


Figure 2. Propagation of the 26 December 2004 tsunami across the Indian Ocean. Units on the axes are longitude and latitude. The top two frames show the Bay of Bengal at two early times. A coarse grid is used where nothing is yet happening and the grid cells are shown on this "Level 1 grid", which has a mesh width of one degree (approximately 111 km). The rectangular region where no grid lines are shown is a Level 2 grid with mesh width 8 times smaller, about 14 km. Red represents water elevation above sea level, dark blue is below the undisturbed surface (see the webpage [39] for color versions of these images). Figure (c) shows a zoomed view of the southern tip of India and Sri Lanka at a later time. The original Level 1 grid is still visible along the left-most edge, but the rest of the region shown has been refined by a Level 2 grid. Part of Sri Lanka has been refined by Level 3 grids. The grid lines on Level 3 are not shown; the mesh width on this level is about 1.7 km, a factor of 8 finer than Level 2. Figure (d) shows a later time, as the wave diffracts around Sri Lanka, moving slowly through the shallow water in this coastal region. The calculation shown here was run on a single-processor 3.2 GHz PC under Linux and took about 40 minutes of wall time for the computation shown here. Movies of this simulation can be viewed on the webpage [39], which also contains pointers to finer grid calculations and more recent work on this problem.

Even for results that are not published, it would be valuable to have more examples and test problems available on-line than what is provided on the CLAWPACK web pages. Please let me know with a brief email if you have created such a page, or published a paper where CLAWPACK was successfully used.

I am also always interested to hear about problems that arise with the software or suggestions for improvements, though as an academic researcher with a small group of graduate students I cannot promise to provide as much technical support as I would like to.

The website [39] contains the codes used to generate the two figures in this paper, two very different examples of what can be provided in conjunction with a publication. The programs for Figure 1 are each less than a page of MATLAB, while the program for Figure 2 is about 13,000 lines of Fortran and also requires a large set of bathymetry and earthquake source data. The webpages also contain movies that illustrate the figures much better than the static versions shown in this paper, and links to other related work.

Acknowledgments. Numerous people read drafts of this paper and provided valuable comments and pointers to related work. In particular I would like to thank Marsha Berger, Donna Calhoun, Benjamin LeVeque, William LeVeque, and Nick Trefethen for their extensive comments, and David George for pulling together the necessary pieces for Figure 2.

References

[1] *Caltech center for simulation of dynamic response of materials*. http://csdrm.caltech.edu/, 2005.

[2] Bale, D., Rossmanith, J. A., and LeVeque, R. J., CLAWMAN software. http://www.amath.washington.edu/~claw/clawman.html.

[3] Bale, D. S., Wave propagation algorithms on curved manifolds with applications to relativistic hydrodynamics. PhD thesis, University of Washington, 2002. http://www.amath.washington.edu/~rjl/people.html.

[4] Berger, M., Adaptive mesh refinement for hyperbolic partial differential equations. PhD thesis, Computer Science Department, Stanford University, 1982.

[5] Berger, M., and LeVeque, R. J., Cartesian meshes and adaptive mesh refinement for hyperbolic partial differential equations. In *Hyperbolic problems. Theory, numerical methods and applications* (ed. by B. Engquist and B. Gustafsson), Vol. I, Studentlitteratur, Lund 1991, 67–73.

[6] Berger, M., and LeVeque, R. J., Stable boundary conditions for Cartesian grid calculations. *Computing Systems in Engineering* **1** (1990), 305–311.

[7] Berger, M., and Oliger, J., Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.* **53** (1984), 484–512.

[8] Berger, M., and Colella, P., Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.* **82** (1989), 64–84.

[9] Birkhoff, G., Mathematics and computer science. *American Scientist* **63** (1975), 83–91.

[10] Bolstad, J. H., Chan, T. F., Coughran, W. M., Jr., Gropp, W. D., Grosse, E. H., Heath, M. T., LeVeque, R. J., Luk, F. T., Nash,S. G., and Trefethen, L. N., *Numerical Analysis Program Library User's Guide: NAPLUG*. SLAC User Note 82, http//www.slac.stanford.edu/spires/... find/hep/www?r=slac-scip-user-note-082, 1979.

[11] Bornemann, F., Laurie, D., Wagon, S., and Waldvogel, J., The SIAM 100-digit challenge, a study in high-accuracy numerical computing. http://www-m3.ma.tum.de/m3old/bornemann/challengebook/index.html.

[12] Bornemann, F., Laurie, D., Wagon, S., and Waldvogel, J., *The SIAM 100-Digit Challenge, A Study in High-Accuracy Numerical Computing*. SIAM, Philadelphia, PA, 2004.

[13] Buckheit, J. B., and Donoho, D. L., WaveLab and reproducible research. http://www-stat.stanford.edu/~donoho/Reports/1995/wavelab.pdf, 1995.

[14] Calhoun, D. A., Colella, P., and LeVeque, R. J., CHOMBO-CLAW software. http://www.amath.washington.edu/~calhoun/demos/ChomboClaw.

[15] Calhoun, D. A., Helzel, C., and LeVeque, R. J., Logically Rectangular Grids and Finite Volume Methods for PDEs in Circular and Spherical Domains. In preparation; http://www.amath. washington.edu/~rjl/pubs/circles, 2005.

[16] Colella, P., et al., CHOMBO software. http://seesar.lbl.gov/anag/chombo/, 2005.

[17] Crowder, H., Dembo, R. S., and Mulvey, J. M., On reporting computational experiments with mathematical software. *ACM Trans. Math. Software* **5** (1979), 193–203.

[18] Deiterding, R., AMROC software. http://amroc.sourceforge.net/, 2005.

[19] Deiterding, R., , Construction and application of an amr algorithm for distributed memory computers. In *Adaptive mesh refinement - theory and applications* (ed. by T. Plewa), Lecture Notes in Comput. Sci. Engrg. 41, Springer-Verlag, Berlin 2005, 361–372.

[20] Denlinger, R. P., and Iverson, R. M., Granular avalanches across irregular three-dimensional terrain: 1. Theory and computation. *J. Geophys. Res.* **109** (2004), F01014.

[21] Denlinger, R. P., and Iverson, R. M., Granular avalanches across irregular three-dimensional terrain: 2. Experimental tests. *J. Geophys. Res.* **109** (2004), F01015.

[22] Denlinger, R. P., and O'Connell, D., Two dimensional flow constraints on catastrophic outflow of glacial Lake Missoula over three dimensional terrain. Invited abstract, 3rd International Paleoflood Workshop, Hood River, OR, 2003.

[23] Dongarra, J. J., Bunch, J. R., Moler, C. B., and Stewart, G. W., *LINPACK Users' Guide*. SIAM, Philadelphia, PA, 1979.

[24] Donoho, D. L., and Huo, X., BeamLab and reproducible research. *Int. J. Wavelets Multiresolut. Inf. Process.* **2** (4) (2004), 391–414.

[25] Forsythe, G. E., Galler, B. A., Hartmanis, J., Perlis, A. J., and Traub, J. F., Computer science and mathematics. *ACM SIGCSE Bulletin* **2** (1970), 19–29.

[26] Fritze, J., Extracorporeal shockwave therapy (ESWT) in orthopedic indications: a selective review. *Versicherungsmedizin* **50** (1998), 180–185.

[27] Garbow, B. S., Boyle, J. M. Dongarra, , J. J., and Moler, C. B., *Matrix Eigensystem Routines — EISPACK Guide Extensions*. Lecture Notes in Comput. Sci. 51, Springer-Verlag, Berlin 1977.

[28] Geist, E. L., Titov, V. V., and Synolakis, C. E., Tsunami: wave of change. *Scientific American* **294** (2006), 57–63.

[29] Gentleman, R., and Lang, D. T., Statistical analyses and reproducible research. Bioconductor Project Working Papers. Working Paper 2. http://www.bepress.com/bioconductor/paper2, 2004.

[30] Godunov, S. K., A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Mat. Sb.* **47** (1959), 271–306.

[31] Greenough, J. A., and Rider, W. J., A quantitative comparison of numerical methods for the compressible Euler equations: fifth-order WENO and piecewise-linear Godunov. *J. Comput. Phys.* **196** (2004), 259–281.

[32] Hammer, A. S., Rupp, S., Ensslin, S., Kohn, D., and Seil, R., Extracorporal shock wave therapy in patients with tennis elbow and painful heel. *Archives of Orthopaedic and Trauma Surgery* **120** (2000), 304–307.

[33] Jackson, R. H. F., Boggs, P. T., Nash, S. G., and Powell, S., Guidelines for reporting results of computational experiments. Report of the ad hoc committee. *Math. Programming* **49** (1991), 413–425.

[34] Knuth, D. E., Computer science and its relation to mathematics. Amer. Math. Monthly, 81 (1974), pp. 323–343.

[35] Knuth, D. E., *Algorithmic thinking and mathematical thinking*. *Amer. Math. Monthly* **92** (1985), 170–181.

[36] Langseth, J. O., Wave Propagation Schemes, Operator Splittings, and Front Tracking for Hyperbolic Conservation Laws. PhD thesis, Department of Informatics, University of Oslo, 1996.

[37] Langseth, J. O., and LeVeque, R. J., A wave-propagation method for three-dimensional hyperbolic conservation laws. *J. Comput. Phys.* **165** (2000), 126–166.

[38] Lee, C.-Y., Bard, J., Pinedo, M., and Wilhelm, W. E., Guidelines for reporting computational results in IEE Transactions. *IIE Trans.* **25** (1993), 121–123.

[39] LeVeque, R. J., http://www.amath.washington.edu/~rjl/pubs/icm06.

[40] LeVeque, R. J., CLAWPACK software. http://www.amath.washington.edu/~claw.

[41] LeVeque, R. J., CLAWPACK *User's Guide*. http://www.amath.washington.edu/~claw/doc.html.

[42] LeVeque, R. J., *Numerical Methods for Conservation Laws*. Lectures Math. ETH Zürich, Birkhäuser, Basel 1990.

[43] LeVeque, R. J., *Wave propagation algorithms for multi-dimensional hyperbolic systems*. *J. Comput. Phys.* **131** (1997), 327–353.

[44] LeVeque, R. J., *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts Appl. Math., University Press, Cambridge 2002.

[45] LeVeque, R. J., The dynamics of pressureless dust. *J. Hyperbolic Differential Equations* **1** (2004), 315–327.

[46] Liska, R., and Wendroff, B., Comparison of several difference schemes on 1D and 2D test problems for the Euler equations. *SIAM J. Sci. Comput.* **25** (2003), 996–1017.

[47] Mitran, S., BEARCLAW software. http://www.amath.unc.edu/Faculty/mitran/bearclaw. html.

[48] Moler, C., The origins of MATLAB. MATLAB *News & Notes*, December, 2004. http://www.mathworks.com/company/newsletters/news_notes/clevescorner/dec04.html.

[49] Pelanti, M., Wave Propagation Algorithms for Multicomponent Compressible Flows with Applications to Volcanic Jets. PhD thesis, University of Washington, 2005.

[50] Pelanti, M., and LeVeque, R. J., High-resolution finite volume methods for dusty gas jets and plumes. *SIAM J. Sci. Comput.*, to appear.

[51] Poludnenko, A. Y., Frank, A., and Mitran, S., Clumpy flows in protoplanetary and planetary nebulae. 2003. http://xxx.lanl.gov/abs/astro-ph/0310286.

[52] Poludnenko, A. Y., Frank, A., and Mitran, S., Strings in the Eta Carinae nebula: Hypersonic radiative cosmic bullets. 2003. http://xxx.lanl.gov/abs/astro-ph/0310007.

[53] Roache, P. J., *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, NM, 1998.

[54] Rossmanith, J. A., A wave propagation method for hyperbolic systems on the sphere. *J. Comput. Phys.* **213** (2006), 629–658.

[55] Rossmanith, J. A., Bale, D. S., and LeVeque, R. J., A wave propagation algorithm for hyperbolic systems on curved manifolds. *J. Comput. Phys.* **199** (2004), 631–662.

[56] Roy, C. J., Review of code and solution verification procedures for computational simulation. *J. Comput. Phys.* **205** (2005), 131–156.

[57] Schaden, W., Fischer, A., and Sailler, A., Extracorporeal shock wave therapy of nonunion or delayed ossesous union. Clinical Orthopaedics & Related Res. **387** (2001), 90–94.

[58] Schmitt, J., Haake, M., Tosch, A., Hildebrand, R., Deike, B., and Griss, P., Low-energy extracorporeal shock-wave treatment (ESWT) for tendinitis of the supraspinatus. *J. Bone & Joint Surgery* **83-B** (2001), 873–876.

[59] Schwab, M., Karrenbach, M., and Claerbout, J., Making scientific computations reproducible. http://sepwww.stanford.edu/research/redoc/cip.html.

[60] Shu, C.-W., High order ENO and WENO schemes for computational fluid dynamics. In *High-Order Methods for Computational Physics* (ed. by T. J. Barth and H. Deconinck), Lecture Notes in Comput. Sci. Engrg. 9, Springer-Verlag, Berlin 1999, 439–582.

[61] Smith, B. T., Boyle, J. M., Dongarra, J. J., Garbow, B. S., Ikebe, Y., Klema, V. C., and Moler, C. B., *Matrix Eigensystem Routines — EISPACK Guide*. Lecture Notes in Comput. Sci. 6, Springer-Verlag, Berlin 1976.

[62] Trefethen, L. N., *Spectral Methods in Matlab*. SIAM, Philadelphia, PA, 2000.

[63] Trefethen, L. N., *The SIAM 100-digit challenge*. http://www.comlab.ox.ac.uk/nick.trefethen/hundred.html, 2004.

[64] Trefethen, L. N., *Ten digit algorithms*. http://www.comlab.ox.ac.uk/nick.trefethen/ten_digit_algs.htm, 2005.

Department of Applied Mathematics, University of Washington, Box 352420, Seattle, WA 98195-2420, U.S.A.

E-mail: rjl@amath.washington.edu