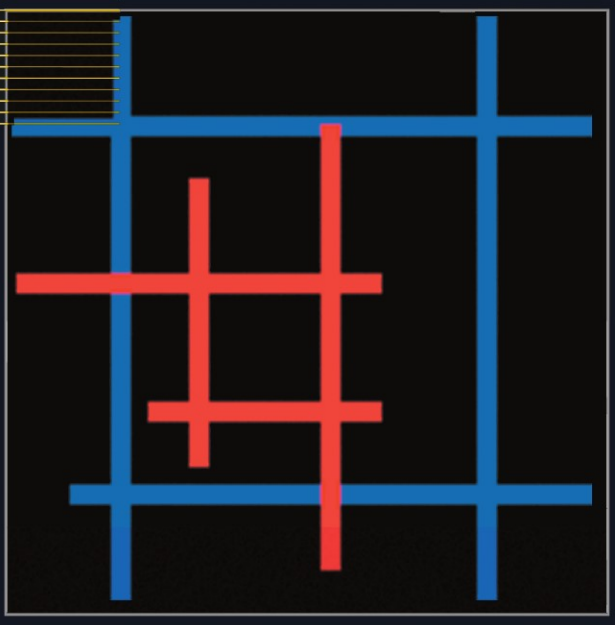
Lecture Notes in Computational Science and Engineering

41



Tomasz Plewa
Timur Linde
V. Gregory Weirs
Editors

Adaptive Mesh Refinement – Theory and Applications

Springer

Lecture Notes
in Computational Science
and Engineering

41

Tomasz Plewa
Timur Linde
V. Gregory Weirs
Editors

Adaptive Mesh Refinement – Theory and Applications

Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3–5, 2003

With 264 Figures and 36 Tables

Springer

Editors

Tomasz Plewa
Timur Linde
V. Gregory Weirs

ASCI Flash Center
Department of Astronomy and Astrophysics
University of Chicago
5640 S. Ellis Avenue
Chicago, IL 60637, USA
e-mail: tomek, linde, weirs@flash.uchicago.edu

Cover figure by Natalia Vladimirova

Library of Congress Control Number: 2004114481

Mathematics Subject Classification (2000): G18009, M12155, M1400x, M14026, M14042, M14050, P19005, P19013, P21026, P22006

ISSN 1439-7358
ISBN 3-540-21147-0 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springeronline.com

Printed in Germany

Cover production: *design & production*, Heidelberg
Typeset by the authors using a Springer TEX macro package
Production: LE-TEX Jelonek, Schmidt & Vöckler GbR, Leipzig

Printed on acid-free paper 46/3142/YL - 5 4 3 2 1 0

Preface

This volume presents a collection of overviews and research articles on recent advances in adaptive mesh refinement (AMR) methods and their applications presented at the *Chicago Workshop on Adaptive Refinement Methods* at the University of Chicago in September, 2003. Collection contains about 40 original articles grouped broadly into sections on methods, applications, and software.

The AMR approach was initially employed to solve hyperbolic systems of conservation laws. Consequently, mathematical approaches developed for hyperbolic equations provide a basis for modern AMR algorithms. Several research articles in this volume focus on improving the AMR capabilities for hyperbolic problems, such as treatment of complicated geometries or representation of discontinuous data on adaptive meshes. Others expand the concepts of AMR to applications which do not feature hyperbolic regimes, requiring methods for elliptic and parabolic equations. Finally, several papers discuss cross-disciplinary perspectives; some provide alternative approaches to structured AMR, while others propose ideas from fields from which AMR could potentially benefit. Many algorithmic aspects presented in this volume are common to applications across a wide range of disciplines and form a foundation AMR practitioners are uniformly familiar with.

For a number of scientific applications AMR techniques have become a method of choice while in others, a method of necessity. For example, the large disparity of scales found in astrophysics make AMR a basic component of numerical modeling of star formation, evolution of galaxy clusters, and growth of large scale structure in the Universe. A similar trend can be observed in numerical combustion where problems involving detonations and deflagrations demand resolving thin reaction zones. In many areas of space physics, atmospheric modeling and engineering, in particular aerospace engineering, simulations with adaptive mesh refinement have become routine practice. Uniting such diverse disciplines is the presence of narrow structures in complex fluid and solid media. They create ideal settings for exploiting the ability of AMR to accurately describe evolution of systems with multiple spatial scales. This poses questions concerning optimal discretization, accuracy and convergence of solutions on hierarchical meshes, and coupling of individual processes in com-

plex multi-physics applications. Papers presented in this volume review some of the recent advances made in many of these areas.

Computers and computer codes are behind all simulation results discussed in this book. Obtaining these spectacular results would not be possible without a major effort focused on design and development of modern scientific codes. AMR kernels are often central elements of these codes. We choose not to enter a debate on whether AMR kernels should dictate the way our codes manage data or vice versa. We will instead stress that in many situations the design of AMR-capable scientific codes determines their flexibility, portability, and types of applications that a particular code can be used for. Some of these design issues are of very technical nature; others present themselves as open and difficult topics for research. Still others are dependent on technological advances in computer hardware. Scalability and performance are the two enduring aspects scientists and engineers are concerned with. These concerns will inevitably become more serious as systems with tens of thousands processing elements become available. These systems will generate huge data sets and will stress storage, networking, visualization and analysis facilities, all of which will present challenges for future software and algorithms. It is certain, then, that existing numerical methods and implementations will undergo significant evolution to accommodate changes in a changing computing environment. Several papers in this volume discuss possible paths for this evolution. In particular, the volume opens with a discussion of whether the modus operandi in scientific computing is optimal or even efficient.

Adaptive mesh refinement methods have become an enabling tool in large-scale computing in engineering and science. It is no secret that many of these simulations are still difficult to conduct and analyze; some are unique computational experiments. These experiments are often of relatively little value to new adepts of scientific computing. It is essential that forthcoming developments in computational science lead to tools that are flexible and powerful and, at the same time, easy to use and maintain. In addition to themselves being products of research, such tools must enable groundbreaking scientific achievements by people not intimately familiar with algorithm or software development - they must become training and work platforms for future generations of scientists and engineers. With this vision in mind, we hope that this volume will inspire and motivate further development of adaptive mesh techniques and software.

Chicago,
October 2004

Tomasz Plewa
Timur Linde
V. Gregory Weirs

List of Participants

Name	Affiliation	Email
Alexakis, Alexandros	Univ. of Chicago	alexakis@flash.uchicago.edu
Asida, Shimon	Hebrew Univ.	sasida@phys.huji.ac.il
Baeza, Antonio	Universitat de Valéncia	antonio.baeza@uv.es
Balsara, Dinshaw	Univ. of Notre Dame	dbalsara@nd.edu
Banerjee, Robi	McMaster Univ.	banerjee@physics.mcmaster.ca
Barth, Timothy	NASA Ames	barth@nas.nasa.gov
Behrens, Joern	TU München, Germany	behrens@ma.tum.de
Bell, John	LBNL	JBBell@lbl.gov
Bergmans, Jeroen	Utrecht Univ.	J.Bergmans@astro.uu.nl
Berzins, Martin	Univ. of Utah	mb@sci.utah.edu
Bordner, James	UC San Diego	jbordner@cosmos.ucsd.edu
Bower, Richard	Durham Univ.	r.g.bower@durham.ac.uk
Caceres, Alvaro	Flash Center	acaceres@uchicago.edu
Calder, Alan	Univ. of Chicago	calder@flash.uchicago.edu
Calhoun, Donna	Univ. of Washington	calhoun@amath.washington.edu
Chin, Lee	ConocoPhillips	Lee.Chin@conocophillips.com
Choi, Dae-Il	NASA/GSFC and USRA	choi@milkyway.gsfc.nasa.gov
Colella, Phillip	LBNL	PColella@lbl.gov
Collins, David	UC San Diego	dcollins@physics.ucsd.edu
Dalla Vecchia, Claudio	Univ. of Durham	claudio.dalla-vecchia@durham.ac.uk
Dawes, Alan	AWE plc	Alan.Dawes@awe.co.uk
Debreu, Laurent	INRIA	Laurent.debreu@imag.fr
Deiterding, Ralf	CalTech	ralf@cacr.caltech.edu
Donat, Rosa	Universitat de Valéncia	donat@uv.es
Dubey, Anshu	Univ. of Chicago	dubey@flash.uchicago.edu
Dupont, Todd	Univ. of Chicago	t-dupont@uchicago.edu
Dwarkadas, Vikram	Univ. of Chicago	vikram@flash.uchicago.edu
Falgout, Robert	LLNL	rfalgout@llnl.gov
Falle, Samuel	Univ. of Leeds	sam@amsta.leeds.ac.uk
Fisher, Robert	LLNL	bobf@astron.berkeley.edu
Fournier, Aime'	NCAR	fournier@ucar.edu
Frank, Adam	Univ. of Rochester	afrank@pas.rochester.edu
Frazier, Njema	DOE/NNSA	njema.frazier@nnsa.doe.gov
Freis, Wolfgang	Flash Center	freis@flash.uchicago.edu
Fryxell, Bruce	Univ. of Chicago	fryxell@flash.uchicago.edu
Gallagher, Brad	ASCI Flash Center	jbgallag@flash.uchicago.edu
Gardiner, Thomas	Univ. of Maryland	gardiner@astro.umd.edu
Germaschewski, Kai	Univ. of New Hampshire	kai.germaschewski@unh.edu
Gisler, Galen	LANL	grg@lanl.gov

Name	Affiliation	Email
Groth, Clinton	Univ. of Toronto	groth@utias.utoronto.ca
Henshaw, William	LLNL	henshaw1@llnl.gov
Hensinger, David	SNL	dmhensi@sandia.gov
Howell, Louis	LLNL	nazgul@llnl.gov
Jorgenson, Philip	NASA Glenn	Philip.C.Jorgenson@nasa.gov
Jourdren, Herve	CEA/DAM-ILE de France	herve.jourdren@cea.fr
Klein, Richard	UC Berkeley and LLNL	rklein@astron.berkeley.edu
Kowal, Grzegorz	Jagiellonian Univ.	kowal@oa.uj.edu.pl
Kravtsov, Andrey	Univ. of Chicago	andrey@oddjob.uchicago.edu
Lamb, Don	Univ. of Chicago	d-lamb@uchicago.edu
Lan, Zhiling	Illinois Inst. of Technology	lan@iit.edu
LeVeque, Randall	Univ. of Washington	rjl@amath.washington.edu
Li, Shengtai	LANL	sli@lanl.gov
Linde, Timur	ASCI Flash Center	t-linde@uchicago.edu
MacNeice, Peter	Drexel Univ.	macneice@alfven.gsfc.nasa.gov
Matt, Sean	McMaster Univ.	matt@physics.mcmaster.ca
Micheletti, Stefano	Politecnico of Milan	stefano.micheletti@mate.polimi.it
Mignone, Andrea	Univ. of Chicago	mignone@flash.uchicago.edu
Mitran, Sorin	UNC - Chapel Hill	mitran@amath.unc.edu
Miyaji, Shigeki	Chiba Univ.	miyaji@astro.s.chiba-u.ac.jp
Mulet, Pep	Universitat de Valéncia	mulet@uv.es
Murphy, Gareth	Dublin Inst. Adv. Studies	gmurphy@cp.dias.ie
Nikiforakis, Nikolaos	Univ. of Cambridge	nn10005@damtp.cam.ac.uk
Norman, Michael	UC San Diego	mlnorman@ucsd.edu
O'Shea, Brian	UC San Diego	bwoshea@cosmos.ucsd.edu
Olson, Kevin	NASA/GSFC-GEST	olson@bohr.gsfc.nasa.gov
Otmianowska, Katarzyna	Jagiellonian Univ.	otmian@oa.uj.edu.pl
Pember, Richard	LLNL	pember1@llnl.gov
Pernice, Michael	LANL	pernice@lanl.gov
Philip, Bobby	LANL	bphilip@lanl.gov
Plewa, Tomasz	ASCI Flash Center	tomek@flash.uchicago.edu
Poludnenko, Alexei	Univ. of Rochester	wma@pas.rochester.edu
Powell, Kenneth	Univ. of Michigan	powell@umich.edu
Quirk, James	LANL	quirk@lanl.gov
Rider, Bill	LANL	rider@lanl.gov
Rijkhorst, Erik-Jan	Leiden Observatory	rijkhorst@strw.leidenuniv.nl
Riley, Katherine	Univ. of Chicago	kmriley@flash.uchicago.edu
Ritzerveld, Jelle	Leiden Observatory	ritzervj@strw.leidenuniv.nl
Robinson, Allen	SNL	acrobin@sandia.gov
Rosenberg, Duane	NCAR	duaner@ucar.edu

Name	Affiliation	Email
Rosner, Robert	ASCI Flash Center	r-rosner@anl.gov
Rudd, Douglas	Univ. of Chicago	drudd@oddjob.uchicago.edu
Samtaney, Ravi	PPPL	rsamtaney@pppl.gov
Shyue, Keh-Ming	National Taiwan Univ.	shyue@math.ntu.edu.tw
Siegel, Andrew	Univ. of Chicago	siegela@flash.uchicago.edu
Spotz, William	SNL	wfspotz@sandia.gov
St-Cyr, Amik	NCAR	amik@ucar.edu
Takahashi, Keiko	Earth Simulator Center	takahasi@jamstec.go.jp
Truran, James	Univ. of Chicago	truran@nova.uchicago.edu
Van Straalen, Brian	LBNL	bvstraalen@lbl.gov
Varniere, Peggy	Univ. of Rochester	pvarni@pas.rochester.edu
Velarde, Pedro	Instituto de Fusin Nuclear	pedro@din.upm.es
Vladimirova, Natalia	Univ. of Chicago	nata@flash.uchicago.edu
Weaver, Robert	LANL	rpw@lanl.gov
Weirs, Greg	ASCI Flash Center	weirs@flash.uchicago.edu
Whalen, Daniel	UC San Diego	dwhalen@cosmos.ucsd.edu
Xu, Zhiliang	SUNY at Stony Brook	xuzhi@ams.sunysb.edu
Ziegler, Udo	UZ	uziegler@aip.de

Contents

Part I

Scientific Computing

Computational Science
"Same Old Silence, Same Old Mistakes"
"Something More Is Needed . . ."

James J. Quirk

Los Alamos National Laboratory, quirk@lanl.gov

Today it is fashionable to portray computation as the third leg of science, the other legs being the classical disciplines of experiment and theory. But in the rush to promote computational science's strengths, a blind eye is often turned to its weaknesses. This paper aims to increase awareness of a number of key deficiencies in the hope that the community can galvanize itself and tackle the identified issues head on. Specifically, the thesis to be developed here is that software automation could be used to package worked examples — in the form of dynamic electronic documents — that would allow interested parties, from different backgrounds, to communicate more effectively than at present. The hope, by making work easily repeatable, is that practical expertise can be properly archived. Currently, many avoidable mistakes are repeated time and time again as the mistakes do not lend themselves for journal publication and so go unrecorded.

1 Mission Impossible

The subtitle — "Same Old Silence, Same Old Mistakes" — is taken from a newspaper article[Col97] that highlights the billions, upon billions, of dollars wasted each year on failed software projects. The article draws insightful analogies between software failures and engineering disasters and it argues the case for more openness in owning up to software mistakes. Although written for a general audience, the article offers serious food for thought for the computational science community, especially now that many algorithms rely more on clever programming than on sophisticated mathematics. A reader of the article, G. N. G. Tingey, sums the situation up best when he writes, in a follow-up letter: *the usual reasons for classical engineering failures are ignorance, arrogance and pride, including the shooting of the bearer of bad tidings.* In the early part of the 21st Century, computational science appears to be failing, to some degree, on all these counts.

The challenge for this author is to rise above the anecdotal, with substantive evidence of community failings. Otherwise, you will be within your rights to serve

charges of ignorance and arrogance. In many ways this represents a mission impossible as it is notoriously difficult to get technical experts to agree to anything, en masse. A case in point at this adaptive mesh refinement (AMR) workshop was the willingness with which delegates criticized smooth particle hydrodynamics[Mon92] while engaging in circular arguments regarding the merits of patch-based AMR[BC89] versus cell-based AMR[PRQ92]. The view adopted here is that such arguments are moot, for algorithmic advantages depend on the actual application and they can be undermined by implementation details, as well as operator error.

Like mainstream society, computational science does not have a good grasp on software engineering. If it did, by now the idea of a standalone simulation code would be quite dead. Instead there would be sufficient standardization of operation that users could, if they so wished, run multiple algorithms on the same problem, from within some universal run-time system. This would allow rigorous head-to-head comparisons to be made for minimal effort and it would soon become clear which schemes performed well on which problems. Of course, the practicality of such a system lies in its many details.

Moving swiftly to the substantive elements of this paper. Section 2 revisits a blast wave simulation from 1991[Qui91] to show how increases in computing power, over the last decade, warrant a rethinking of how computational business is done. This theme is developed in Sect. 3 with an examination of the American Institute of Aeronautics and Astronautics (AIAA) policy on numerical accuracy[AIAA]. The policy was drafted in response to *considerable concern with the quality of published numerical results* and it is a solid attempt to raise computational standards. But in the spirit of this paper, the most illuminating aspect of the AIAA policy is the absence of any recognition that software could help power standards.

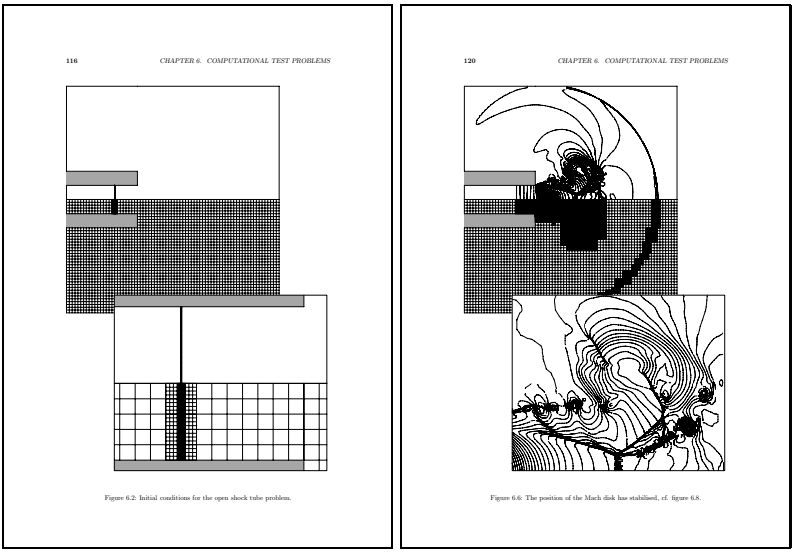
The aerospace industry is a mature discipline which, over the years, has been shaped by engineering failures, such as the ill-fated *Comet*[RAE54] and the *Challenger* disaster[RPC86]. Thus the AIAA policy can be thought of as raising the concept of *computational technique* to complement established practices like *wind tunnel technique*[PH52]. The usual difficulty, however, of defining technique is that it is a catch-all for all manner of hard-won, tricks-of-the-trade. But in the case of computations, Sect. 4 shows that software technologies do exist that would allow sound technique to be demonstrated using automated, worked examples. Again, obstacles to progress include software ignorance, software arrogance and software pride. Sect. 5 covers these sensitive issues from the perspective of the DOE Advanced CompuTational Software Collection[DM03].

Section 6 concludes with a set of recommendations. The way forward, essentially hinges on recognizing that software tools and management can be used to facilitate communication. And improved communication is needed for computational science to fulfill its potential.

2 A Blast From The Past

The AMR simulation shown in Figure 1 was computed July 1990[Qui91]. It took just over 12 hours to run on a *Sun SPARCstation 1*. In 2003 it can be run on an *IBM T30* laptop in a shade over two minutes. The specifications for the two machines are given in Table 1. Such dramatic increases in desktop computing-power begs the question:

> *What should the computational science community be doing over and above scaling up the sizes of the problems it computes?*



(a) **(b)**

Fig. 1. Two snapshots from a simulation of a blast wave escaping from an open-ended tube: (a) initial conditions; (b) late-time flowfield.

Table 1. Comparison of the computing power on the author's desk.

Sun SPARCstation 1, 1990	*IBM T30*, 2003
8 MBytes Ram	1000 MBytes Ram
100 MBytes Disk	60000 MBytes Disk
33 MHz Fujitsu MB86901A	2400 MHz Intel Pentium 4M
$\approx$ 12.5 MIPS, 1.4 Mflops	$\approx$ 4500 MIPS, 1200 MFlops

It is certainly important for computational science to push the envelope with ever larger and larger simulations, but a measure of introspection is called for to uncover the dangers of operating in a bigger-is-better fashion. If nothing else, it results in a throwaway culture where work is here today, gone tomorrow. In 1990, the above blast wave simulation turned one head sufficiently that its owner stole a poster of the simulation from the 12th Intl. Conf. on Numerical Methods in Fluid Dynamics, held in Oxford. But in this workshop proceedings, with today's computing power, the simulation will barely attract a second glance. By the same token, the largest simulation in this proceedings will look paltry come 2016.

The challenge for computational science is to migrate to a scale invariant view of the world where simulations, large and small, are judged on their strict technical merits. Such an approach would allow pertinent numerical observations to stand the passage of time. Thus the technical gems in this proceedings will continue to sparkle in 2016 even though hardware developments will have reduced the recorded research projects down to homework exercises for students. If computational science cannot rise to this challenge, each new generation will ignore much of the work of earlier generations on the mistaken grounds that it has no relevance to the cutting-edge problems of the day. As a result, many good ideas will be lost and others will be rediscovered, often as pale imitations of the original.

If the preceding argument sounds pessimistic, many of the issues discussed at this AMR workshop were covered in depth by an earlier workshop held in Hampton, Virgina, 1994[STV95]. This is perhaps not surprising as several individuals from the 1994 workshop attended the present one. Nevertheless the bulk of the participants in Chicago were not at Hampton. Therefore, as an experiment, readers of this proceedings should dig out the earlier one to see how the AMR community has developed over the last decade. In particular, the 1994 proceedings has a very articulate discussion on the pro's and con's of black-box CFD codes. It also highlights the technical need for temporal refinement. And the very cover of the proceedings shows a computational grid that captures the challenge of designing refinement criteria. Namely, it is very important to be able to distinguish between a small local error that has a large global effect and a large local error that has no global effect.

This author's contribution to the 1994 workshop ended with the observation that common ground must be found between theoreticians and the practical exponents of mesh refinement before any real progress can be made in eliminating the heuristic elements from AMR algorithms. This observation provided impetus for the development of a programming system [AMRITA] that would allow interested parties to engage in computational conversations using self-substantiating, dynamic documents. As the programming approach is general purpose, not much more will be said here on AMR. Instead, the motivation for such a system will be examined from the perspective of the AIAA policy on numerical accuracy.

3 AIAA Numerical Accuracy

Any discipline that undergoes the rapid development exhibited by computational science is likely to suffer growing pains. For example, in response to *considerable concern with the quality of published numerical results* the AIAA felt compelled to issue, January 1998, the editorial policy[AIAA]:

> *The AIAA journals will not accept for publication any paper reporting (1) numerical solutions of an engineering problem that fails adequately to address accuracy of computed results or (2) experimental results unless the accuracy of the data is adequately presented.*

The AIAA policy, like this paper, does not deny the existence of the many solid computational investigations that have been published. But it recognizes that poor work acts to undermine the credibility of computations as an engineering tool. In the case of computational fluid dynamics (CFD), critics underwhelmed by the rigour of many investigations refer to CFD as *coloured fluid dynamics*. The knee-jerk response, adopted by many CFD'ers, is to label such critics ignorant. But the more measured response is to examine work practices, from the ground up, and take remedial action where it is needed.

In this regard, the effectiveness of the AIAA policy hinges on their *chosen* definition of accuracy[AIAA]:

> *The accuracy of the computed results is concerned with how well the specified governing equations in the paper have been solved numerically. The appropriateness of the governing equations for modelling the physical phenomena and comparison with experimental data is not part of this evaluation. Accuracy of the numerical results can be judged from grid refinement studies, variation of numerical parameters that influence the results, comparison with exact solutions, and any other technique the author selects.*

The background to this definition, such as why it is careful to preclude comparisons with experimental data, are discussed by Roache[Roa98]. Here it is more pertinent to ponder why the recommended activities are often missing. The usual charges of ignorance and arrogance apply, as does sloth.

In the case of the above blast-wave simulation, the emphasis was placed solely on computing the largest simulation possible, given the available computing resources. Such an approach, although not unusual, tends to undermine the activities the AIAA is trying to promote. For instance, there was insufficient disk space to store the results from a credible parameter study of the blast-wave phenomena, and it was not possible to perform a reliable grid refinement investigation as the computed resolution, which did not yield converged results, could not be improved upon.

Operating on the limit encourages the conceit that a simulation is good simply because it exceeds the scale that anyone else can compute. But a concrete example is needed — taken from the world of shock-capturing — to show that there is a much more fundamental, community-failing that acts to undermine computational rigour.

3.1 A Cautionary Tale

Godunov's theorem[Lan98] — *linear monotonicity-preserving methods are first-order accurate at best* — is an important result, in shock-capturing circles, for it indicates that a step function cannot be propagated numerically, using a fixed coefficient stencil, of second-order or higher, without introducing unwanted oscillations into the computational solution. The theorem, published in 1959, is incontrovertible mathematical fact, and no amount of software engineering or parallel computing can change it.

At first sight Godunov's theorem seems to preclude the possibility of generating accurate, well behaved simulations of flows that contain shock waves. Godunov's genius, however, was to recognize that the theorem could be circumvented by using computational stencils that employed variable coefficients. This led him to devise a method based on solving Riemann problems[Lan98].

Unfortunately for the community, Godunov's pioneering work was published in Russian[God59] and lay largely forgotten until in 1979 van Leer proposed a higher-order extension known as MUSCL[Van79]. But given the then available computing power, the operation-count of solving Riemann problems made the method unattractive. In 1981, Roe realised that a linearized Riemann solver could be used in place of an exact solver, reducing the operation count down to more manageable proportions[Roe81]. Nevertheless, the popularity of the approach did not really take off until the mid-to-late 1980's.

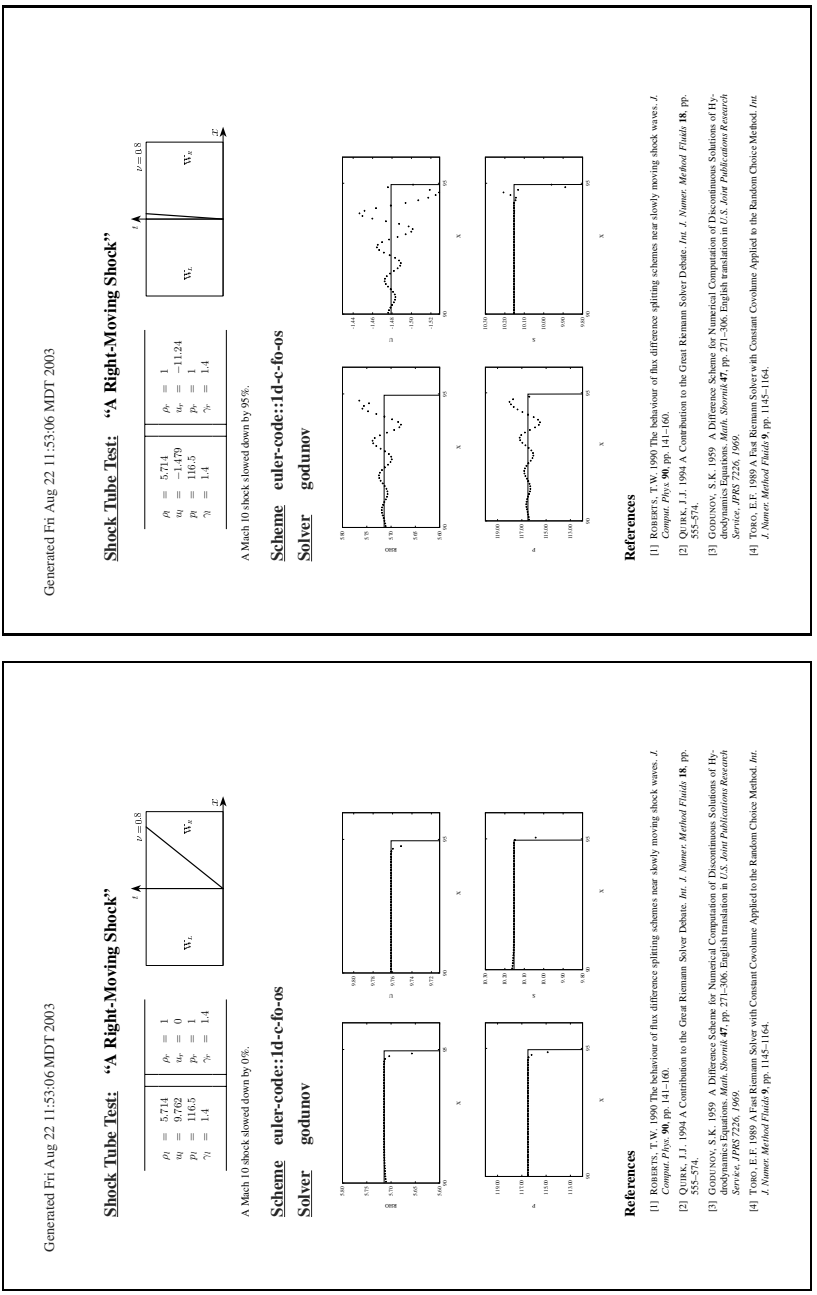
The thirty years needed for Godunov's ideas to reach general acceptance is disappointing, but it is not unusual for revolutionary ideas to spend years in the wilderness. The tragedy, at least in the eyes of this author, came to light in 1990 when it was reported that Godunov-type methods suffer from low-frequency, post-shock oscillations[Rob90], see Figure 2. Thereby negating the perceived advantage of the approach over other methods.

The use of the word *tragedy* may seem overly melodramatic, but the fact remains that it took the community thirty years to discover that Godunov-type methods are not Galilean invariant when applied to the Euler equations [1]. A collective-performance of which the CFD community should not be proud, given the literal hundreds of people who came into contact with the numerical approach. If it were held, an inquest into the tragedy might ask:

What went wrong?
Where did CFD rigour break down?

It should be stated up front that the theory behind Godunov-type methods is not wrong. It just happens to paint an incomplete picture when applied to non-linear systems of equations. Thus the criticism above is not aimed at Godunov, van Leer and Roe whose pioneering works are beyond reproach and remain required reading. The criticism is aimed squarely at how the computational community does business.

[1] An earlier work[WC84] alluded to the failing, in passing, and so it can be argued the delay was *only* twenty five years.



(a) Monotone results in laboratory reference frame.

(b) Oscillatory results close to shock reference frame.

Fig. 2. Godunov's method is not Galilean invariant. It took the shock-capturing community thirty years to discover that a simple shift in reference frame was sufficient to break a supposedly robust method. Computational science needs to develop a rigorous approach to testing to identify where theory is weak.

3.2 Automated Testing

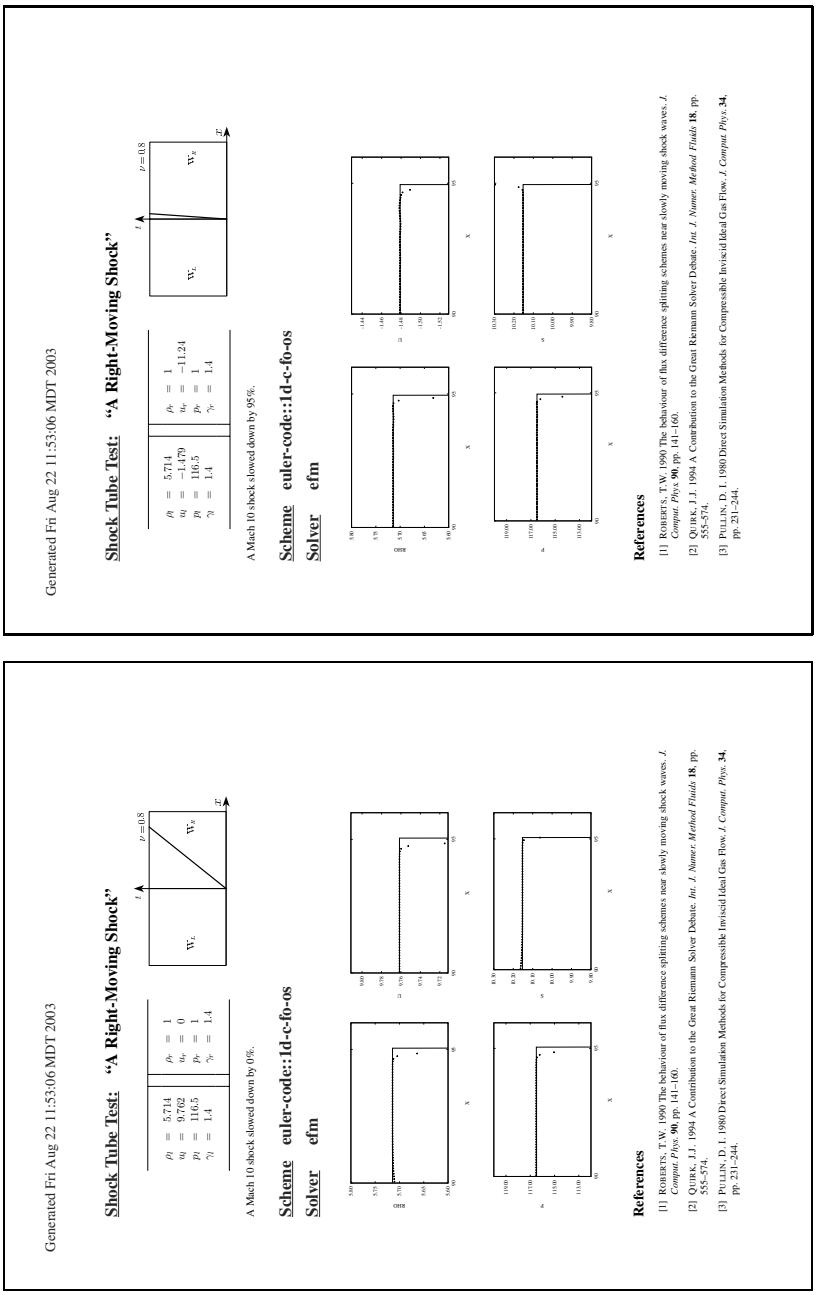
Given that CFD theory is incomplete, commonsense would suggest that weaknesses could be identified via a rigorous programme of testing. Then, once found, a weakness could be plugged with a heuristic stop-gap, pending the development of new theory to provide a more lasting solution. Unfortunately, there are several reasons why the research community does not operate this way. At the practical level, the community apparently considers testing to be a labour intensive activity without much intellectual reward. At least it sticks with the hit-and-miss approach of testing a small sample of a large parameter space by hand, when it could just as well develop software tools and management to both widen the test-net and reduce the work involved.

Figure 3, for instance, shows that not all shock-capturing schemes suffer the failing shown in Figure 2. Taken in isolation these results can be used to argue that scheme *B* is superior to scheme *A*, and the literature is littered with papers that make claims and counter claims based on isolated numerical evidence. The pragmatic view adopted by this author is that all numerical schemes have strengths and weaknesses and it is just a matter of selecting the right test cases to map out their respective flight envelopes![Qui94].

A Riemann problem has an uncountable infinity of initial conditions and so automated testing is never going to replace shock-capturing theory. But automated testing could be used to remove operational latitudes so as to allow interested parties to exchange numerical results, unequivocally, with the aim of developing consensus of opinion on matters not covered by theory. Today, the failing shown in Figure 2 is widely acknowledged, but it was stumbled across by accident in a nozzle-flow calculation[Rob90]. Similarly, the blast-wave simulation, shown earlier, unearthed another numerical failing, purely by accident. This failing is also now widely acknowledged, but at the time of its discovery it was routinely dismissed, by sceptics, as a coding error.

Figure 4 is a zoom of Figure 1 to show the numerical failing: odd-even decoupling occurs along the length of a strong shock aligned with the computational grid. The failing was exasperated by the AMR algorithm used for the simulation, but the failing is inherent to Godunov's method and can be reproduced on a uniform mesh. Now the write-up of this failing[Qui94] highlights a number of fundamental weaknesses of computational science.

Starting with the review of the manuscript: one reviewer refused to provide a review on the grounds that the reported work was scurrilous; the second reviewer thought that the work was significant and recommended that it should be published with no modifications whatsoever. Readers of the paper appear equally polarized: many loathe it; many praise it. If computational science were a truly rigorous discipline, there would be no room for such polarized views. Party *A* could show party *B*, via an unbroken sequence of logical steps, some viewpoint, and if the logic were flawed, *B* could respond with a counter viewpoint. Then, in an ideal world, both parties would iterate until some consensus was reached.



(a) Monotone results in laboratory reference frame. **(b)** Monotone results close to shock reference frame.

Fig. 3. Pullin's method is Galilean invariant. The two test-sheets shown here, and those shown in Figure 2, are dynamic *PDF* documents. Embedded withing the *PDF* is active code that can replicate the reported work. Thus interested parties can retread the investigations for no effort, and if mistakes are found they can be corrected.

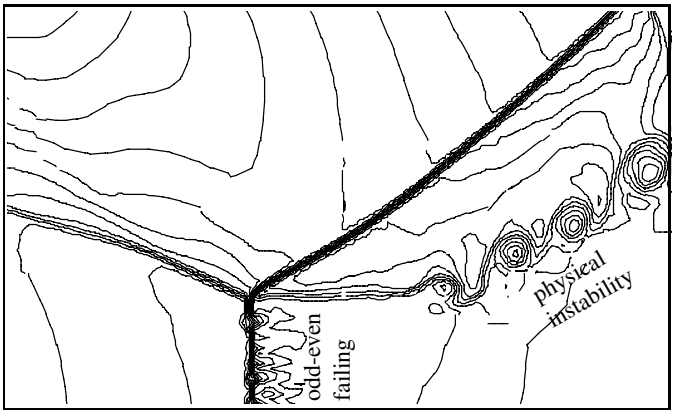


Fig. 4. The blast-wave simulation in Fig. 1 helped unearth the odd-even decoupling, numerical failing described in [Qui94].

In the real world, computational science is a complex amalgam of numerical analysis, physics, computer science, and common sense (not to mention personality clashes). As such, it is often difficult to present a watertight case. The AIAA policy ends with the diktat:

> *Finally, the accepted documentation procedures for a technical investigation must be used. For computational papers, the author must provide an adequate description of the numerical solution procedure, if not documented elsewhere. In addition, the complete governing equations must be specified with sufficient detail along with the input parameters to the code so that a reader could reproduce the results of the paper.*

but it is far easier for a sceptical reader to reject a paper, out of hand, than to go to the trouble of reworking its details as part of a rigorous evaluation. Thus the weakness of [Qui94] is that it could not provide concrete evidence in a sufficiently digestible form for all readers to take its message on board.

Now, if no consensus can be reached on Riemann solvers, a class of algorithm rooted in numerical analysis, traditional reporting methods are doubly inadequate for reaching consensus on software bound algorithms. The community needs more than one AMR scheme. But one AMR scheme per researcher, as often seems the case with Riemann solvers, is not a sustainable route. A more co-operative approach is called for now that the software element of computational science has grown beyond that which individual researchers, or even individual groups, can be expected to maintain.

The next section will demonstrate the concept of a self-substantiating, self-replicating electronic document that would allow the AIAA diktat to be met in the literal sense of allowing a reader to reproduce a paper, for no effort. The idea of such automation is to eliminate subjective differences of opinion. Party A and party B can settle their technical differences, substantively, via the exchange of concrete examples.

4 Document Engineering

The AIAA policy arises from a desire to improve computational standards. But the AIAA vision of *accepted documentation procedures* has not kept pace with what software technologies would now allow. Specifically, *Adobe*'s portable document format [PDF] makes possible the construction of electronic documents that enable the AIAA diktat to be met to the letter. Here Figure 2 and Figure 3 appear as static images, but in reality they are dynamic *PDF* files that would allow you to repeat the reported work, for the sake of clicking on the time-stamps that appear in the top-left corner of each page. They also contain material to help you *understand* the work performed.

PDF is a binary format and it is intended to be used as a means of archiving information rather than as a pure mechanism for placing ink on the page, such as *PostScript*. Specifically, a *PDF* file contains a hierarchy of objects along with a reference table that gives the positions of the objects within the file. Each object has a type to indicate its purpose. For example, *content stream* objects paint ink on the page, *file stream* objects embed files, and *annotation* objects introduce interactive elements. The time-stamps in Figure 2 and Figure 3 are annotations that unpack and run an embedded file stream. The file stream contains the program source used to generate the document, which includes the code needed to run the Riemann problems. Thus the time-stamps provide a means for cloning the reported work.

It is hard in a static document, such as these workshop proceedings, to get across the details of how these self-replicating, test-sheets are created. But as a first step, it should be appreciated that a new field is emerging, on the back of formats like *PDF*, known as document engineering[DE], that views the creation of electronic documents as a form of programming. For example, the text you are reading here is typeset using [TEX]. But this contribution was not created in the usual fashion of typing TEX instructions into a file. Instead a program was written to output the TEX instructions to a file. Introducing a level of indirection, in this fashion, allows the manuscript to be intertwined with active code to generate information on-the-fly.

To understand the motivations for this programming approach, consider Figure 5. It shows how the *PDF* version of this manuscript appears within *Acroread*. The document can be viewed as any other *PDF*. But it can enter a viewing mode, the one shown here, which is more like a windowing system. This gives access to a range of auxiliary material. For example, the *FAQ* outlines the document's operation, and it alerts the reader to the fact that the *PDF*, although it can be read using regular *Acroread*, requires custom software to be able to extract and run files on-the-fly. This system software can be downloaded using the *Amrita* button. It is needed because *Acroread*, unlike *Acrobat*, does not have the ability to extract embedded files. It is also needed to safeguard against computer viruses. Once the system software is installed the *ViewSource* button gives access to the program used to create the document, and buttons in the text give access to the embedded examples.

The aim here is not to sell [AMRITA] as a turn-key, document preparation system. It is to give a feel for how so-called literate-programs could be harnessed to drive the AIAA policy on numerical accuracy. For example, the numerical parame-

ters shown in Figures 2 and 3 are guaranteed to be correct as they are typeset directly from the values used for the calculations, c.f.[AIAA]:

> *Accuracy of results from a validated code must still be established to show that proper input parameters have been used with the code.*

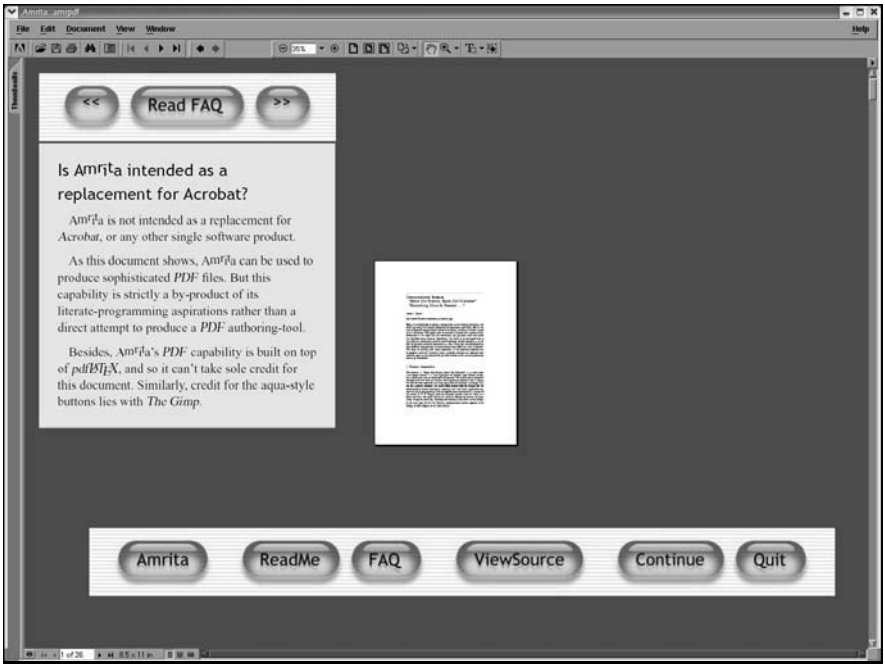


Fig. 5. The *PDF* version of this paper includes the actual source used to create the document. It also includes a FAQ that covers issues ranging from computer security through the operation of the document's Amrita-powered features that will allow you to try out the reported work first hand.

4.1 Literate Programming

The term *literate programming* was coined by Knuth[Knu92]:

> *I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: "Literate Programming."*
>
> *Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.*
>
> *The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style.*

> *Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means.*
>
> *He or she strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that reinforce each other.*

The approach was intended to overcome the all too common situation where the documentation of a program fails to reflect its true workings and so is rendered useless, or even dangerous. But, while many programmers agree that the one thing worse than no documentation is wrong documentation, literate programming has not caught on.

The incongruity, for computational science, is that Knuth's typesetting system [TEX] is widely used for reporting scientific work, including these proceedings, but the use of his literate programming system [CWEB], which employs TEX, is almost non-existent[2]. It is not, however, difficult to see why *CWEB* has been slow to catch on, for it literally turns programming inside out. Whereas a classical program consists primarily of computer code with a sprinkling of comments, a *CWEB* program is primarily documentation with a sprinkling of computer code. Nevertheless, the key point Knuth is making with literate programming is straightforward[CWEB]:

> *The philosophy behind CWEB is that an experienced system programmer, who wants to provide the best possible documentation of his or her software products, needs two things simultaneously: a language like TeX for formatting, and a language like C for programming.*
>
> *Neither type of language can provide the best documentation by itself; but when both are appropriately combined, we obtain a system that is much more useful than either language separately.*

With the development of *PDF* and the power of today's desktop computers, the time is truly ripe for significantly better documentation of software. The technique employed here[AMRITA] generalizes the co-operative idea that underpins *CWEB*. Program folds allow arbitrary languages to be mixed re-entrantly, such that the linguistic whole is greater than the sum of its parts.

4.2 Program Folds

The *Occam Programming System* introduced program folds as a means of outlining computer code[BK89]:

> *Folding is analagous to taking a document with headed paragraphs and then folding it so that the text is hidden, leaving only headings visible.*

They allow a code's structure to be digested without getting bogged down by its details. Figure 6 (a), for instance, shows the outlined source for this paper. In a fold-compliant editor, the lines starting `...` act like bookmarks in a *PDF* document. Thus click on `... document engineering` and the view narrows to the fold shown in Figure 6 (b). Then click on `... program folds` and the view narrows to Figure 6 (c), which shows the source for this page.

[2]The situation is particularly ironic as Knuth rates *CWEB* above TEX.

(a)
```
Latex2eHead {
    ... parameters
}
fold::latex { jjq@AMR2003
    ... title
    ... abstract
    ... mission impossible
    ... a blast from the past
    ... numerical accuracy
    ... document engineering
    ... doe acts
    ... something more
    ... references
}
LatexTail
Latex clean=no
```

(b)
```
fold::latex { document engineering
    \section{Document Engineering}
    \label{sec:document-engineering}
    ... leadin
    ... literate programming
    ... program folds
    ... domain specific languages
}
```

(c)
```
fold::latex { program folds
    \subsection{Program Folds}
    The {\sl Occam Programming System} introduced program folds
    as a means of outlining computer code\cite{occam}:
    ... quote from occam2 programming
    They allow a code's structure to be digested without getting
    bogged down by its details. Figure~\ref{fig:thispaper} (a),
    for instance, shows the outlined source for this paper.
    In a fold-compliant editor, the lines starting {\tt ...}
    act like bookmarks in a {\sl PDF} document.
    Thus click on {\tt ... document engineering} and
    ... etc. etc. etc.
}
```

Fig. 6. The program that generates this paper: (a) outline of the whole program; (b) the section on document engineering; (c) the source for this page.

In *Occam*[BK89] a program fold starts with {{{ and ends with }}}. The present program folds are multi-lingual. A program fold starts with:

`fold::`*parser* { *optional comment*

where *parser* identifies the agent responsible for parsing the fold-body, and the *optional comment* gives an indication of the fold's purpose. A program fold is ended by the first } that column-matches the start of the fold. The present program folds are also re-entrant. Thus the *fold::latex* shown in Figure 6 can contain *fold::amrcc* to compile a *C* program, and *fold::amrcc* can contain *fold::latex* to typeset a document.

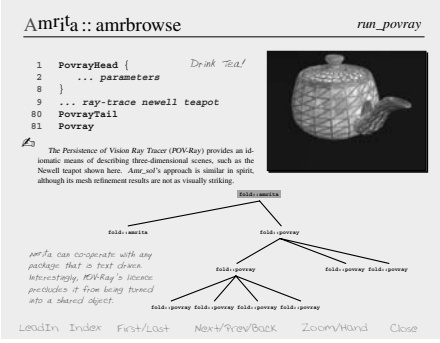
The design idea is that any one programming activity can be folded inside any other programming activity. Thus, as shown in Figure 7, the approach encompasses the traditional view of a code as well as Knuth's concept of a literate program. Nesting folds gives rise to the concept of a fold-tree. Figure 8, for example, shows two

nodes from a fold-tree that ray traces a photo-realistic image. Storing a program in a tree structure facilitates manipulating the component code, on-the-fly. Thus the code fragments in Figure 6 are guaranteed to be correct as they are obtained, automatically, via formal transformations.

(a)
```
fold::lang'A { activity one
    fold::lang'B { activity two
    }
}
```

(b)
```
fold::lang'B { activity two
    fold::lang'A { activity one
    }
}
```

Fig. 7. Program folds allow different type of work to be nested: (a) could be a traditional program, predominantly code with a sprinkling of comments; (b) could be a literate program, predominantly documentation with a sprinkling of code.



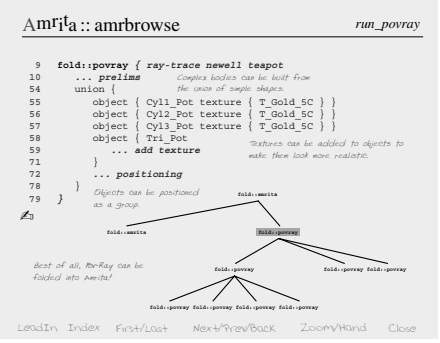


Fig. 8. Two nodes from a fold-tree that ray traces a teapot: (a) the root node; (b) the node that defines the teapot. The code annotations are housed in information nodes that have been pruned from the depicted tree. They can contain active code to generate material dynamically. For instance, the ray traced image is obtained by running the program. Thus the documentation is guaranteed to be correct.

4.3 Domain Specific Languages

Domain specific languages allow common patterns of work, within some field, to be programmed efficiently, and precisely, using constructs targeted direcly at the job in hand. For example, [TEX] provides idioms for typesetting and [POVRAY] provides idioms for ray tracing. The advantage of an idiomatic approach, over a general purpose language, and its relevance here, is that it greatly reduces the documentation burden on programmers. Code author and code reader are bound by a common interest which has a specialist vocabulary documented in a central dictionary, and only the structure of an individual program need be described, for the code author can assume that the code reader can use the dictionary to look up the underlying details.

Figure 9 shows a mesh refinement example. Fully folded none of its AMR idioms are visible, but the structure of the simulation is perfectly clear. Here the use of a domain specific language allows a separation between the scheduling of computational work and the computational agents that perform the work. In effect the fold-tree represents a wish list, and when it is parsed the wish list is translated to a low-level instruction stream that it is then fed to specialist agents to carry out the work as they see fit. This instruction stream is analagous to *DVI* in TEX, or an *XML* schema. Therefore although a fold-tree is reminiscent of [XML]'s document object model (DOM) with its hierarchy of object nodes, it is intended as a higher level entity to be programmed and read directly by humans.

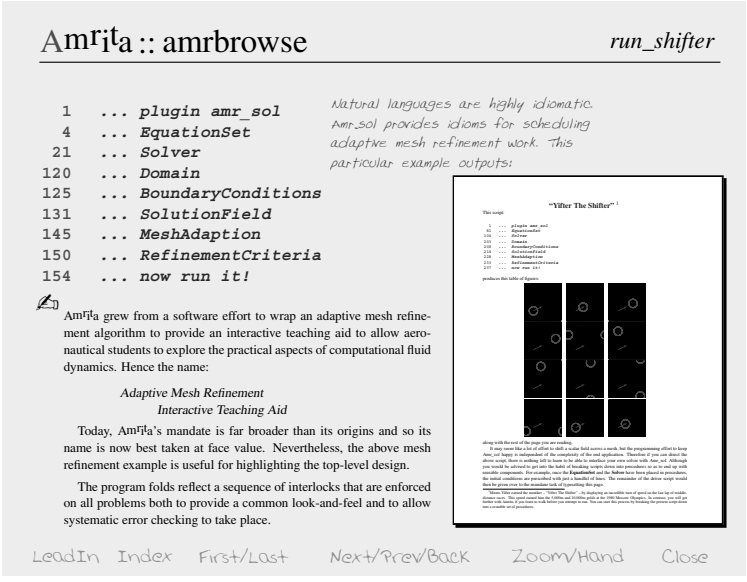


Fig. 9. The root of a fold-tree that runs through the software design of the mesh refinement engine used to compute Figure 1.

5 DOE ACTS Collection

The Department of Energy, Advanced CompuTational Software Collection (DOE ACTS) is a set of tools for constructing high-performance scientific applications on parallel computers[DM03]. The collection was set up as part of an effort to raise awareness in academia and industry of a number of software packages that have been developed at DOE laboratories, over the years, by various research groups.

Like the AIAA policy on numerical accuracy, the ACTS initiative recognizes the fundamental importance of documentation[DM03]:

> *A very important element in promoting the use of a tool is the documentation that describes the tool functionality, and its interfaces. In addition, the practice of providing proper documentation is the only vehicle for instructing users on the correct utilization of a tool, and its limitations. Adequate levels of documentation is [are] also effective in reducing the amount of time a user will spend prototyping her/his codes using the tool.*
>
> *Currently, the tools in the ACTS Collection have various different levels of documentation. In the ACTS Collection we are working to uniformly provide appropriate levels of documentation for all tools, as well as develop [a] didactical mechanism to teach users through examples at different levels of complexity and tool expertise.*

Again, however, documentation is viewed in the traditional sense that work is done in one corner, but written up in another. Consequently the ACTS initiative can be used as a vehicle with which to present issues that work to undermine computational science, as a whole.

For instance, to understand the practical problems computational science faces, you need only visit the ACTS repository and see how many packages you can install within an hour. As with any web-site there are the usual collection of broken links. For example, on the day this author visited (12th Nov. 2003) *PAWS*, *SILOON*, *TAU*, and *PETE* were all missing. Thus 4 of 18 packages were unavailable. Of the remaining packages, each has its own look and feel, and so the effort to install them is not inconsequential. Indeed, the installation notes for *PETSc*, one of the more polished packages in the collection, contains the software truism:

> *Installing PETSc can be the hardest part of using the software, since each machine's configuration is slightly different.*

Another software truism is that software developers are configured differently to software users. The developers of some of the packages at ACTS have spent close to twenty years honing their respective products, but many potential users are likely to ditch a package after an hour, unless they see tangible benefits. Consequently it is difficult to see how computational science can truly flourish until clear societal standards are established regarding the obligations that software developers and software users have towards one another. This raises the spectre of social engineering.

5.1 Social Engineering

It is clearly unrealistic for a user to expect to be driving an AMR package intelligently within an hour, of a standing start. But it is equally unrealistic for an AMR developer

to expect users to serve a ten year apprenticeship, before they can drive the software independently. Where is the middle ground?

The ACTS initiative recognizes the basic tensions between software users and software developers, but in the spirit of this paper it is worth stepping back and examining a more fundamental issue. Specifically, researchers are judged by the technical papers they publish and not by the care and attention they put in to developing and maintaining software. The reward system of publish-or-perish makes sense for algorithms, like Riemann solvers, where the intellectual content lies with mathematical details that can be typeset and followed by a broad audience. But for software bound algorithms, like AMR, traditional journal articles cover only a small part of the work performed.

Document engineering, of the kind discussed in the previous section, has the potential to improve matters, for it would allow practical examples to be packaged within a traditional-style journal paper, without increasing the printed page length, and without introducing external links. Thus readers would have access to all the work performed, at any future date, and not just selective highlights as at present. This would force authors to address the sorts of details which today are often neglected because they are not open to scrutiny. Then just as literary classics inspire their readers, computational classics would emerge to set standards for activities not covered by theory.

Of course, there is no point archiving code if it cannot be understood, or if it cannot be run because a dependent resource has gone missing. But the idea of literate programming is to allow an author to lead a reader through a code in a way that maximizes the reader's understanding of the code's workings. And the idea of idiomatic programming is to ensure that the archived code is long lived. Solution techniques come and go, but the prescription of a physical problem is time invariant and so it is separated out as an idiomatic wish list for an obliging Babel Fish to translate for a target computational engine.

If the approach sounds fanciful, it should be noted that the Association for Information and Image Management (AIIM) is actively developing a standard that defines the use of *PDF* for archiving and preserving documents[AIIM]. Their aim is to ensure that a *PDF* authored today can be read in 100 years time. Also, Babel Fish translation services of human languages are now commonplace, for example, see [BFT]. Consequently, if the community-will exists, such broad-base technologies could be deployed with the aim of improving communication within computational science.

The hope, by making work easily repeatable, would be that it would lead to a raising of standards by peer review. The idea is no different from the current peer review system, except it would now be applied to the practical elements of computational science. A brief examination of two packages from the ACTS collection shows the need to archive practical knowledge.

5.2 *POOMA*

POOMA (Parallel Object-Oriented Methods and Applications) is a collection of templated *C++* classes for writing parallel solvers using finite-difference and particle

methods. It has been retired from the ACTS Collection as the original developers have disbanded and so the product is no longer considered viable. Here it is instructive to review *POOMA* in the spirit of [Col97]. The package is a placeholder for community-wide failings and no personal criticism of *POOMA*'s developers is intended. The aim is strictly to increase awareness of age-old mistakes.

The first issue to tackle is software arrogance. *POOMA* offered no migration path for pre-existing application codes to exploit the package and so the chances of it developing a sustainable userbase was small. A software package that preaches an-out-with-the-old-and-in-with-the-new approach will itself be usurped further down the track. And as the ACTS initiative notes *users demand long term support of a tool*[DM03]. Therefore it is imperative that new software technologies are promoted, and adopted, in ways that are sympathetic to existing code activities. The design trick is to allow forward evolution, while maintaining backwards compatibility.

Moving to the issue of software pride, *POOMA* made extensive use of *C++* templates. But as Stroustrup, the architect of *C++*, himself noted[Str91]:

> *... as programs get larger, the problems associated with their development and maintenance shift from being language problems to more global problems of tools and management.*

Therefore many of the claims *POOMA* made regarding templates are moot. Specifically, replace *programs* by *simulations* and *languages* by *algorithmic* and you will immediately see why there is a need for an ACTS-like initiative:

> *As simulations get larger, the problems associated with their development and maintenance shift from being algorithmic problems to more global problems of tools and management.*

The challenge for computational science is to realise that the required *tools and management* involve non-classical activities, like document engineering, which are currently shunned to the detriment of technical progress.

POOMA's software ignorance needs little introduction to AMR experts. AMR was viewed as a bolt-on to data parallelism, rather than the skeleton upon which a simulation is built. A very similar ignorance is exhibited in CFD. A hydro-code is added to AMR, AMR is not added to a hydro-code[Qui98]. By the same token, AMR should be viewed as but a cog in a larger activity. It is not a scientific end in itself. Hence the low profile given to AMR here.

In the spirit of even handedness, this paper can also be critiqued along the lines used for *POOMA*. Its software arrogance lies with the claim that accepted procedures for reporting computational science are poor. Its software pride lies with the promotion of program folds. Its software ignorance permeates the text, for this author is just as fallible as the next person.

5.3 *PETSc*

PETSc (Portable, Extensible Toolkit for Scientific Computation) is a suite of data structures and routines that provide building blocks for writing application codes on

parallel computers. Its documentation is much more complete, and more honest, than many scientific packages. For example[PETSC]:

> *PETSc is a sophisticated set of software tools; as such, for some users it initially has a much steeper learning curve than a simple subroutine library. In particular, for individuals without some computer science background or experience programming in C or C++, it may require a significant amount of time to take full advantage of the features that enable efficient software use. However, the power of the PETSc design and the algorithms it incorporates may make the efficient implementation of many application codes simpler than "rolling them" yourself.*

highlights a key difficulty of developing sustainable scientific software.

Those who could benefit most from a package, like *PETSc*, have the least programming background to call upon. Often so little, they cannot even begin to help themselves. While those with the requisite programming background, who could contribute the most, tend to prefer to roll their own software. The problem is compounded by code changes:

> *PETSc is still undergoing development, we reserve the right to make minor changes to function names and calling sequences. Although this can make keeping your code up-to-date annoying, it will pay off in the long run for all PETSc users with a cleaner, better designed, and easier-to-use interface.*

Programmers like to write their own code for the very reason that they can insulate themselves against unwanted changes. Thus, seeing the above, some individuals will outright reject *PETSc* and decide to build their own toolkit. At the other end of the spectrum, some reluctant programmers, who do adopt *PETSc*, will deem it too much trouble to upgrade when new versions are released. Either way, the stirling effort of *PETSc*'s developers is undermined.

A singular failing of computational science, from a software perspective, is that there is no minimum programming standard needed to become a computational scientist. Therefore developers are left guessing as to the software skills of prospective users. For instance, *PETSc* offers some advice on code management[PETSC]:

> *In this file we list some of the techniques that may be used to increase one's efficiency when developing PETSc application codes. We have learned to use these techniques ourselves ...*

that a professional programmer would think distinctly quaint:

> *Work in two windows next to each other. Keep the editor running all the time, rather than continually exiting and entering it. Organize directories so code is not scattered all over the place.*

but which is already beyond that exhibited by many computational scientists. A mechanism is sorely needed to package software technique.

5.4 Packaging Software

Traditionally, computational science has concentrated exclusively on developing algorithmic capability, ignoring the sorts of issues raised by the *DOE Acts* initiative.

The world, however, is changing and the subtleties of building sustainable software activities are now more widely appreciated. Consequently, over the next few years, the packaging of scientific software is likely to receive much more attention than in the past. If you have not considered this issue before, the document-engineering approach taken in this paper will appear out of place in an AMR proceedings. But AMR, by virtue of its software complexity over traditional numerical algorithms, has a pressing need to develop methods that allow training examples to be packaged in an accessible form.

The system used here, [AMRITA], was spawned by working with AMR, but the more pertinent part of the acronym, ITA, stands for *interactive teaching aid.* Here interactive is not used in the sense of point-and-click but in the sense of reasoned experiment. An observation leads to a hypothesis whose validity is tested for, and in the process more observations are made, leading to additional hypotheses, and so on. A key observation that led to the system's development is that the accepted, linear-listing approach to programming results in code that is so hard to read it is generally ignored. Thus the feeling is that computational science could be heading for an intellectual disaster, for it is building an ever larger, and larger, edifice on software foundations that are not inspected with the same attention as given to computational theory.

With this fear in mind, [AMRITA] has evolved into a general purpose programming system geared towards making source code observable, and it has little to do with AMR per se. Figure 10, for instance, shows an example distributed with *PETSc*:

$PETSC_DIR/src/snes/examples/tutorials/ex18.c

that has been refactored to use program folds. The original author of *ex18.c*, David Keyes, might not immediately recognize his code, but it remains intact and could be recovered by formal manipulations of the fold-tree.

The example is now a fold-based literate program. The *PETSc* developers already employ literate-programming techniques to extract procedure call information for their documentation. Therefore, seeing Figure 10, do they spend the time to dissect the principles behind program folds, or do they roll their own? Interestingly, the whole idea of program folds is to eliminate a not-invented-here mentality. So long as any new approach to program folds are re-entrant, the old approach can leverage off it, seamlessly. The same is true in reverse, thus removing traditional ownership fears. Through their co-operative design, program folds can add value to any programming activity.

There is insufficient space here to remove all suspicions you might have. But as shown by Figure 11, a documentation autopilot can be added to a fold-tree to lead readers through a code's construction and multiple documentation threads can be added to cater for programmers of different persuasions. Thus if your concerns were known, they could be addressed directly.

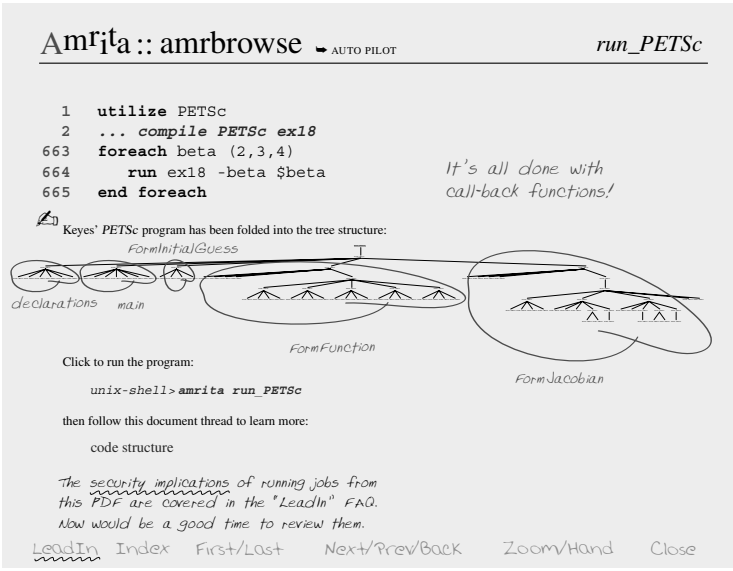


Fig. 10. *PETSc* code can be refactored using program folds, as can *any* program, in *any* language. Folds offer a straightforward migration path from the traditional linear view of code to a tree-based approach. Observe that Keyes' program can be run directly from the listing, as it can from the *PDF* version of this document.

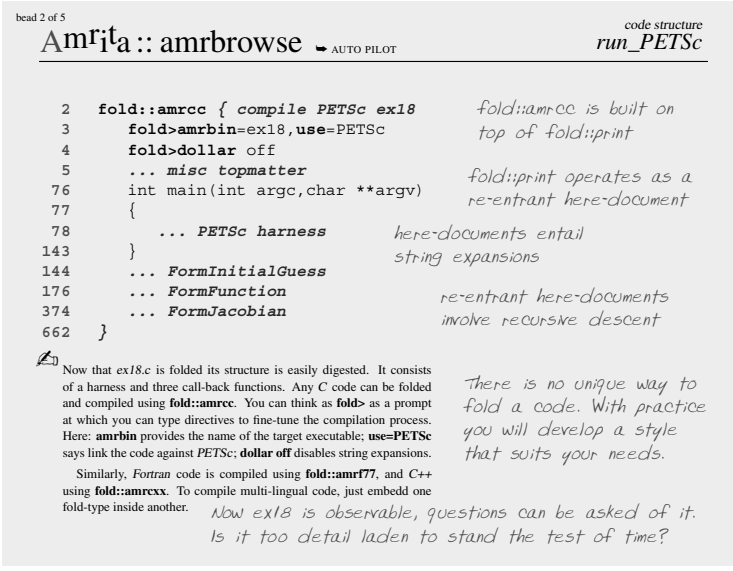


Fig. 11. A fold-tree can contain a number of document threads. Thus an author can cater to readers from different backgrounds. Now *ex18* is observable, a case can be made for adding an idiomatic layer to *PETSc* that would act as buffer between the system's developers and the system's end-users.

6 Something More Is Needed . . .

Clearly the arguments put forward here are not watertight in the sense of a mathematical proof. Therefore, if you feel inclined, there is plenty of ammunition you can gather so as to shoot this bearer of bad tidings. But you are not asked to agree with every last line of thought. All that matters is that you are willing to entertain the idea that the reporting of computational science might fall short of that which technology now allows. You should also be willing to question the current out-of-sight-out-of-mind approach to programming: is it inherently dangerous?

It is easy to dismiss [Col97] — the inspiration for the present work — on the grounds that it is only a newspaper article. But the journalist in question, Tony Collins, is no ordinary hack. He is executive editor of *Computer Weekly*[CW], a British trade publication aimed specifically at information technology professionals. He was writing in *The Telegraph* to promote his book *Crash: Ten Easy Ways to Avoid a Computer Disaster*[CB97]. This 400 page tome has subsequently received sufficient plaudits that it should be made required reading for all computational scientists.

For his undergraduate degree, this author was required to read the engineering classic *Structures: Or Why Things Don't Fall Down*[Gor78] which contains many examples of why, contrary to the title, structures fail with regrettable frequency. Thus *Crash* can be thought of as continuing a long tradition of engineering introspection aimed at eliminating disasters. The recommendation here is that computational science needs to develop a similar culture of introspection to safeguard against intellectual disasters.

Whether computational science can rise to the challenge is anyone's guess. But in 2016 it would be nice to think that a workshop proceeding such as this one would take the form of an electronic document, that retains the rigour of this typeset book, but which would allow readers to sample the reported work firsthand, without having to wade through the humdrum software details which today are viewed as unavoidable. The hope is that the time saved programming can be employed on more lasting scientific activities. Instead of scientific discovery through advanced computing, there will be scientific discovery through accessible computing. This simple shift in emphasis might not appear to be that big a deal, but it will result in a movement away from general programming languages, like *Fortran 90* and *C++*, to domain specific languages, like [POVRAY].

Although *Povray* is a hobbyist activity it contains many elements that put computational science's software efforts to shame. For example, *Povray*'s users are not passive operators of a code. The international ray tracing competition[IRTC] has harnessed their competitive spirit to produce wondrous photo-realistic images that act to inspire that community's programming skills. Computational science desperately needs something similar. If nothing else, it will help erode the mentality where many scientists, today, wear their programming ineptitudes as a badge of honour.

Another weakness is that many individuals entered the field because of a scientific interest, such as aerodynamics, but their time is now almost exclusively spent on programming activities for which they have no formal training. It is imperative

that the computational community addresses its software training needs. The current anything goes approach is unacceptable, but the community should look before it leaps. For example, a one-size-fits-all approach will fail, as will lurching from one programming fad to another. A commonsense sense approach is called for. One designed to make simulations more repeatable, more accessible, and more accountable.

For his undergraduate degree, this author had to demonstrate proficiency both in technical drawing and in the use of workshop tools and machinery, as laid down by the Institution of Mechanical Engineers[IMECHE]. A software equivalent, for computational science, would give code developers a prototype person to design for. At present a developer has to guess the community's stomach for programming. This hit-and-miss approach is not working.

The code which produced Figure 1 was constructed in a small corner of an aircraft hangar. Another corner of the hangar housed an applied psychology unit that was charged with conducting aircraft evacuation tests. It quickly became evident that when passengers co-operate with one another a plane can be evacuated much more quickly and safely than when individuals stampede towards the available exits. Since research is an innately selfish activity, it is perhaps not surprising that the development model for scientific software is more of a stampede than an orderly procession. Consequently funding agencies must be made to realise that solicitations for scientific software development must address the need for community co-operation. The roles of software developers and software users needs to be clearly defined.

Human psychology being what it is, this author is conscious that the recommendations made here may be off beam. For an individual's capacity to critique his, or her, own work is usually vastly less than the capacity to rubbish the efforts of others. Therefore, in the spirit of egoless programming, it must be made clear that the present manifesto is not a clumsy attempt to promote[AMRITA]. In 1996, Neil Armstrong — the first man on the moon — stood fifty yards from where Figure 1 was constructed to give a commencement speech entitled *Something more was needed.* The something more to which Armstrong refers was engineering, without which the development of man-powered flight would have stalled. In a hundred years time, a scientific historian may well identify document engineering as the key that allowed computational science to reach its moonshot. But feel free to substitute your own dreams for how the discipline can move forward. *Something more is needed.*

Acknowledgements

I am grateful to Tomek Plewa for giving me this opportunity to climb onto my computational soap-box. I am also grateful to colleagues, past and present, for helping to shape the views expounded here.

References

AIAA. Editorial policy statement on numerical accuracy and experimental uncertainty. AIAA Journal, **36**, 7 (1998)

AIIM. PDF-Archive Committee http://www.aiim.org/pdf_a

AMRITA. Adaptive Mesh Refinement Interactive Teading Aid http://www.amrita-ebook.org/drink-me

BC89. Berger, M.J., Colella, P: Local adaptive mesh refinement for shock hydrodynamics. J. Comp. Phys., **82**, 67–84 (1989)

BFT. Babel Fish translation http://babel.altavista.com

BK89. Bowler, K.C., Kenway, R.D., Pawley, G.S., Roweth, D., Wilson, G.V.: An introduction to OCCAM2 programming (2nd ed.). Chartwell-Brandt, Bromley, Kent (1989)

CB97. Collins, T., Bicknell, D.: Crash: ten easy ways to avoid a computer disaster. Simon & Schuster, London (1997)

Col97. Collins, T.: Same old silence, same old mistakes. The Daily Telegraph, London Manchester (July 15th 1997)

CW. Computer Weekly. http://www.computerweekly.com

CWEB. Knuth, D.K.: The CWEB system of structure documentation. Addison-Wesley (1994)

DE. The ACM Symposium on Document Engineering http://www.documentengineering.org

DM03. Drummond, T., Marques, O.: The ACTS Collection, robust and high-performance tools for scientific computing, guidelines for tool inclusion and retirement. http://acts.nersc.gov

God59. Godunov, S.K.: A difference scheme for numerical computation of discontinuous solutions of hydrodynamic equations. Math. Sbornik., **47**, 271–306 (1959) English translation in U.S. Joint Publications Research Service, JPRS 7226 (1969)

Gor78. Gordon, J.E.: Structures, or why things don't fall down. Penguin, London (1978)

IMECHE. Institution of Mechanical Engineers http://www.imeche.org.uk

IRTC. Intl. Ray-Tracing Competition http://www.irtc.org

Knu92. Knuth, D.K.: Literate Programming (1984). CLSI (Center for the Study of Language and Information) Lecture Notes **27**, Stanford University (1992)

Lan98. Laney, C.B.: Computational Gasdynamics. Cambridge University Press (1998)

Mon92. Monaghan, J.J.: Smooth Particle Hydrodynamics. Ann. Rev. Astron. Astro. **30**, 543–574 (1992)

PDF. PDF Reference (4 ed.) Adobe Portable Document Format version 1.5. Adobe Systems Incorporated, San Jose, California (2003)

PETSC. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Portable, Extensible Toolkit for Scientific Computation. Version 2.1.5, see files *docs/installation/index.htm* and *docs/manual.pdf* http://www.mcs.anl.gov/petsc

PH52. Pankhurst, R.C., Holder, D.W.: Wind tunnel technique. Pitman, London (1952)

POVRAY. Persistence of Vision Ray-Tracer http://www.povray.org

PRQ92. Powell, K.G., Roe, P.L., Quirk, J.J.: Adaptive-mesh algorithms for computational fluid dynamics. In: Hussaini, M.Y., Kumar, A., Salas, M.D. (eds) Algorithmic trends in computational fluid dynamics. Springer, Berlin Heidelberg New York (1992)

Qui91. Quirk, J.J.: An adaptive grid algorithm for computational shock hydrodynamics. Ph.D. Thesis, Cranfield Institute of Technology, Cranfield (1991).

Qui94. Quirk, J.J.: A contribution to the great Riemann solver debate. Int. J. Numer. Methods Fluids, **18**, 555–574 (1994)

Qui98. Quirk, J.J.: AMRITA - A computational facility (for CFD modelling). In: Deconinck, H. (ed) 29th Computational Fluid Dynamics. von Karman Institute, Brussels (1998)

RAE54. Royal Aircraft Establishment report on the Comet investigation. Her Majesty's Stationery Office (1954) `http://www.hmso.gov.uk`

Roa98. Roache, P.J.: Verification and validation in computational science and engineering. Hermosa Publishing, Albuquerque (1998)

Rob90. Roberts, T.W.: The behaviour of flux difference splitting schemes near slowly moving shock waves. J. Comp. Phys., **90**, 141–160 (1990)

Roe81. Roe, P.L.: Approximate Riemann solvers, parameter vectors, and difference schemes. J. Comp. Phys., **43**, 357–372 (1981)

RPC86. Report of the Presidential Commission on the Space Shuttle Challenger Accident (1986) `http://www.hq.nasa.gov/office/pao/History/sts51l.html`

Str91. Stroustrup, B.: The C++ programming language (2nd ed.). Addison-Wesley (1991)

WC84. Woodward, P.R., Colella, P: The numerical simulation of two-dimensional fluid flow with strong shocks. J. Comp. Phys., **54**, 115–173 (1984)

STV95. ICASE/LaRC workshop on adaptive grid methods (Nov. 7-9 1994). Eds. South, J.C, Thomas, J.L, Van Rosendale, J. NASA conference publication 3316 (1995)

TEX. Knuth, D.K.: TEX The Program. Addison-Wesley (1986)

Van79. Van Leer, B.: Towards the ultimate conservative difference scheme V. A second-order sequel to Godunov's method. J. Comp. Phys., **32**, 101–136 (1979)

XML. Extensible Markup Language (XML) `http://www.w3.org/XML`

Massively Parallel Simulations with DOE's ASCI Supercomputers: An Overview of the Los Alamos Crestone Project

Robert P. Weaver[1] and Michael L. Gittings[1,2]

[1] Applied Physics (X) Division, Los Alamos National Laboratory, Los Alamos, NM USA 87545

[2] Science Applications International Corporation, La Jolla, California 92121
rpw@lanl.gov and *gittings@lanl.gov*

Summary. The Los Alamos Crestone Project is part of the Department of Energy's (DOE) Accelerated Strategic Computing Initiative, or ASCI Program. The main goal of this software development project is to investigate the use of continuous adaptive mesh refinement (CAMR) techniques for application to problems of interest to the Laboratory. There are many code development efforts in the Crestone Project, both unclassified and classified codes. In this overview I will discuss the unclassified SAGE and the RAGE codes. The SAGE (SAIC adaptive grid Eulerian) code is a one-, two-, and three-dimensional, multimaterial, Eulerian, massively parallel hydrodynamics code for use in solving a variety of high-deformation flow problems. The RAGE CAMR code is built from the SAGE code by adding various radiation packages, improved setup utilities, and graphics packages. It is used for problems in which radiation transport of energy is important. The goal of these massively-parallel versions of the SAGE and RAGE codes is to run extremely large problems in a reasonable amount of calendar time. Our target is scalable performance to $\sim$10,000 processors on a 1 billion CAMR computational cell problem that requires hundreds of variables per cell, multiple physics packages (e.g., radiation and hydrodynamics), and implicit matrix solves for each cycle. A general description of the RAGE code has been published in [1], [2], [3] and [4].

Currently, the largest simulations we do are three-dimensional, using around 500 million computation cells and running for literally months of calendar time using $\sim$2000 processors. Current ASCI platforms range from several 3-teraOPS supercomputers to one 12-teraOPS machine at Lawrence Livermore National Laboratory, the White machine, and one 20-teraOPS machine installed at Los Alamos, the Q machine. Each machine is a system comprised of many component parts that must perform in unity for the successful run of these simulations. Key features of any massively parallel system include the processors, the disks, the interconnection between processors, the operating system, libraries for message passing and parallel I/O, and other fundamental units of the system.

We will give an overview of the current status of the Crestone Project codes SAGE and RAGE. These codes are intended for general applications without tuning of algorithms or parameters. We have run a wide variety of physical applications from millimeter-scale laboratory laser experiments, to the multikilometer-scale asteroid impacts into the Pacific Ocean, to parsec-scale galaxy formation. Examples of these simulations will be shown. The goal of our effort is to avoid ad hoc models and attempt to rely on first-principles physics. In addition

to the large effort on developing parallel code physics packages, a substantial effort in the project is devoted to improving the computer science and software quality engineering (SQE) of the Project codes as well as a sizable effort on the verification and validation (V&V) of the resulting codes. Examples of these efforts for our project will be discussed.

1 Overview of the Department of Energy (DOE) ASCI Program

The ASCI program at the DOE was started in 1996 as part of the Department's Stockpile Stewardship Program (SSP). One of the main activities of the DOE National Laboratories is to maintain the safety and reliability of the United States nuclear stockpile in the absence of nuclear testing. For decades, the testing of our nuclear weapons allowed weapons scientists to demonstrably certify to the national leadership that nuclear weapons would perform as expected should the President require their use. Additionally, the weapons in the current active stockpile will be retained well beyond their expected lifetimes. Through surveillance of these stockpiled weapons, we have found further complications caused by the aging of various materials in these systems. Without testing the effects of these changes, the scientists at the National Laboratories are given the difficult task of continuing to certify the safety and reliability of these systems. A variety of programs and initiatives were started to address this task. No individual program under SSP constitutes a "replacement" for experimental validation of a particular design (i.e., nuclear testing). Rather, the collection of programs (e.g., Above Ground Experiments [AGEX] such as DAHRT and NIF and others; the Advanced Simulation and Computing Initiative [ASCI] for improved computational modeling; Enhanced Surveillance; etc.) constitutes the SSP program and allows the National Laboratories to make the most informed decisions about the safety and reliability of the nuclear stockpile in the absence of testing.

One of the components of the SSP is the enhancement of the complex computing codes used to simulate the operation of a nuclear weapon. Complex codes have been used for decades in the design and analysis of nuclear weapons tests. Now, without testing to experimentally verify the proper behavior of these complex designs, it was decided to accelerate the development of the codes used to simulate weapons. Furthermore, as the systems in the nuclear stockpile age, the resulting material changes invariably result in significant three-dimensional issues that require verified and validated three-dimensional (3D) codes for evaluation and assessment. The addition of the third physical dimension to the simulation codes results in a huge increase in the computer power and speed required in order to answer such 3D questions in reasonable amounts of time. These requirements led to the ASCI Program at the DOE. The fundamental ideas involved in the ASCI program were to accelerate the development of high-end supercomputers and the associated (necessarily parallel) software in order to have approximately 100 TeraOperations per sec (TOps) capability by the year 2004. The ASCI program is a concerted collaboration among the National Laboratories, computing hardware vendors, academic institutions and many contractors to provide the capability to simulate a model that has a billion computational cells and has an extremely complex geometry with multiple-nonlinear physics. Additionally,

this capability must be verified calculationally and validated against a wide variety of experimental results, both Above-Ground Experiments (AGEX) and the previous nuclear tests that have been done. This effort is a daunting task. Some of what is presented in this paper (the unclassified parts of the Crestone Project) documents our progress towards these goals.

1.1 An Evolution of ASCI Supercomputers

The ASCI supercomputers began arriving at the DOE labs (Sandia, Livermore, and Los Alamos) in ∼1997, with one 3 TOps machine each at the three DOE labs. One of the fundamental characteristics of these machines was their massively-parallel structure. Each machine could be used as a single supercomputer with ∼6000 processors available for use. At Sandia there was the ASCI Red machine, a MPP design with Intel chips. At Livermore and Los Alamos there were ASCI Blue machines (SMP architecture) with SGI building the Los Alamos Bluemountain supercomputer (6144 cpus) and IBM building the Livermore Blue Pacific machine. The first ASCI Level-

Fig. 1. The Bluemountain 3TOps Los Alamos supercomputer.

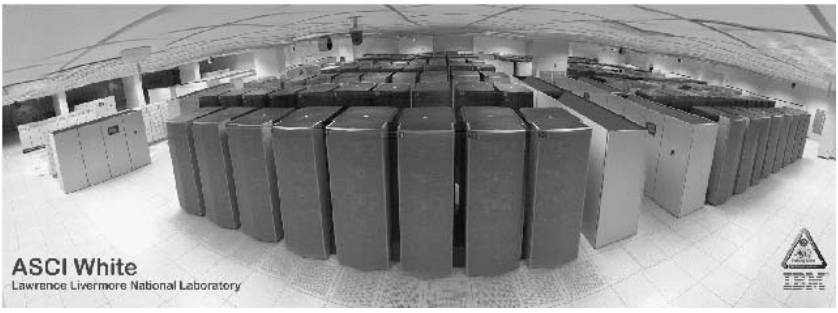


Fig. 2. The White 12TOps ASCI supercomputer at Livermore.

Fig. 3. The Q 20TOps ASCI supercomputer at Los Alamos.

1 milestones were accomplished on these machines, each Laboratory fundamentally using its own resources to complete the milestones. We will present more on these results later. The next major step in the ASCI hardware was the delivery in 2001 of a 12 TOps ASCI White machine at Livermore, built by IBM on the SP3 SMP architecture. Again many thousands of processors were available for use on single simulations, but this time the chip speed was greater, the interconnect was faster, and the disk storage and I/O rates were greatly enhanced over the previous generation machines. The Global Parallel File System (GPFS) on the White machine holds 77 Terabytes (TB) of data! From the 3 TOps machines to the ASCI White 12 TOps machine, we actually saw a 3-4 fold increase in speed, or more to the point, a 4-fold decrease in the time-to-completion for a major 3D simulation. These machines were actually performing simulations that would have taken literally hundreds of years on any prior supercomputer, all due to the enhanced performance from parallel computations. The ASCI hardware acceleration continued in 2002 with the delivery of the ASCI Q machine at Los Alamos, a 20+ TOps capability supercomputer built by HP/Compact using alpha chips. Using about the same number of processors, this new machine could complete the same run $\sim$10 times or 3 times faster than either the Bluemountain or the White machine, respectively. So, not only were we running the same simulations faster, but we were also including more complex physics and running even more complex geometries than before. A complex 2D simulation that actually took about one month to finish on the Bluemountain machine would run in about a week on the White machine and about two days on the Q machine. (So naturally we actually ran more complex, detailed runs on the Q machine that finished in a few weeks time!) The next generation of ASCI supercomputers is still being developed, but a contract to IBM for a 100 TOps supercomputer at Livermore is scheduled for delivery in $\sim$2006. I have refrained from discussing the details of the costs for these supercomputers. If interested, the reader can contact the individual Laboratories for information.

1.2 An Evolution of ASCI Software – "Think Parallel"

With the advent of massively-parallel hardware available for general production use at the National Laboratories, there was a concerted effort to transition working codes and develop new codes that could take advantage of these resources. For the Los Alamos Crestone Project, this transition was a natural occurrence. The source code for the SAGE and RAGE codes that were delivered to Los Alamos from SAIC as part of the collaborative contract were initially written for the supercomputers of the early 1990's: a single vector processor architecture such as the CRAY machines. Since the capability to setup and run 3D simulations was available from the start of the contract ($\sim$1994), we soon realized that the supercomputers of the time (CRAY's) could not be effectively utilized for large 3D work. These machines were just not fast enough to complete 3D simulations (even rudimentary ones) in any reasonable amount of calendar time. From the 2D runs that were done during this period, we estimated that 3D runs would take more than 60 years to complete on CRAY architecture. Thus, with the advent of the ASCI program in 1996, we were already starting a conversion process from the original CRAY vector Fortran 77 code to a new paradigm: parallel and modular code with effective SQE. There were several initial starts along this path, but the one chosen was to employ Fortran 95 with C program interfaces for I/O and the explicit use of message passing to effect communication among the processors. Domain decomposition was used to spread the computational cells of the problem among the processors. More details of this plan are given below and in some of the references ([3],[5],[6]). We needed to "Think Parallel" in our software design to begin running 3D simulations in a reasonable amount of time. For the "60 year" run to complete in just a month we needed an effective increase in computing power of 720 or almost 3 orders of magnitude! The "Think Parallel" phase is a reminder that simple conversions of algorithms developed for efficient use of CRAY vector single-processor architectures is much more that simply duplicating the computational mesh on each processor or even rearranging loops of the physics algorithms. In order to "Think Parallel" one must must completely change the framework and thought process (and hence the entire coding) for developing algorithms. An excellent, albeit brief, description of this thought process is given in a paper we wrote for the 1999 ISSW22 meeting in London [3].

2 An Overview of the Los Alamos Crestone Project

The Los Alamos Crestone Project developed from an extension of the highly successful collaboration between the scientists in the Thermonuclear Applications groups (X-2) of Los Alamos and the scientists at a major defense contractor, Science Applications International Corporation (SAIC). This high-performance collaboration team with members from both Los Alamos and SAIC continues today as one of the premier code-development successes at Los Alamos.

A little history helps put some of the important aspects of this collaboration in perspective. During the early 1990's one of us (Weaver) took an unusual Change-of-

Station assignment (COS) for Los Alamos. Instead of the usual assignment of reporting to the DOE or DOD headquarters in Washington, DC, Weaver decided to perform a more technical COS, and thus he moved to San Diego, California, working on an enhanced collaboration for nuclear weapons effects that had been active for more than five years at that point. During this COS Weaver observed a code called SAGE, being developed by one of us (Michael Gittings). This code was originally developed to study the detailed effects of an underwater shock wave from a nuclear explosion at distances of many kilometers from the core of the detonation. Accomplishing the required resolution of centimeters to meters at distances of many kilometers from the source required a new approach to grid generation and grid development. The solution that Gittings was developing involved acontinuous adaptive mesh refinement technique that was based upon a modern Eulerian hydrodynamics framework. In this method, one could refine any grid cell in the problem at any time in order to more accurately simulate the physics at that location. Conversely, if the physical gradients in a particular cell of the problem became small, then the method allowed the cells to become more coarsely resolved at that location. This dynamic controlling of the total number of cells in the problem results in minimizing the total memory and computer time required to complete that cycle of the problem, while maximizing the accuracy of the solution by having high-resolution in only those areas of the geometry that required them. We now call this technique Continuous Adaptive Mesh Refinement, or CAMR. Adaptive mesh refinement codes were not universally accepted in the early 1990's, but today the use of adaptive meshing seems to permeate a wide variety of computational physics from astrophysics to radiation transport to hydrodynamics, just to name a few. During this COS assignment in San Diego, Weaver decided to use Gittings' SAGE code to run some unclassified samples of problems of interest to the Laboratory. This simple exercise convinced us that creating a collaboration between SAIC and the Laboratory would result in a tremendous enhancement to the X-Division existing code simulation capability. After the end of the COS assignment in 1993, X-2 initiated a contract with Gittings of SAIC that has continued through the present time. The results of this collaboration between a world-class code developer (Michael Gittings) and the direction of an active code user has resulted in a truly amazing simulation capability for Los Alamos National Laboratory. This collaboration led directly to the creation of the ASCI sponsored Crestone Project at Los Alamos, in which the main goal of the project is to develop, verify, and validate CAMR Eulerian hydrodynamics based capabilities for use in projects of interest to the laboratory.

Since the Crestone Project is funded by the DOE ASCI Program, there are many actual code development efforts managed by the Project. Some of the efforts, such as the SAGE, RAGE, NOBEL, SAGA, and IAGO codes are unclassified codes that can be discussed in open literature such as this publication. Other efforts under the Project are classified, and only some general characteristics of the truly amazing simulations that have been performed over the years can be described here. We believe it is an understatement to say that the Crestone project has far exceeded any expectations of the ASCI program. This fact is demonstrated by the project's continuous completion of many major Level-1 Milestones for the DOE

ASCI program year-after-year (see below). More to the point, the examples of the use of the Crestone Project codes by many end-users has fundamentally changed the way the Laboratory scientists think about computing: massively-parallel simulations are answering questions though simulations that we never even dreamed of asking even as early as a few years ago. Here I will present several examples of these efforts. Other papers at this conference also highlight aspects of this statement; see, for example, the work by Galen Gisler using SAGE and RAGE on complex 3D asteroid impact simulations. Although Verification and Validation (V&V) is a major activity of the project, the funding lines for V&V work are distinct from the basic code development work. Some examples of V&V for the Crestone project codes will be shown here, as well as many references for other examples([7],[4],[8],[10],[11],[12],[13],[2],[18],[23],[24]).

The code continues to be used for a variety of application work, ranging from AGEX experiments, to volcanology, to fundamental physics of complex hydrodynamics ([9], [14], [15], [16], [17], [19], [20], [21], [22], [26], [27]).

The Crestone Project team currently is led by three main individuals: Mike Gittings is the chief code architect for the project; Bill Archer is the Co-Project leader for code development and Bob Weaver is the Co-Project leader for physics design and initial demonstration applications. The FY2002 budget for the Crestone project was ∼$7.2M and the FY2003 budget is ∼$8.5M. This project is a significant code development effort at Los Alamos and supports approximately 25 FTEs (full-time-equivalent staff). There are approximately two-three dozen users of the Project codes, both inside and outside the Laboratory. Although three-dozen users seems like a small number, for the kinds of specialized high-performance computing physics used in the Crestone Project codes, this number is actually larger than any other project we know.

The Crestone Project Unclassified Codes

The Fortran77 version of the SAGE code that Gittings was developing in the late 1980's and the early 1990's had many of the capability elements of the current version of SAGE: 1D, 2D or 3D geometry, CAMR Eulerian-based high-order Godonov hydrodynamics([28],[29]), and adaptive mesh capabilities optimized for hydrodynamic shock problems. Today's version of SAGE does all these things as well, but, it has been completely re-written with a massively-parallel implementation based on Fortran 95 and the message-passing interface (MPI).

The RAGE code is built from the SAGE code base by adding a variety of physics, most notably a two-temperature radiation diffusion package (see [1]). Two key individuals responsible for the development and implementation of the radiation diffusion solver in RAGE are Tom Betlach and R. Nelson Byrne, both of SAIC. They have not only written the original version of the implicit diffusion solver, but they have also maintained this part of the code in production mode since its beginning. They both interact (almost daily) with end-users in order to solve problems as they arise with this package. Additionally, Tom Betlach is aggressively pursuing modern options for implementation of multi-grid methods in this section of the code physics

(see below). The main characteristics of the RAGE code are: multi-material Eulerian CAMR in 1D, 2D, or 3D; a modern treatment of the Eulerian hydrodynamics based originally on the Piecewise-Parabolic Method (PPM) used in astrophysics since the early 1980's([28],[29]); unit aspect ratio cells (square in 2D and cube in 3D); radiation energy transport by a two-temperature (material and radiation) grey diffusion solution; preliminary interface treatment; and a material strength package. The SAGE and RAGE codes clearly have joint intellectual property rights between SAIC and Los Alamos. The CAMR adaption algorithms are the heart of the methodology and are implemented so that any two adjacent cells can differ by no more than a factor of two in mesh size. Each level of refinement is accomplished by halving the size of the mother cell in each direction, so the factor of two requirement between adjacent cells is equivalent to one level of adaption. The implicit radiation diffusion solve uses a conjugate gradient iteration with a point Jacobi (diagonal scaling) preconditioner. There is a variety of efforts underway currently to employ multi-grid techniques to the CAMR Eulerian mesh for the radiation diffusion solver. A couple of these efforts show great promise, where the payoff is measured in an actual reduction in the wall clock time to complete a full cycle of the radiation hydrodynamics. Merely reducing the iteration count is, in our opinion, not the only measure of success. We believe that reducing the overall computation time is the desired result, either by algorithm or by reducing iterations, or both..

The parallelization strategy for these codes is based on the use of Fortran 95 and the message passing interface, or MPI, paradigm. Both allow for portability and scalability. In fact, both SAGE and RAGE are used on all available supercomputer platforms and are even used in the evaluation of new machines. The codes are so portable that they even rundesktop and laptop computers, including the Apple Macintosh with MacOS X unix environment. [In fact, most of the development work is done on the Mac, and then code versions are immediately uploaded and used on 1000 processor supercomputers.] Load leveling among the many processors used in a simulation is based upon a simple reordering of the cell pointer list at the end of each computational cycle. At each cycle of the simulation a determination is made for each cell in the problem as to whether to subdivide a cell (refine the zoning) or merge adjacent cells (coarsen zoning). This decision is based upon the magnitude of the local physical gradients at that location. If the gradients become steep, then the code creates finer zoned daughter cells from coarser cell meshes; alternatively, when the gradients become small enough, then daughter cells are recombined into the original mother cells to produce a coarser mesh. The load leveling of a massively parallel run is maintained effectively by inserting the newly created daughter cells immediately after the mother cell in the global cell list, thereby maximizing the probability of on-processor memory locations. The global cell list is reorganized at each time step (M total new cells at the end of a time step on N processors results in M/N cells per processor), and the result is excellent load-leveling. Empirically we have found that good performance is obtained with cells-per-processor in the range of 10,000 – 100,000.

Many finite difference codes use computational grids that are logically rectangular and structured. In this type of code the finite difference equations are often coded as follows:

$something(i,j) = (a(i+1,j) - a(i-1,j))/dx(i).$

Parallelization of this type of code structure is difficult and typically involves the use of ghost cells at the domain boundaries. This code structure also limits the problems to the number of dimensions that are hard-wired (two dimensions in this example). The philosophy used for the CRAY vector version of the RAGE code was different. This same expression in RAGE would be:

$something(l) = (a(l\_right(l)) - a(l\_left(l))/dx(l),$

where l_right(l) and l_left(l) are the global addresses of the cells to the right and the left of cell l. This method uses more memory and is somewhat slower than the original example, but it makes the coding for unstructured grids much easier. An added benefit is that it allows the same code to do 1D, 2D, and 3D problems by looping over the dimensions. In order to parallelize this concept, our main concern was to have a code that scaled well. The resulting implementation uses message passing to obtain a local *copy* of the neighboring cell's data. The essence of this method is the creation of a local scratch array of values corresponding to the data needed from the cells to the right and left (up and down; near and far) of the current cell. The current version of RAGE uses the following structure to parallelize this example (where NUMDIM is the number of dimensions in the problem):

Do $DIR = 1, NUMDIM$
  call $get\_next(DIR, HI\_SIDE, a, a\_hi)$
  call $get\_next(DIR, LO\_SIDE, a, a\_lo)$
  $something(l) = (a\_hi(l)) - a\_lo(l))/celldimension(l, DIR)$
endDo

The subroutine *get_next* obtains a copy of the requested cell-centered data ("a") from an array which can be located anywhere (i.e., any processor). All communications between processors are hidden from the user and the resulting scaling is very good. This structure allows the code to be very portable, relying only on the machine having the basic MPI libraries. More details on these techniques are given in reference [3].

Since gather/scatter MPI routines are used to copy required data from where it resides (i.e. - which processors memory) to the local processor memory, cache reuse is actually quite good in the parallel implementation. A very nice report of models of the scalability of the SAGE code is given by Kerbyson, et.al., at Supercomputing 2001 and Supercomputing 2003 ([5],[6]). The primary goal of the parallel implementation is maintainability, scalability, and portability to new platforms on problems of 1 billion CAMR cells. Maintainability refers to the suite of software quality practices in the Crestone Project in order to produce an understandable, modular code that has a well-documented and reproducible heritage. Since ASCI hardware, and indeed the

computing industry itself, seems to be changing hardware every 1 or 2 years, the fundamental goal for the structure of our project codes is portability with scalability to 1000's of processors. Ideal scaling in this sense would be to run a problem on 10000 processors (with 10000 times the workload [i.e., cells]) in exactly the same wall-clock time it takes to run on 1 processor with 1/10000th the work. Although we have had much greater success with scalability on MPP based machines, such as the ASCI Red machine or the Cray T3E, the scaling on SMP based machines is reasonably acceptable. Fig. 4 shows a graph of the scaling results on a wide variety of supercomputers.

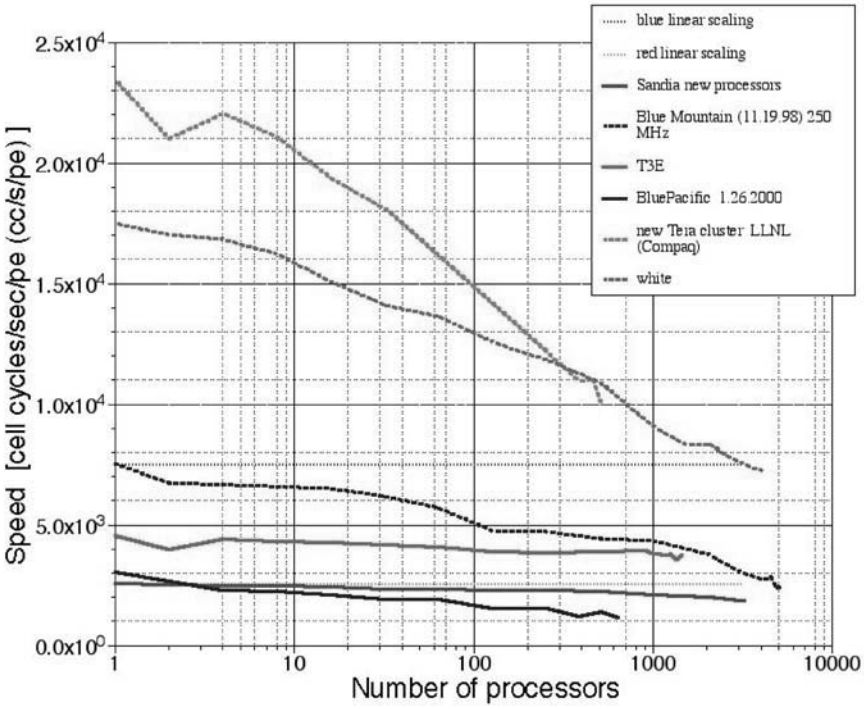


Fig. 4. The scaling performance of the SAGE code on various supercomputers.

2.1 Crestone Project Codes Represent an Improvement Over Previous Generation Techniques

Most of the computational work performed in X-Division at Los Alamos, as well as in the design divisions at Livermore, has traditionally relied on pure Lagrangian hydrodynamic methods. There are two substantial problem areas from past 2D Lagrangian code use that the Crestone Project codes solve. Speaking from the perspective of an end-user, the first major difficulty for a Lagrangian hydrodynamics-based

code is that of setup, or generating the logical and physical mesh with the constraint of having complex geometry following contour interfaces. The second major problem historically with 2D Lagrangian based codes is that of mesh tangling as the simulation proceeds. If there is even a slight aspect of complex hydrodynamics involved in the simulation, such as vorticity or sheer flows, then a major portion of time has been spent in actual hand manipulation of mesh points (manual rezoning) to continue the simulation. The tools and codes in the Crestone Project solve both of these problems in a elegant fashion. We should mention that an alternate path to a modern solution to the 2D Lagrangian problems is accomplished by the use of Arbitrary Eulerian-Lagrangian, or ALE hydrodynamic algorithms. This hydrodynamic framework is, in fact, used by the other major ASCI code-development efforts at both Los Alamos and Livermore. As with any complex multi-physics code projects, there are advantages and disadvantages of any particular formulation.There are strengths and weaknesses of both the CAMR Eulerian approach and the ALE approach. However, for current ALE schemes, complex geometry grid generation as well as mesh-motion both remain difficult and research-oriented aspects of those efforts.

2.2 Problem Setup for Complex 3D Geometries

Although there are mesh generation tools built into the SAGE code, by far the simplest approach to grid generation is by importing 3D solid model representations of the object being simulated. Mainly through the work of Rob Oakes and his coworkers in X-Division of Los Alamos([25]), we have the ability to import nearly any 3D solid model geometry into the code and build a multimaterial mesh based on this geometry. The beauty of this whole process is that once the solid model exists, then the code itself parses the geometry and automatically creates the CAMR grid conforming to the input geometry. The end-user needs only to specify the physical size of resolution required for capturing the appropriate physics in each material. The code does the rest! Examples of this setup are shown in Figs. 5, 6, and 7.

2.3 Continuous Adaptive Mesh Refinement (Cell-Based Refinement)

The second traditional problem of Lagrangian-based hydrodynamics from the past has been the need for hand rezoning of complex simulations in order to prevent time-step crashes and/or badly formed zoned (e.g., bowties or boomerangs; or worse yet negative volume cells). This activity traditionally dominates an end-users time in completing a 2D complex simulation. The continuous adaption logic built into the Crestone Project codes allows the code to do the work as the problem progresses with time. Cells in the simulation are placed (created or re-combined) in the 1D, 2D, or 3D geometry where they will provide the most accurate representation of the physics. This continuous adaption in both space and time is performed every cycle of the simulation. The overhead associated with this CAMR technique has been empirically determined to be $\sim$20% of the total runtime. Since we gain several orders of magnitude in efficiency by the use of CAMR (over uniform meshing), this overhead time is completely acceptable. The true breakthrough for the end-user is

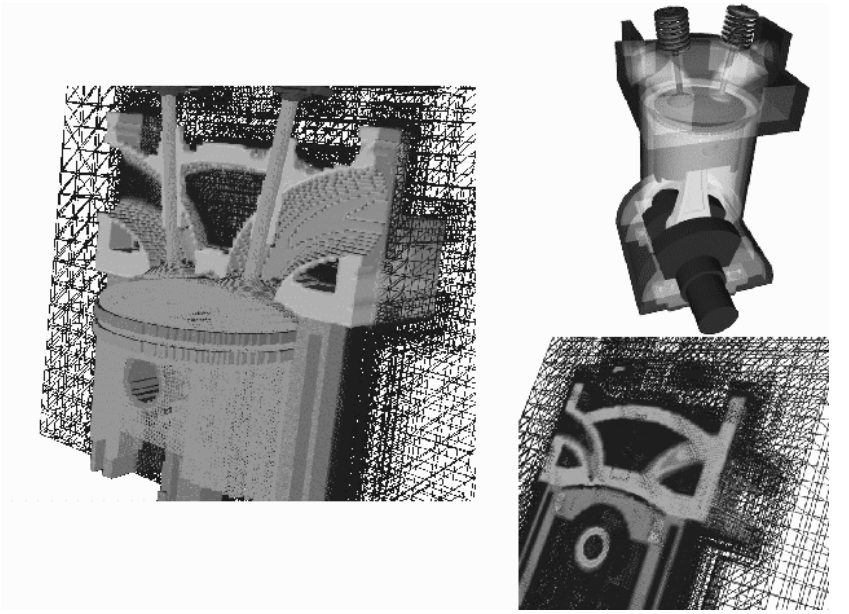


Fig. 5. An example of the use of CAD/CAM 3D solid geometry modeling to generate a continuous adaptive mesh refinement mesh for a portion of a piston engine.

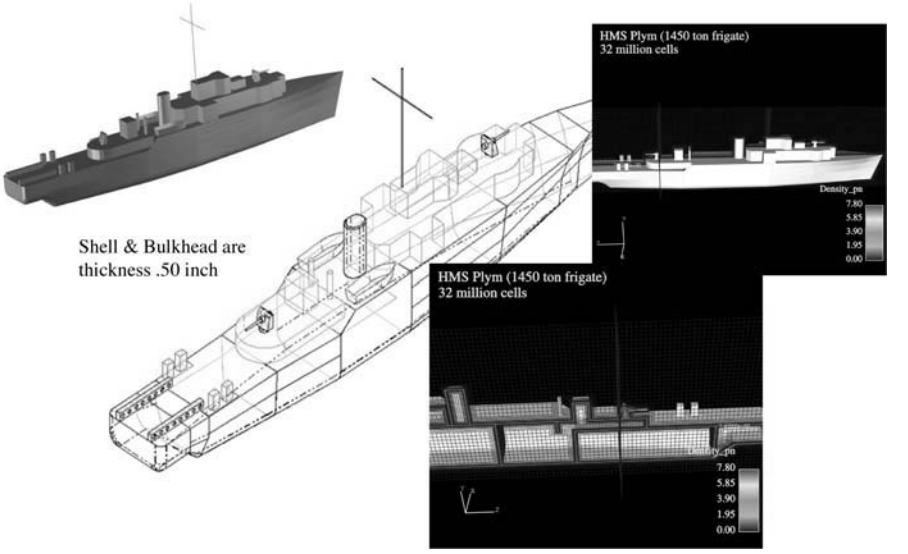


Fig. 6. An example of the use of CAD/CAM 3D solid geometry modeling to generate a continuous adaptive mesh refinement mesh for a portion of a British River-class Frigate.

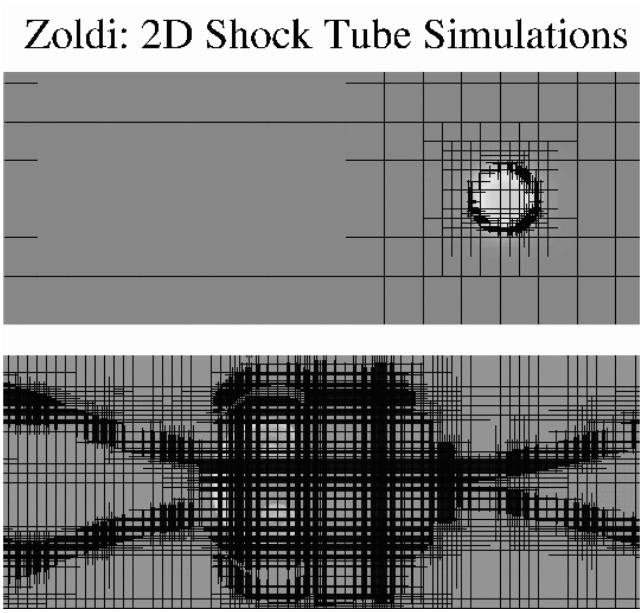


Fig. 7. An example of the continuous adaptive mesh refinement mesh for a simulation of a shock tube experiment involving a cylinder of SF_6 gas in air.

the time saved both in complex problem setup and the benefit that *no* hand-rezoning is required.

Currently, the adaption algorithms in the Crestone Project codes are based independently on two criteria: one level of resolution can be set to resolve material interfaces, while a different level of resolution can be set for the physics within a material. A simple example as shown in Fig. 7 would be to have the grid resolution between the SF_6 cylinder and the air be followed dynamically, for example, with 0.025 cm cells, while the shocks in the air (high-pressure gradients) can be followed dynamically with 0.050 cm cells, and those shocks in the SF_6 material can be followed with 0.0125 cm cells. All of these criteria are used in a dynamic sense, so that cells are created (daughter cells) and cells are coarsened into original mother cells at each time step at each cell in the problem.

A dramatic example of the 3D CAMR method is shown in Fig. 8. This Figure shows a 3D simulation of a shock-generated instability (Richtmyer-Meshkov Instability [RMI]) from the passage of a mach 1.2 shock over a perturbed surface of SF_6 in air. The initial perturbation is in the form of a cosine-cosine distribution, and this figure represents a time at which the interface between the air and SF_6 has been shocked from right-to-left and then reshocked from left-to-right in the Figure. Notice the extremely high resolution in the simulation that defines the complex interface between the two materials.

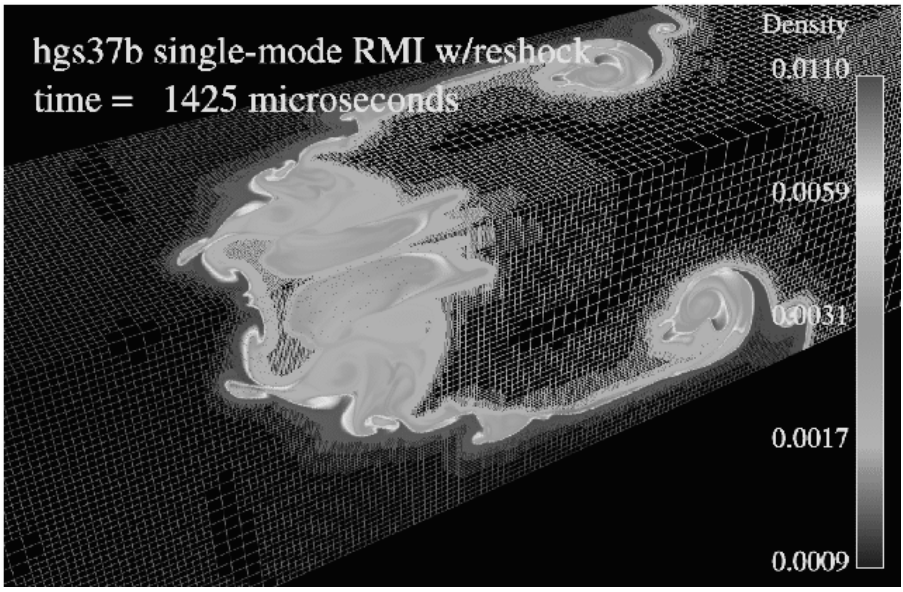


Fig. 8. An example of the 3D CAMR mesh for a simulation of a shock tube experiment involving a perturbed surface of SF_6 gas in air.

3 Examples of Massively-Parallel Simulations by the Crestone Project

In this section we will show several examples of simulations done with the Crestone project codes. The results will be shown chronologically in order to emphasis the success of the portability and scalability of the software framework. We show an evolution from one of the first generation ASCI machines (the 3 TOps Bluemountain supercomputer), to the 12 TOps White machine at Livermore, and finally to the 20 TOps Q machine at Los Alamos. Remember our goal is to have a $\sim$1000-fold decrease in wall clock time for 3D runs compared to single processor run-times.

Although some initial 3D runs were performed on the CRAY computers in 1995 and 1996, the real onset of production 3D simulations with the Crestone Project codes started with the delivery of the Bluemountain supercomputer at Los Alamos in 1997. Some of the first runs that we did were related to hydrodynamic mixing of two materials. In 1998, we did a demonstration run on the Bluemountain machine of the 3D Rayleigh-Taylor instability (RTI). A graphic from this run is shown in Fig 9. With only four levels of refinement and 310 processors we were able to finish a simulation in 360 hours of cpu-time or about 15 days of continuous computing. The actual wall-clock time required to complete this run was about one month of calendar time. A comparison we will make as a standard measure of the parallel performance of these runs is the cpu-equivalent time required to run the simulation on a single processor of the same machine. So for this 310 processor run of 360 hours the single

processor time would have been ∼13 years! The cell-count of this simulation varied with time due to the CAMR because the surface area of the interface between the two mixing fluids grew larger with time. The problem started with about 7 million cells and ended with about 45 million cells, effectively running on only 310 processors.

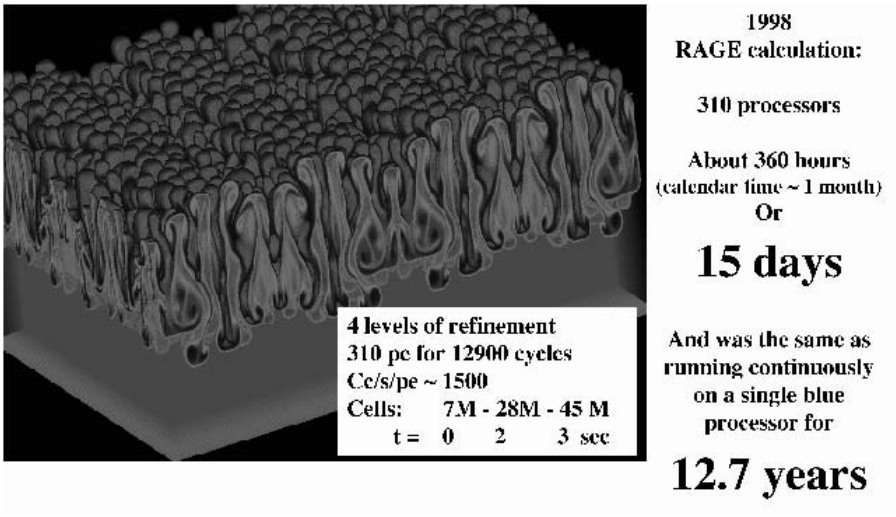


Fig. 9. The first large scale 3D simulation performed with SAGE on the Bluemountain machine: a multimode simulation of Rayleigh-Taylor mixing of two fluids of differing densities in a gravity field.

The next step in the evolution of Crestone Project 3D parallel simulations was the RMI mixing simulation shown earlier in Fig. 8. This run was modeling the nonlinear evolution of RMI from a mach 1.2 shock crossing a perturbed interface between air and SF_6 gas. Another view of this simulation is shown in Fig. 10. These runs were performed in 1999 after the full Bluemountain machine had been delivered to Los Alamos. This full machine of 6144 processors allowed us to perform a parameter study for the same initial conditions on mesh resolution (minimum cell size or maximum number of levels of refinement). In this study we ran the same 3D problem with six, seven, and eight levels of refinement successively. The level six run ran to a problem time of 1.8 ms in 166 cpu-hours or 4.8 days of continuous run

time. A summary of the three runs is shown in Table 1. The Level 6, Level 7, Level 8 runs were equivalent to 2 year, 52 year, and 239 year single processor runs (respectively), while they were actually completed in 4.8 days, 34.5 days and 91 days, respectively. So the most refined run, the Level 8 study, was run in 3 months but would have taken over two centuries to complete on a single processor machine assuming, of course that there would be enough memory! These kinds of numbers clearly demonstrate the success of the parallel implementation of the SAGE code and the power of parallel computing. None of these runs would ever have even been

Table 1. Parameter Study on Mesh Resolution for a 3D RMI Problem

Level 6	Level 7	Level 8
CPU-hours	Processors	Max Cells
116	126	15M
828	1260	98M
2187	1890	181M

started prior to the parallel hardware and parallel software. Within just two years of the start of the ASCI program, the National Laboratory design community was beginning tobelieve that massively-parallel computing would fundamentally change the scope of simulations that were done.

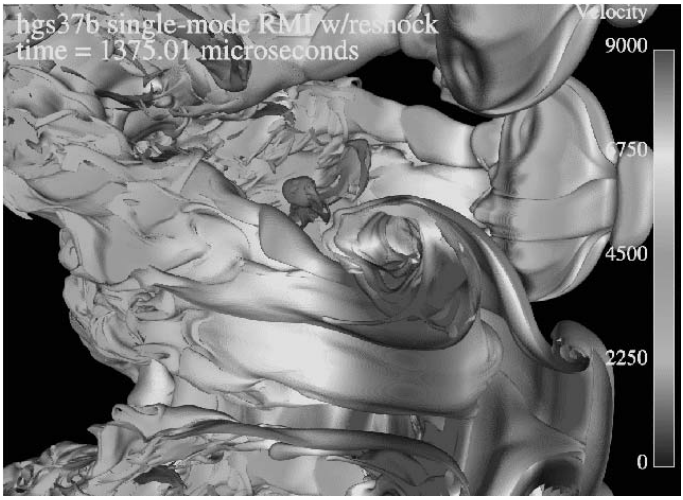


Fig. 10. The second large scale 3D simulation performed with SAGE on the Bluemountain machine: a single mode simulation of RMI mixing of SF_6 gas in air. This visualization is from the Ensight commercial graphics software from the run with seven levels of refinement.

3.1 ASCI Milepost Simulations 1999-2001

In this section we will take a short digression from the unclassified work that has been described in previous sections. As we stated earlier, there are both unclassified and classified code development efforts within the Crestone Project. Although we cannot describe any of the physics or algorithms used for the classified codes, we can summarize the magnitude of the simulations that have been performed. All of the information contained here is unclassified and has been reported in a wide variety of

unclassified venues, such as Laboratory Press Releases, newspaper articles, and DOE sponsored reports. A significant aspect of the initial phases of the ASCI program was the accomplishment of a series of major 3D simulations, known as ASCI milepost runs, that would demonstrate a progression of capability as the parallel software matured and as the size of the ASCI parallel platforms increased. From 1999 - 2001 there were very clearly defined, major 3D runs that each Laboratory was expected to complete on time. A Blue Ribbon panel of experts was formed (the Burn Code Review committee) to review the work done and determine whether the runs that had been completed actually satisfied the requirements. In each case, these initial milepost runs were single large scale 3D simulations which were to be compared to data from actual nuclear tests. It is an understatement to say that these milepost runs were extremely taxing on the resources available as well as on the personnel who developed the software and those who performed the runs. We will get back to the first of these milepost runs (1999) in a moment.

The 2000 Milepost Simulation: A 3D Secondary Prototype Performance Simulation

The first milepost simulation that the Crestone team was scheduled to perform was the 2000 3D Prototype Secondary simulation. Although there are unclassified details of the requirements for this task, we will not describe those here. Instead, we would like to continue the documentation of the success of massively-parallel simulations performed by our team by detailing the statistics of these milepost simulations. This single run was performed on the Bluemountain 3 TOps machine using about 1/3 of the processors. The typical run would use 2016 processors: 126 of the 128 processors per SMP box (or node) and 16 SMP boxes. This arrangement would leave two processors idle to process system level requirements. The 3D secondary simulation we performed used $\sim$100 million cells and ran over the course of a three-month period. It consumed 2 million CPU hours, or the equivalent of a 230 year single processor run! Each processor of the Bluemountain machine has 0.25 gigabytes (Gb) of memory, and this simulation did not stress the memory usage. The main simulation done created 15 terabytes of data. The visualization of the run was done with the Ensight commercial software package. During this timeframe ($\sim$2000) the Ensight code was serial, and it literally required the equivalent of 2 man-years to create a single 100 frame movie of the run. This process was painful and led to the high priority task of parallelizing the graphics software. With this simulation, as well as the one performed at Level 8 for the RMI 3D simulation, we had achieved our goal of a thousand-fold increase in computing power. This 2000 milepost simulation was completed in April 2000, just before the horrendous firestorm in Los Alamos. Since the run was not due to be completed until the end of the calendar year,we successfully completed the task a full nine months ahead of schedule! In the remaining time before the year-end, we actually completed another full sequence of a 3D simulation for another test. These results were all documented and provided to the Burn-Code Review committee in January 2001. The milepost runs we had performed were judged to completely satisfy the requirements.

Towards the end of the year 2000, the Crestone Project team was called upon by the Los Alamos management to help with the completion of the original ASCI

3D performance milestone, the 1999 3D primary prototype simulation. This original milepost run had been successfully completed by Livermore, but still was not complete at Los Alamos in late 2000. After finishing our scheduled milepost run well ahead of schedule, we were in a position to help complete the 1999 milestone run. This was done towards the end of the year 2000 as a joint effort from the Crestone Project team and the Shavano Project team of Los Alamos.

The 2001 Milepost Simulation: A 3D Full-System Prototype Performance Simulation

Following directly on the heels of the completion of the 1999 and the 2000 ASCI Level-1 milepost simulations, the Crestone Project team was again scheduled to perform the next ASCI milepost: a prototype full-system simulation, which was due by the end of the calendar year 2001. Early in 2001, the White 12 TOps machine became available for use by our team. This White machine was physically located at Livermore, California, and was connected to Sandia National Laboratory and Los Alamos National Laboratory by a high-bandwidth, secure Wide-Area Network (WAN). The project codes were moved to the White machine, compiled, and tested. Since we had built in portability by our choice of software, this process of obtaining a production capability on White took only a few weeks. Much of the credit for handling the details of this transition goes to Bob Boland, who spent much time living in Livermore, working with the system staff on the White machine. This milepost simulation, a 3D full-system prototype run, was (and still is) the most challenging task we have undertaken in our careers. This type of simulation had never been done before, either in 2D or in 3D, so we were not confident of a scientifically viable solution to this task until the run was almost complete! We did many of the preliminary simulations in 2D in order to demonstrate feasibility of the full-system concept with much shorter runs. This work of 2001 stands out as one of the Crestone Project team's most significant tasks. The statistics from this run are truly stunning. The main 3D simulation was done on the state-of-the-art (for its time) ASCI machine, the White machine at Livermore, using Los Alamos's entire allocation of processors – 2048. The White machine is comprised of SMP nodes with 16 processors per node. The Los Alamos allocation on this machine was 128 nodes. Thus, this simulation alone used $\sim$3 TOps of computing power. This single run took nine months to complete. It was started in January of 2001 and finished in October 2001. There were actually a couple of months during the summer of 2001 that were devoted to additional code-development, so the the actual calendar time for this run was $\sim$7 months. The actual day-to-day runs were performed in a sequence of runs that produced check-point restart files to start the next run of the sequence. Each of these individual runs was submitted for 96 hours, and many of the runs actually completed the full 96 hour allotment. Many, however, terminated prematurely for literally hundreds of differing reasons. With each terminated run, we would back up to the previously written restart file and continue. This one run consumed over 6 million cpu hours and is equivalent to a single-processor run that would take over 700 years to complete! With the associated 2D and other 3D simulations that went into this effort, we are calling this

run our "millennium" simulation. The maximum cell count used was ∼500 million cells. Restart files ranged in size (depending on the physics) from ∼100 Gb to 240 Gb each. The wall-clock time to run one cycle ranged from 4 minutes to ∼15 minutes. The parallel I/O on the White machine was impressive: average numbers for this run are ∼1 Gb/sec writes and ∼2 Gb/sec reads, so reading a 240 Gb restart file took only ∼2 minutes. The total data written from this run amounted to over 200 terabytes (Tb), with about 20 Tb devoted to Ensight graphics files. The size of the Library of Congress is about the same.

In addition to these amazing statistics, one should realize that this entire process was carried out *remotely*. The runs were actually done at Livermore, while we resided in Los Alamos, NM. Although it is fact of life these days to run remotely on computers around the world, the truly amazing accomplishment was the ability to manipulate and visualize the extremely large data sets involved. By this time (2001) we had a parallel version of the Ensight client-server commercial graphics software (called the server-of-servers [SoS]) that domain decomposed the geometric mesh into blocks (typically 50 - 64) of ∼5 million cells each. With this software, we were able to leave the large data-sets on the White GPFS, use the SoS on several White nodes devoted to visualization, and display the graphics back at Los Alamos on a workstation running the Ensight Client software. This process worked well. We essentially had real-time graphics from our runs a thousand miles away, without the need to move terabytes of data around the country. In fact, the White GPFS was the only file system available that had the capacity to hold these large data sets. This simulation was presented to the Burn-code review panel in January 2002 and was judged to meet or exceed all of the requirements of the milepost. Our thanks go out to all who help make this effort successful, from the Project team members, to the consultants at Los Alamos, to the staff at the Livermore Computing center.

We think it is very clear from this progression of massively-parallel computing simulations, that the concept of parallel computing has been extremely successful. In 1996 at the start of the ASCI program, we could not find many in our community that would have believed this statement.

3.2 Current Massively-Parallel Simulations

Since the completion of the 2001 ASCI milepost simulation, the definition of these mileposts and the overwhelming dedication required to perform them have been scaled back. Current ASCI milestones are more directed at programmatic day-to-day work, as opposed to the performance of a "demonstration" simulation. We learned a lot from the large scale runs we performed, but they also take a significant toll on the personnel involved. Although large-scale 3D simulations continue today, the complexity and the computer science required have not changed much since 2001. Of note, however, is the delivery of the ASCI Q (HP/Compaq alpha processor) machine at Los Alamos in 2002. During the delivery of this machine, a 10 TOps section was devoted to unclassified science runs. One of our team members, Galen Gisler, applied for, and was granted time on this machine for a science run. Gisler's work is featured at this conference and we would direct interested readers to his work ([26],[27]). For

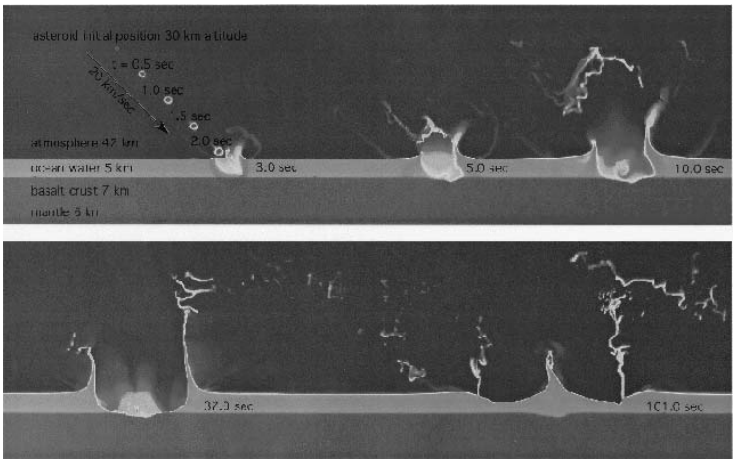


Fig. 11. A montage of 10 separate images from the 3-d run of the impact of a 1-km iron bolide at an angle of 45 degrees with an ocean of 5-km depth. These are density raster graphics in a two-dimensional slice in the vertical plane containing the asteroid trajectory. Note the initial uprange-downrange asymmetry and its disappearance in time. Maximum transient crater diameter of 25 km is achieved at about 35 seconds. The maximum crown height reaches 30 km, and the jet seen forming in the last frame eventually approaches 60 km.

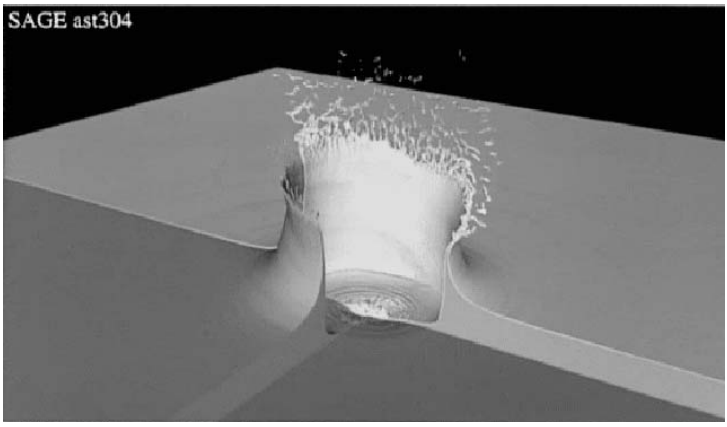


Fig. 12. Perspective plot of three isosurfaces of the density from the 3-d run of a 45-degree impact of a 1-km iron bolide into an ocean of 5-km depth at a time 30 seconds after the beginning of the calculation (27.5 seconds after impact). The isosurfaces are chosen so as to show the basalt underlayment, the bulk of the ocean water, and cells containing water spray (mixed air and water). The asymmetry of the crown splash is evident, as is its instability to fragmentation. Cratering in the basalt is seen, to a depth of 1 km. The diameter of the transient cavity is at this time 25 km.

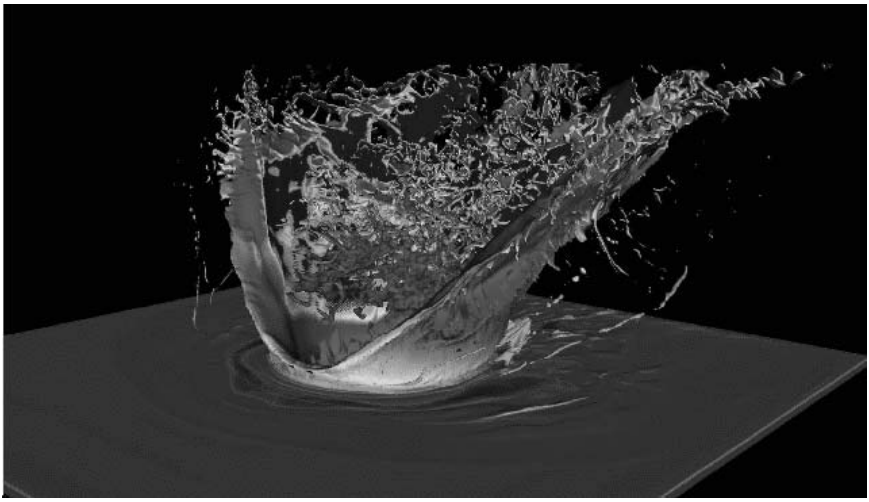


Fig. 13. A simulation of the impact crater at the Chicxulub site in the Yucatan Peninsula of Mexico. Forty-two seconds after impact, the ”rooster tail” has left the simulation volume and gone far downrange. The image shows the extent of the calculational mesh: 256 km on a side. The dissipation of the asteroid’s kinetic energy, some 300 teratons TNT equivalent, produces a stupendous explosion that melts, vaporizes, and ejects a substantial volume of calcite, granite, and water. The dominant feature in this picture is the ”curtain” of the debris that has been ejected and is now falling back to Earth. The ejecta follows ballistic trajectories with its leading edge forming a conical surface that moves outward from the crater as the debris falls to form the ”ejecta blanket”. The turbulent material interior to the debris curtain is still being accelerated upward by the explosion produced during the excavation of the crater.

this discussion, Gisler’s 3D simulations started on the White machine and continued on to the ASCI Q machine at Los Alamos. The scale of these runs is comparable to those of the milepost runs and the highly-resolved 3D RMI work shown previously. Here we show some images from Gisler’s Asteroid impact work: Figs. 11, 12 and 13.

4 Verification and Validation for SAGE and RAGE

During the course of the Crestone Project activities, substantial time has been devoted, either directly or indirectly, to the verification and validation (V&V) of the Project codes. Direct V&V results from a Project team member running specific simulations to compare to analytic results (verification) or to results from experimental data (validation). Indirect validation results from the use of the Project codes by end-users for their own purposes, either in designing a new AGEX experiment, or in the analysis of a previous experiment, or merely as a tool for analysis of a physical problem. There exists a large suite of verification test problems in our community, including the Sedov blast wave, various geometries for the Noh problem, Marshak

radiation wave problems, and many more. Details of the specifications and results from these types of problems is written up elsewhere and will not be discussed at length here. A detailed report on the suite of verification problems is being prepared (see e.g., [31],[32]). One example of such a verification problem is the Noh problem, with uniformly converging flow velocities. The results from the cylindrical Noh problem are shown in Fig. 14, comparing the SAGE adaptive mesh hydrodynamics to the analytic result. In general we are very pleased with the success of the SAGE and RAGE algorithms in comparing to analytic and semi-analytic results (see e.g., [2] for an example of comparisons to semi-analytic problems).

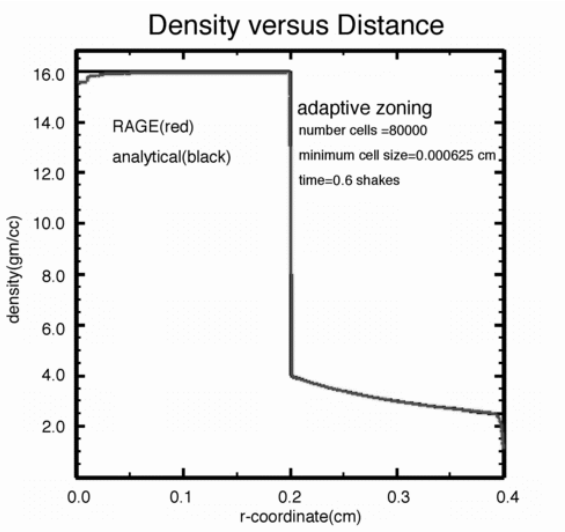


Fig. 14. An example from our verification suite: the cylindrical Noh problem. The red curve is the SAGE CAMR result compared to the black analytic solution.

Direct project validation runs are subject to conflict-of-interest discussions, so in this brief section on validation of the Project codes, we will refer the reader to several end-user applications that clearly demonstrate a high level of code algorithm validation (in some cases, e.g., [24], [30] results of code simulations are used for design and prediction of future experiments). The first validation paper published in a refereed journal was "The Simulation of Shock Generated Instabilities" [4] in which detailed code simulations of shock tube experiments were quantitatively compared to the experimental results. The results from this work are summarized in Fig. 15.

Another example of code validation is given in C. Zoldi's Ph.D. thesis for SUNY Stony Brook [33], [7]. In this work, code simulations were done for a similar shock tube environment, but in this case the mach 1.2 shock interacted with a dense cylinder of gas, instead of a gas curtain. The reader is referred to her thesis work, with one example of comparisons between experimental images and SAGE results shown here in Fig. 16.

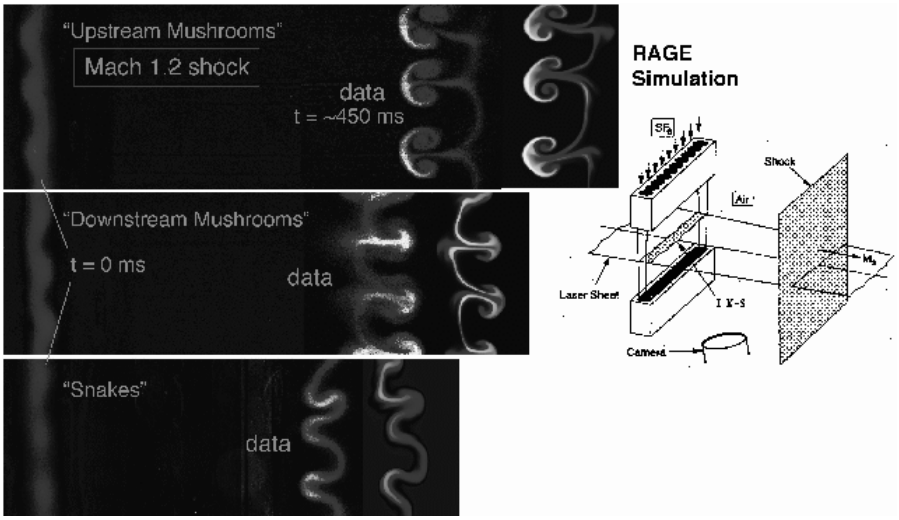


Fig. 15. A graphical comparison of three shock tube experiments to the corresponding RAGE simulations. The three experiments had different initial conditions for the SF_6 gas curtain in the shock tube that resulted in three drastically different dynamical instabilities. The RAGE CAMR hydrodynamics solutions for these experiments are shown offset from the data and is analyzed quantitatively in [4].

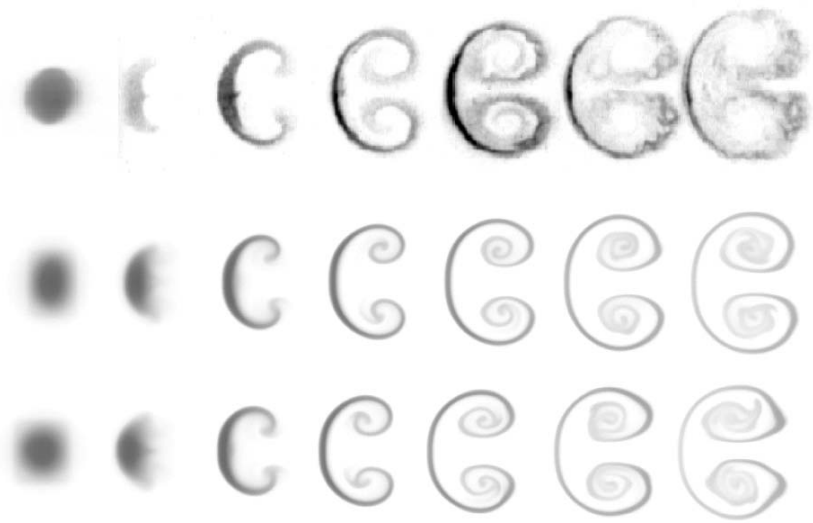


Fig. 16. A graphical comparison of shock tube experimental images to the corresponding SAGE simulations. The images show a progression of dynamical shapes as the evolution of the initial cylinder moves downstream. The top sequence shows experimental images and the bottom two show corresponding numerical simulations. Also included in Zoldi's work is a good comparison of code results for velocity distributions to those obtained from the experiment.

The final example here for validation is the work being done as a multi-lab, indeed, a multinational collaboration on the simulation of supersonic jets and shock interactions [24]. I refer to this work as validation of the "complex hydrodynamics" of the CAMR Eulerian Godunov algorithms in RAGE. Detailed, quantitative intercomparisons have been done among several computational tools (including RAGE), and the actual experimental data obtained from a series of AGEX experiments. All the code techniques, including CAMR RAGE and ALE codes from both Livermore and AWE (the UK Atomic Weapons Establishment), show good agreement with the data.

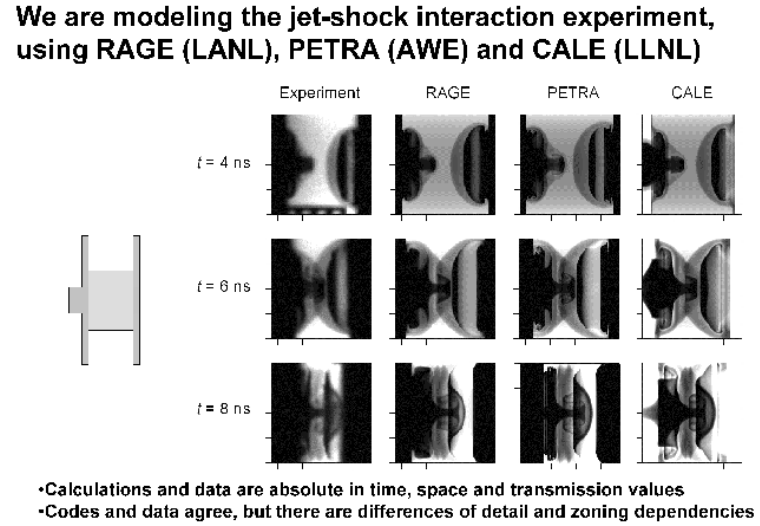


Fig. 17. Multinational modeling of complex hydrodynamics from supersonic jets and shock interactions compared to experimental results. RAGE code was used in the design of the experiments.

These examples touch on the breadth of validation efforts completed and underway in the on-going effort to understand the range of capability for the CAMR Eulerian-based hydrodynamics and radiation energy flow as coded in the massively-parallel, modular version of the RAGE code. More detailed documentation efforts are underway. Two of the areas of active research in the Crestone Project currently are the treatment of interfaces between materials and the improvement of the material strength package. Both of these areas of physics in the project have been flagged for improvement, and significant progress in this direction is forthcoming.

5 Summary

An overview of the Los Alamos Crestone Project has been given. This project is arguably one of the key success stories of the DOE ASCI program. We have shown a progression of capabilities in massively-parallel simulations, starting from runs of order 100 processors to state-of-the-art multi-physics simulations using thousands of processors and consuming months of supercomputer time. We have demonstrated that the RAGE code is a sophisticated CAMR hydrocode, with a long history of verification and validation. The Crestone Project user community tends to dominate the cycles used on all available high-end supercomputers, but the codes SAGE and RAGE can also be run efficiently on desktop machines, even Macintosh's under MacOS X unix environment. The Crestone Project user-community is fully utilizing each new ASCI supercomputer that is delivered to the complex.

We hope to have demonstrated from the discussion here that massively-parallel computing is fundamentally changing the way we think about computer simulations: simulations that even just a few short years ago were *unthinkable* are not not only being done, but are becoming routine. The 2D and 3D simulations performed by the Crestone Project team and its end-user community in the last two years represent orders of magnitude more computation than was done by all DOE calculations previously performed.

Acknowledgement. The authors would like to thank all the members of the Crestone Project team, both past and present. Special thanks go to Mike Clover and Bill Archer, the former and current code-project leaders. It takes a very special talent to coordinate and actually develop and improve such complex, multi-physics integrated codes as exist in the Crestone Project. We have been extremely fortunate to have found that talent in these two individuals. The unique collaboration between SAIC and Los Alamos has proven to be highly successful, primarily due to the outstanding individuals on our team. Bob Greene and Bob Kares continue to produce outstanding graphics from our simulations using the Ensight visualization software, a product of CEI, Inc. many of which are shown in this paper.

This work was supported by the University of California, operator of the Los Alamos National Laboratory under Contract No. W-7405-ENG-36 with the U.S. Department of Energy.

References

1. "The SAGE Code Multimaterial Equation of State Methods" M.L. Gittings in Numerical Methods Symposium, 28-30 April 1992; "The RAGE Code" R.N. Byrne, T. Betlach, and M.L. Gittings in Numerical Methods Symposium, 28-30 April 1992. Copies may be ordered from the Defense Nuclear Agency (currently the Defense Threat Reduction Agency), 56801 Telegraph Road, Alexandria, VA 22310-3398.
2. "2D and 3D simulations of RM instability growth with RAGE: a continuous adaptive mesh refinement code" R. Weaver, M.L. Gittings, R.M. Baltrusaitis, Q. Zhang and S.Sohn, July 1997, proceedings of the 21st International Symposium on Shock Waves (ISSW21), Australia, paper 8271.

3. “The parallel implementation of RAGE: a 3D continuous adaptive mesh refinement radiation-hydrodynamics code” R. Weaver, M.L. Gittings, M.L. Clover, July 1999, proceedings of the 22st International Symposium on Shock Waves (ISSW22), London, paper 3560.
4. “The Simulation of Shock-Generated Instabilities” R.M. Baltrusaitis, M.L. Gittings, R. Weaver, R. Benjamin and J. Budzinski 1996, Physics of Fluids, 8 (9), p. 2471 (also as LA-UR-95-4000).
5. “Predictive Performance and Scalability Modeling of a Large-Scale Application” Darren J. Kerbyson, Hank J. Alme, Adolfy Hoisie, Fabrizio Petrini, Harvey J. Wasserman, and Michael Gittings, in Proc. of IEEE/ACM SC2001, Denver, November 2001.
6. “Verifying Large-Scale System Performance During Installation using Modeling” Darren J. Kerbyson, Adolfy Hoisie, and Harvey J. Wasserman, in Hardware/Software Support for Parallel and Distributed Scientific and Engineering Computing, L.T. Yang (eds), Kluwer, September 2003.
7. “Code Validation Experiments — A Key to Predictive Science” Brian Fishbine, Los Alamos Research Quarterly, Fall 2002, LALP-02-194.
8. “Two and Three Dimensional Calculations of a NOVA Hydro Instability Experiment” B.H. Wilde, F.J. Swenson, R.P. Weaver, M.L. Gittings, and W.J. Powers Proceedings of the 6th International Workshop on Compressible Turbulent Mixing, France, July, 1997.
9. “Hydrothermal Pressure Instabilities related to Magmatic Steam Injection and Reflected in Long-Period Seismicity” B.A. Chouet, M.M. Morrissey, M.L. Gittings, and R. Weaver, December 1997, EOS, Transactions, American Geophysical Union 1997 Fall meeting, Vol 78, p F764.
10. “ICF Targets with Joints and Gaps” S. R. Goldman, M. D. Wilke, D.C. Wilson, S.E. Caldwell, W. W. Hsing, R. P. Weaver, Bull. Am. Phys. Soc. 41, 1353 (1996)
11. “Inertial-Confinement Fusion at the Los Alamos National Laboratory: Current Trends in International Fusion Research” Erick Lindman, D. Baker, C. Barnes, B. Bauer, J.B. Beck, R. Berggren, B. Bezzerides, P. Bradley, R. E. Chrien, M. Clover, J. Cobble, C. A. Coverdale, M. Cray, N. Delamater, D. DuBois, B. H. Failor, J. C. Fernandez, L. Foreman, R. Gibson, P. Gobby, S. R. Goldman, D. Harris, A. Hauer, J. Hoffer, N. Hoffman, W. W. Hsing, R. Johnson, K. Klare, R. Kopp, W. Krauser, G. Kyrala, G. Magelssen, R. Mason, D. Montgomery, T. J. Murphy, J. Oertel, G. Pollak, H. Rose, K. Schoenberg, D. P. Smitherman, M. S. Sorem, F. Swenson, D. Tubbs, W. Varnum, H. Vu, J. Wallace, R. Watt, R. Weaver, B. Wilde, M. Wilke, D. Wilson, W. M. Wood, Review and Assessment, Washington, DC, Proceedings of the 2nd Symposium, Plenum Press (March 10-14, 1997).
12. “Two and Three Dimensional RAGE Calculations of a NOVA Instability Experiment” B.H. Wilde, F.J. Swenson, R.P. Weaver, M.L. Gittings, and W.J. Powers, presented at the JOWOG 32m, Lawrence Livermore National Laboratory, August 13, 1997.
13. “Two and Three Dimensional Calculations of a NOVA Hydro Instability Experiment” B.H. Wilde, F.J. Swenson, R.P. Weaver, M.L. Gittings, and W.J. Powers proceedings of the 6th International Workshop on Compressible Turbulent Mixing, France, July, 1997.
14. “Shock Propagation in the Nonlinear Regime” W.W. Hsing, S. Caldwell, S.R. Goldman, R.P. Weaver, Bull Am. Phys Soc. 42, 1952 (1997)
15. “Shock Structuring Due to Fabrication Joints in ICF Targets” S.R. Goldman, S.E. Caldwell, G.A. Kyrala, M.D. Wilke, D.C. Wilson, C.W. Barnes, W.W. Hsing, N.D. Delamater, J. Grove, J.M. Wallace, R.P. Weaver, A.M. Dunne, M.J. Edwards, P. Graham, B.R. Thomas, Bull Am Phys. Soc. 43, 1667 (1998)
16. “Simulations of Richtmyer-Meshkov Instability in Two and Three Dimensions” Richard L. Holmes, Cindy A. Zoldi, Robert P. Weaver, Michael Gittings and Michael Clover,

July 1999, proceedings of the 22st International Symposium on Shock Waves (ISSW22), London, paper 4480.
17. "Shock Structuring Due to Fabrication Joints in Targets" S.R. Goldman, S.E. Caldwell, M.D. Wilke, D.C. Wilson, C.W. Barnes, W.W. Hsing, N.D. Delamater, G.T.Schappert, J. W. Grove, J.M. Wallace, Robert P. Weaver, A.M. Dunne, M.J. Edwards, P. Graham, and B. R. Thomas, Physics of Plasmas, 6(8), pp. 3327 – 3336, 1999. [also as LA-UR-98-4250].
18. "Richtmyer-Meshkov Instability Growth: Experiment, Simulation and Theory" Richard L. Holmes, Guy Dimonte, Bruce Fryxell, Michael L. Gittings, John W. Grove, Marilyn Schneider, David H. Sharp, Alexander L. Velikovich, Robert P. Weaver, and Qiang Zhang, Journal of Fluid Mechanics 39, pp. 55 –79. [Also as LA-UR-97-2606].
19. "Planar,ablative Rayleigh Taylor Instability Growth in Copper Foils" G. T. Schappert, W.W. Hsing,D. E. Hollowell, R. P. Weaver and B. A. Remington 38th Annual Meeting of the Division of Plasma Physics; 11-15 November, 1996; Denver CO
20. "Rayleigh-Taylor Instability Growth in Copper Foils Driven by NOVA Hohlraum Radiation" G. T Schappert, W.W. Hsing,D. E. Hollowell, R. P. Weaver and B. A. Remington; 37th Annual Mtg. of the Division of Plasma Physics; November 6-10, 1995; Louiville,Ky
21. "Ablative Rayleigh-Taylor Instability Modeling" D. Hollowell, G. Schappert, S.Caldwell, W. Hsing, R. Weaver; June 1997 Anomolous Absorption Conference, Vancouver Canada
22. "2-D Rayleigh-Taylor Experiments and Modeling" David Hollowell, Gottfried Schappert, Robert Weaver, Warren Hsing, Rodney Mason, Steve Caldwell; LA UR 98-1828 and 25th European Conference on Laser Interaction With Matter, Formia, Italy May, 1998
23. "Production of Enhanced Pressure Regions due to Inhomogeneities in Interial Confinement Fusion," S.R. Goldman, C.W. Barnes, S.E. Caldwell, D.C. Wilson, S.H. Batha, J. W. Grove, M.L. Gittings, W.W. Hsing, R.J. Kares, K.A. Klare, G.A. Kyrala, R.W. Margevicius, R. P. Weaver, M.D. Wilke, A.M. Dunne, M.J. Edwards, P. Graham, and B. R. Thomas, Physics of Plasmas, 7(5), pp. 2007 – 2013, 2000.
24. "Supersonic Jet and Shock Interactions," J.M. Foster, B.H. Wilde, P.A. Rosen, T.S. Perry, M. Fell, M.J. Edwards, B.F. Lasinski, R.E. Turner and M.L. Gittings, Physics of Plasmas, 9(5) Part 2, pp. 2251 – 2263, 2002.
25. "On 3D, Automated, Self-Contained Grid Generation Within the RAGE CAMR Hydrocode" by W.R. Oakes, P.J. Henning, M.L. Gittings, and R.P. Weaver, in proceedings of the 7th International Conference on Numerical Grid Generation in Computational Field Simulations, p 973, Chateau Whistler Resort, September 2000.
26. "Two- and Three-Dimensional Simulations of Asteroid Ocean Impacts" by Galen Gisler, Robert Weaver, Charles Mader, and M.L. Gittings, Los Alamos internal report, LA-UR-02-30, 2002.
27. "Two- and Three-Dimensional Simulations of Asteroid Impacts" by Galen Gisler, Robert Weaver, M.L. Gittings, and Charles Mader, Los Alamos internal report,accepted for publication in Computers in Science and Engineering Journal, 2004.
28. "The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations" P. Colella and P. R. Woodward, J. of Computational Phys. 54(1):174-201, 1984.
29. "Local adaptive mesh refinement for shock hydrodynamics" Berger, MJ and Collela, P (1989), J. Comp. Phys. 82, 64-84
30. private discussion with Dr. B. H. Wilde of Los Alamos National Laboratory.
31. "2-D Convergence Analysis of the RAGE Hydrocode" J. R. Kamm and W. J. Rider, Los Alamos National laboratory internal report, LA-UR-98-3872.
32. "Comparing the Fidelity of Numerical Simulations of Shock-Generated Instabilities" J. R. Kamm, W. J. Rider, P.V. Vorobieff, P.M. Rightley, R.F. Benjamin, K. P. Prestridge,

July 18-23, 1999, proceedings of the 22st International Symposium on Shock Waves (ISSW22), London, paper 4259.
33. “A Numerical and Experimental Study of a Shock-Accelerated Heavy Gas Cylinder” C. Zoldi Ph.D. thesis for the State University of New York at Stony Brook, May 2002.

Part II

Methods

Adaptive Mesh Refinement on Overlapping Grids

William D. Henshaw

Centre for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, henshaw1@llnl.gov

In this chapter a short description will be given of the use of block structured adaptive mesh refinement with overlapping grids. The combination of overlapping grids and AMR leads to a powerful approach for efficiently solving problems with multiple space and time scales in complex geometry. Sample calculations will be presented demonstrating the approach on problems ranging from a simple convection-diffusion equation to the reactive Euler equations.

1 Background

A composite overlapping grid consists of a collection of structured grids that cover a domain. In the typical situation, as shown in Fig. (1), curvilinear grids conform to the curved boundaries while one or more Cartesian grids fills the interior. The overlapping grid method, as discussed in more detail in Chesshire and Henshaw [CH90], allows complex domains to be represented with smooth grids that can be aligned with the boundaries. The use of smooth grids is advantageous in many problems in order to reduce grid induced numerical artifacts. Also, the majority of an overlapping grid often consists of Cartesian grid cells so that the speed and low memory usage inherent with such grids is retained. The first overlapping grid computations were apparently performed by Starius [Sta77b, Sta77a, Sta80]. Since then overlapping grids have been used successfully for the numerical solution of a variety of problems involving inviscid and viscous fluid flows, see [CH90, SB87, HC87, HF91, OY93, Mea93, PSM$^+$93, MB94, TF95, Pet99, Mea99], for example.

Block structured adaptive mesh refinement was originally developed by Berger and Oliger [BO84] for hyperbolic equations. In this approach a hierarchy of refinement grids is constructed dynamically based on a suitable error estimate of the solution. Every few time steps a new error estimate is computed and a completely new hierarchy of grids is determined. Combining the AMR approach with overlapping grids presents a number of challenges. Some of these issues will be addressed here in the context of the AMR capabilities in Overture[BHQ99]. Overture is an objected

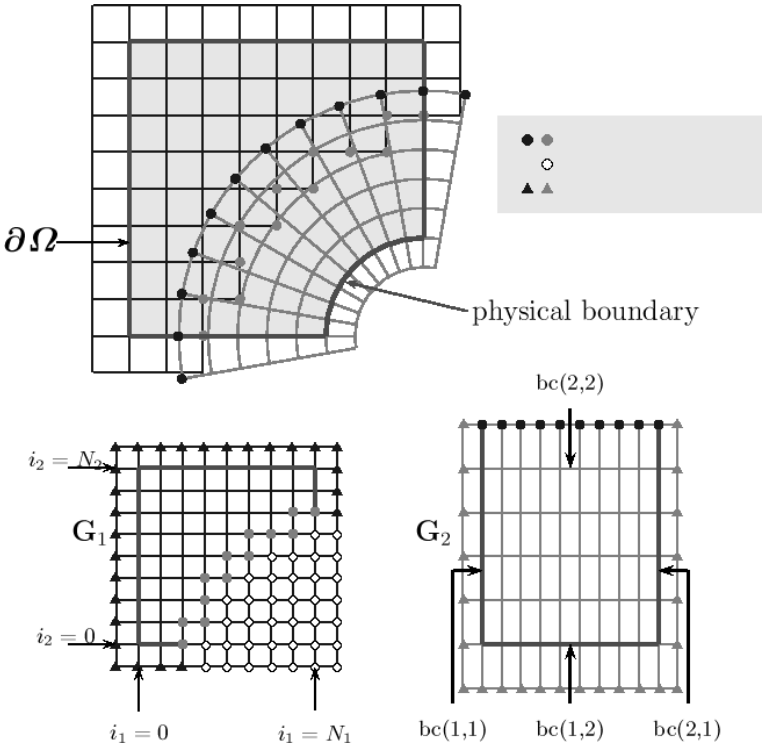


Fig. 1. An overlapping grid consisting of two structured curvilinear component grids. Each component grid is represented by a mapping from the unit square to physical space. Each grid point is classified as either a discretization point, interpolation point or unused point. Ghost points are used to apply boundary conditions.

oriented framework that can be used to solve PDE's on overlapping grids. Overture has a variety of classes that support the AMR functions of error estimation, regridding and interpolation. This AMR infrastructure includes support for curvilinear grids and the issues involved when refinement patches meet at the interface between different base grids of an overlapping grid. Overture is freely available for download from the internet [BCF$^+$03].

The coupling of overlapping grids with AMR has been considered by a number of researchers. Meakin[Mea99] has used adaptive mesh refinement on the off-body Cartesian grids of moving overlapping grids. Boden and Toro[BT97] have used AMR on overlapping grids to solve the Euler equations in two-dimensions. In Brislawn, Brown, Chesshire and Saltzman[BBCS95] AMR was added to the CMPGRD framework; CMPGRD was the Fortran based precursor to Overture.

```
CompositeGrid cg;  // create a composite grid
getFromADataBaseFile(cg,"myGrid.hdf");
floatCompositeGridFunction u(cg);  // create a grid function
u=1.;
CompositeGridOperators op(cg);  // operators
u.setOperators(op);
float t=0, dt=.005, a=1., b=1., nu=.1;
for( int step=0; step<100; step++ )
{
  u+=dt*( -a*u.x()-b*u.y()+nu*(u.xx()+u.yy()) ); // forward Euler
  t+=dt;
  u.interpolate();
  u.applyBoundaryCondition(0,dirichlet,allBoundaries,0.);
  u.finishBoundaryConditions();
}
```

Fig. 2. Overture supports a high-level interface. This sample C++ program can be use to solve the convection diffusion equation, $u_t + au_x + bu_y = \nu(u_{xx} + u_{yy})$, with a simple forward-Euler time-stepping scheme

2 AMR and Overlapping Grids

This section discusses overlapping grids and adaptive mesh refinement and how they fit together.

Overlapping Grids:

A composite overlapping grid $\mathcal{G}$ for a domain Ω in d dimensions consists of a collection of overlapping component grids G_g that cover Ω and match the boundary $\partial\Omega$:

$$\mathcal{G} = \{G_g\}, \qquad g = 1, 2, \ldots, \mathcal{N}_g \, .$$

Each component grid is a logically rectangular, curvilinear grid defined by a smooth mapping $\mathbf{C}_g$ from parameter space $\mathbf{r} \in [0,1]^d$ (the unit-square or unit-cube) to physical space $\mathbf{x} \in \mathbb{R}^d$:

$$\mathbf{x} = \mathbf{C}_g(\mathbf{r}) \quad \text{(mapping defining the component grid)}.$$

The mapping is used to define grid points at any desired resolution as required when a grid is refined.

In Fig. (1) a simple overlapping grid is shown consisting of two component grids, an annular grid and a background Cartesian grid. The figure shows the overlapping grid in physical space in addition to each component grid in its parameter space. In this example, the annular grid cuts a hole in the Cartesian grid resulting in a number of unused points marked by open circles. Points on a component grid are classified as discretization points (where the PDE or boundary conditions are discretized), interpolation points or unused points. This classification of points is determined by the

overlapping grid generator Ogen [Hen98] and stored in an integer mask array. The bit representation of each entry of the mask holds additional grid information including, for example, which points are hidden by refinement grids.

Values of the solution at interpolation points of some given grid are determined by interpolation from *interpolee* (donor) points on another grid. Interpolation is performed in the unit-square coordinates using standard tensor-product polynomial interpolation. The appropriate order for interpolation is discussed in Chesshire and Henshaw [CH90].

The Overture framework supports the generation of overlapping grids and the solution of partial differential equations (PDEs) on these grids. Overture can be used at a high level to define PDE solvers. An example of this is shown in Fig. (2). This high level interface is useful for prototyping new algorithms. A more efficient approach is to use a lower level interface. For efficiency, many of the computational kernels in Overture are written in Fortran 77.

```
PDEsolve(G, t_final)
// Advance a solution in time on a overlapping grid
{
  t := 0; n := 0;
  u_i^n := applyInitialCondition(G);
  while t < t_final
    if(n mod M = 0)  // regrid?
      e_i := estimateError(G, u_i^n);
      G* := regrid(G, e_i);  // Build a new grid
      u_i* := interpolateToNewGrid(u_i^n, G, G*);
      G := G*; u_i^n := u_i*;
    end

    Δt := computeTimeStep(G, u_i^n);
    u_i^{n+1} := timeStep(G, u_i^n, Δt);

    t := t + Δt;  n := n + 1;

    interpolate(G, u_i^n);
    applyBoundaryConditions(G, u_i^n, t);
  end
}
```

Fig. 3. A pseudo-code algorithm for solving a time-dependent PDE with adaptive mesh refinement

Adaptive Mesh Refinement

An algorithm for advancing a solution on a AMR grid is given in Fig. (3). The basic components of the AMR algorithm, as presented in the algorithm, are error estima-

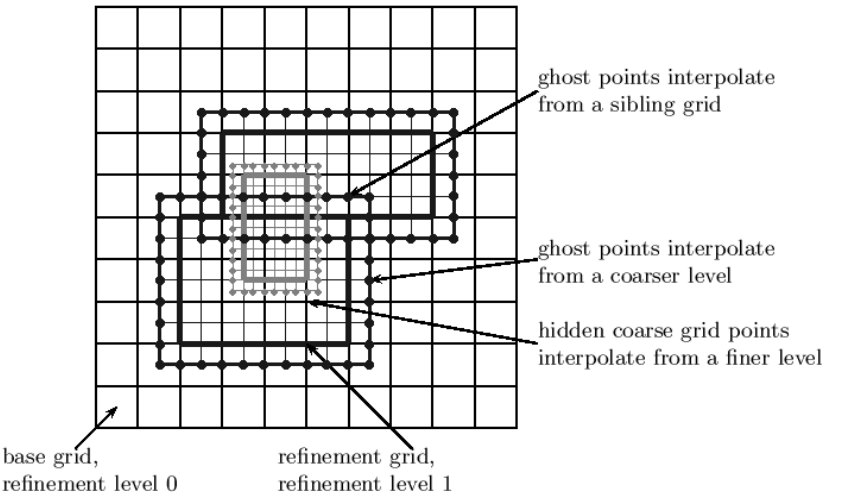


Fig. 4. An example showing three levels, a base grid and two refinement levels, of a block structured AMR grid. Ghost points on refinement grids are interpolated from sibling grids at the same level or parent grids on the next coarser level. Coarse grid points are interpolated where they are covered by refinement grids.

tion, regridding and interpolation. These AMR components are now described along with the changes required for use with overlapping grids.

Error Estimation:

The purpose of error estimation is to identify and tag cells where additional refinement is required. A good general purpose error estimator can be based on a combination of first and second differences in the numerical solution:

$$e_{\mathbf{i}} = \sum_{k=1}^{m} \frac{1}{d} \sum_{\alpha=1}^{d} \left(\frac{c_1}{s_k} |\Delta_{0\alpha} u_{k,\mathbf{i}}| + \frac{c_2}{s_k} |\Delta_{+\alpha}\Delta_{-\alpha} u_{k,\mathbf{i}}| \right) \tag{1}$$

Here s_k is a scale factor for $u_{k,\mathbf{i}}$, c_1 and c_2 are constants (weights), and $\Delta_{0\alpha}$, $\Delta_{+\alpha}$ and $\Delta_{-\alpha}$ are the un-divided central, forward and backward difference operators, respectively, in the α-direction in index space. For smooth solutions, the scaled undivided differences should be small when the grid is sufficiently fine.

After the error estimate is computed for all grids, it is smoothed with a few sweeps of an under-relaxed Jacobi iteration. At the end of each sweep, the error is interpolated to neighboring component grids. This smoothing process will propagate the error to nearby grid cells whether they be on refinement grids from the same component grid or on neighboring component grids. In the latter case, the error smoothing ensures that refinement grids are created across the overlap ahead of any approaching feature. As a result, by the time the feature reaches the overlap, refinement grids will already be in place on the neighbouring component grid.

Regridding

The adaptive mesh refinement approach adds new refinement grids in regions where the error is estimated to be large. The refinement grids are aligned with the underlying base grid and are arranged in a hierarchy with the base grids belonging to level $\ell = 0$, the next finer grids being added to level $\ell = 1$ and so on, see Fig. (4). Grids on level ℓ are refined by a refinement ratio r from the grids on level $\ell - 1$. Refinement ratios of $r = 2$ and $r = 4$ are commonly used. The grids are properly nested which means that a grid on level ℓ is completely contained in the set of grids on the coarser level $\ell - 1$ (except at physical boundaries where refinement grids are allowed to align with the boundary).

The adaptive grid hierarchy is usually rebuilt after every $M = r \times b$ time steps, where r is the refinement ratio and b is the number of buffer zones (discussed below). Cells are tagged where the error estimate exceeds a chosen tolerance. Following the algorithm of Berger and Rigoutsos [BR91], a set of boxes is generated in index space which covers the region of tagged cells, and these boxes form the boundaries of the new refined grids. This regridding step is repeated independently for each base grid of the overlapping grid. Since there is a non-negligible computational cost associated with regridding, it is desirable to increase the number of time steps that can be taken safely for the current AMR grid. To do this, the boundary of the region of tagged cells is increased slightly according a chosen integer b, known as the number of buffer cells.

Determination of the location of the boxes in index space is usually a very fast process. However, after the new boxes have been created, it is necessary to determine the location of the new grid points in physical space, their classification (i.e. discretization, interpolation or unused), and their connectivity to neighboring grids. For a non-Cartesian grid, the grid point locations are determined by evaluating the mapping, $\mathbf{x} = \mathbf{C}_g(\mathbf{r})$, associated with the grid, a feature of our AMR framework that is particularly important when refining boundary fitted grids. For interpolation points on the boundary between discretization and unused points, it is necessary to determine the donor grid from which grid to interpolate, see Fig. (5). Some care is required to make this step efficient. An algorithm have been developed that uses information from coarser refinement levels to determine the required information for finer levels. There is sometimes more than one choice for donor grid; the order of preference is to firstly interpolate from a refinement grid at the same level belonging to a different base grid, or secondly interpolate from a refinement grid at a lower level belonging to a different base grid. Interpolation points of grids on level ℓ never interpolate from finer grids on level $\ell + 1$ to avoid coupling refinement levels.

Interpolation:

Once a new set of grids is generated, the solution is transferred from the old AMR grid hierarchy to the new one. As a general rule, solution values on the new grid are interpolated from the finest level grid available on the old grid. Although many points on the new grids can be determined as a simple copy from the old grids, in general some care is required during the regridding and interpolation steps to ensure that

accurate values are obtained and that the manipulation of a potentially large number of refinement grids is done in an efficient manner.

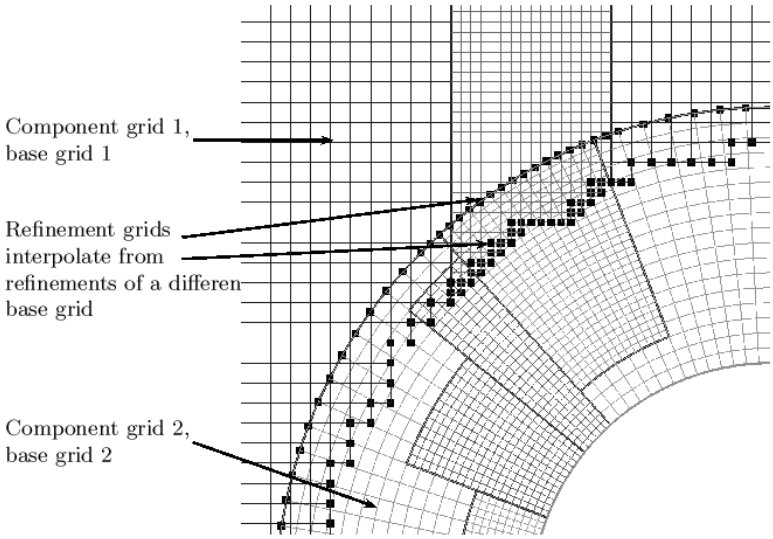


Fig. 5. Overlapping grids and AMR; a view of the overlap region showing the interpolation between refinement grids from different base grids. The black squares indicate interpolation points

3 Testing using the Method of Analytic Solutions

The method of analytic solutions, also known as the method of manufactured solutions[Roa98] or the twilight-zone method[CH90], is a extremely useful approach for testing the accuracy of a numerical implementation of an algorithm. Given a PDE initial-boundary value problem

$$L(u_t, u_x, u_y, \ldots) = F(\mathbf{x}, t)$$

and given any known (usually analytically defined) function $U(\mathbf{x}, t)$, by choosing the forcing function to be

$$F(\mathbf{x}, t) = L(U_t, U_x, U_y, \ldots)$$

it will then follow that U will be an exact solution of the forced PDE. The Overture OGFunction class defines a variety of exact solutions and their derivatives to support the method of analytic solutions. For example, one may define a polynomial, trigonometric polynomial, or *pulse function*

$$U(\mathbf{x},t) = (x^2+2xy+y^2+z^2)(1+\frac{1}{2}t+\frac{1}{3}t^2)$$
$$U(\mathbf{x},t) = \cos(\pi\omega x)\cos(\pi\omega y)\cos(\pi\omega z)\cos(\omega\pi t)$$
$$U(\mathbf{x},t) = a_0\exp(-a_1\|\mathbf{x}-\mathbf{b}(t)\|^{2p}) \ , \ \ \mathbf{b}(t) = \mathbf{c}_0+\mathbf{v}t$$

The polynomial solution is particularly useful since this solution is often an exact solution to the discrete equations on Cartesian grids. The pulse function is good for AMR. Figure (6) shows the solution of a convection diffusion equation using the pulse function as the exact solution. This solution with computed with the amrHype program that is distributed with Overture. Table (1) compares the errors at time $t = 1$ for 3 cases of equivalent grid resolution, (1) using a 124×124 base grid with no AMR, (2) using a 31×31 base grid with two levels of refinement ratio 2, and (3) using a 31×31 base grid and one level of refinement ratio 4. The results in the table show that nearly the same errors are achieved in all three cases.

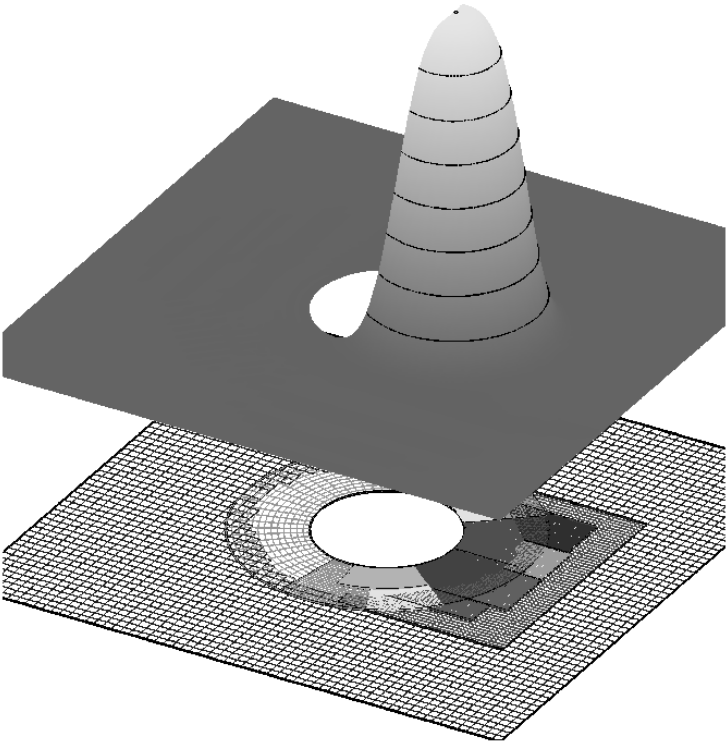


Fig. 6. AMR solution to a convection diffusion equation on an overlapping grid for square region with a circular hole. The exact solution is a Gaussian pulse which translates through the domain.

Table 1. Computed maximum errors at $t = 1.$ for a pulse crossing the circle in a square domain. The solution was computed in three ways designed to have an equivalent effective resolution

	No AMR	2 levels of AMR	1 Level of AMR
	124×124	$31 \times 31 + (r = 2)^2$	$31 \times 31 + (r = 4)^1$
error	$1.98e-02$	$2.07e-02$	$2.07e-02$
grid points	20498	7820	6560
	Effective resolution 124×124		

4 Sample calculations

In this section a few examples will be presented along with some performance statistics. The solutions here were computed with the OverBlown flow solver using a high-order Godunov method[Hen99].

Figure (7) shows the solution of the Euler equations for a shock hitting two airfoils. The base grid consisted of two boundary fitted curvilinear grids and a background Cartesian grid with a total of about $150,000$ grid points. The solution was computed with two additional refinement levels of ratio 4. To compute the same solution without AMR would thus require about $150,000 \times 16^2 = 3.84 \times 10^7$ grid points. At the later stages in the computation the AMR solution required a maximum of about 1 million grid points and 325 grids.

Figure (8) shows the computation of an expanding detonation in a quarter plane. The unsteady Euler equations are solved with a one-step Arrhenius reaction. The overlapping grid consisted of a backgounded Cartesian grid together with an annular grid. In [HS03] this solution was compared to that from a single Cartesian base grid and with a highly resolved one-dimensional computation. Excellent agreement was found between the three different computations. The detonation passed cleanly through the interface between the overlapping grids.

Figure (9) shows the later stages of a computation that illustrates a mechanism of detonation failure/rebirth. An over-driven detonation propogates from left to right into an expanding channel. The detonation temporarily fails in a portion of the expanding region but later reforms when the leading shock hits the lower wall and re-strengthens. For this problem the reactive Euler equations were solved with a three-step chain-branching model. See [HS03] for further details.

Table (2) presents some CPU timings for these two reacting flow simulations. As shown, the overhead due to the use of overlapping grids and adaptive mesh refinement is quite acceptable with a majority of the CPU time spent in the Fortran 77 code that evaluates the Godunov approximation to the reactive Euler equations. The time spent on computing boundary conditions or updating the overlapping grid interpolation points is quite small. The time spent on AMR regridding and interpolation depends on the number of AMR grids required during the calculation. The largest value occurred for the expanding-channel calculation, but this value, 11.6%, is still relatively small. These computations were performed on a Linux desktop with a 2.2 GHz Xeon processor and 2 Gbytes of memory.

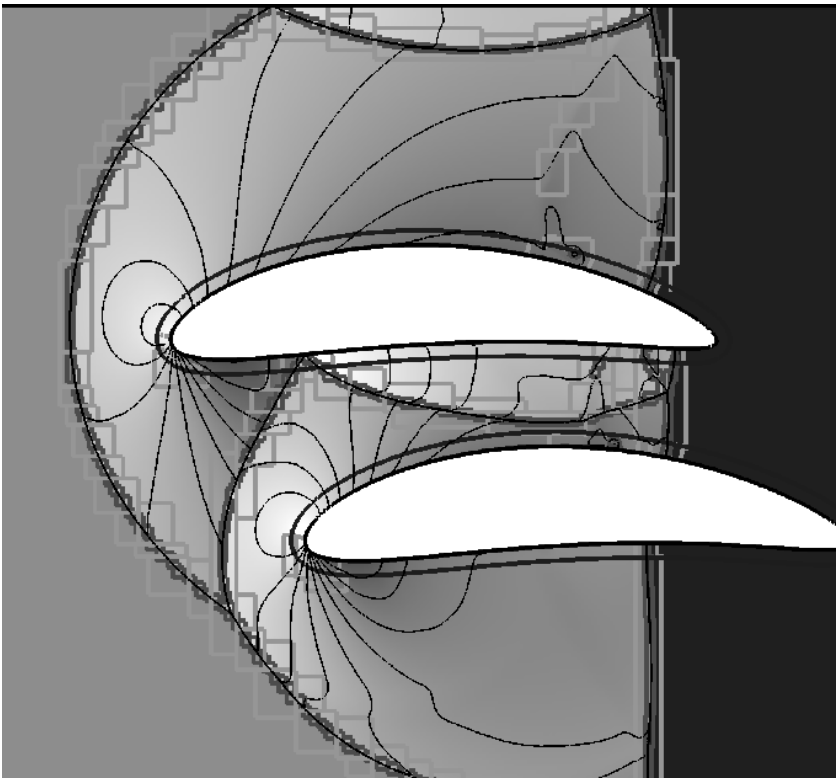


Fig. 7. Shock hitting two airfoils. The boundaries of the grids are shown super-imposed on contours of the density. Two additional refinement levels of refinement ratio 4 were used

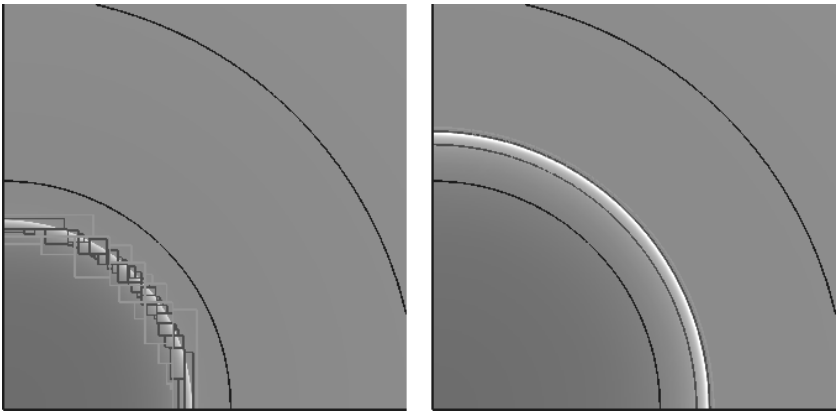


Fig. 8. Detonation in a quarter plane showing the solution and grids as the front propogates through the interfaces between the annular component grid and the rectangular grid. The solution and grids are shown at times 1.7 and 1.8

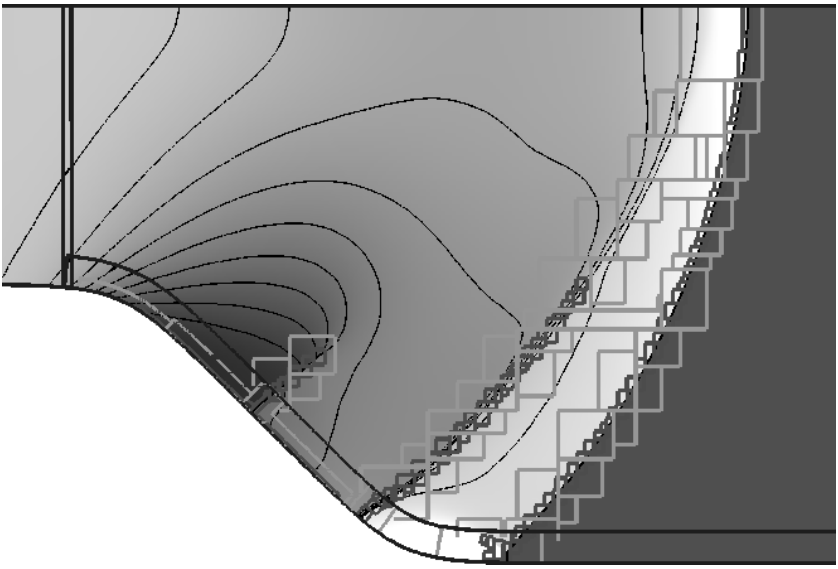


Fig. 9. Detonation propogating through an expanding channel. The density is shown along with the boundaries of the refinement grids. Two additional refinement levels of refinement ratio 4 were used

Table 2. CPU time (in seconds) per step for various parts of the code and their percentage of the total CPU time per step

	Quarter plane		Expanding channel	
	s/step	%	s/step	%
compute $d\mathbf{u}/dt$	13.85	92.7	11.50	82.4
boundary conditions	.12	.8	.14	1.0
interpolation (overlapping)	.09	.6	.45	3.2
AMR regrid/interpolation	.54	3.6	1.62	11.6
other	.34	2.3	.25	1.8
total	14.94	100	13.96	100

Acknowledgements

The work discussed in this paper was performed in collaboration with Professor Don Schwendeman and was supported under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

BBCS95. K.D. Brislawn, D. L. Brown, G.S. Chesshire, and J.S. Saltzman. Adaptively-refined overlapping grids for the numerical solution of hyperbolic systems of conservation

laws. report LA-UR-95-257, Los Alamos National Laboratory, 1995.

BCF+03. David Brown, Kyle Chand, Petri Fast, William Henshaw, Anders Petersson, and Daniel Quinlan. Overture. Technical Report http://www.llnl.gov/CASC/-Overture.html, Lawrence Livermore National Laboratory, 2003.

BHQ99. D. L. Brown, William D. Henshaw, and Daniel J. Quinlan. Overture: An object-oriented framework for solving partial differential equations on overlapping grids. In *Object Oriented Methods for Interoperable Scientific and Engineering Computing*, pages 245–255. SIAM, 1999.

BO84. M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53:484–512, 1984.

BR91. M. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Trans. Systems Man and Cybernet*, 21:1278–1286, 1991.

BT97. E. P. Boden and E. F. Toro. A combined chimera-amr technique for computing hyperbolic pdes. In Djilali, editor, *Proceedings of the Fifth Annual Conference of the CFD Society of Canada*, pages 5.13–5.18, 1997.

CH90. G. Chesshire and W.D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *J. Comp. Phys.*, 90(1):1–64, 1990.

HC87. W.D. Henshaw and G. Chesshire. Multigrid on composite meshes. *SIAM J. Sci. Stat. Comput.*, 8(6):914–923, 1987.

Hen98. W.D. Henshaw. Ogen: An overlapping grid generator for Overture. Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

Hen99. W.D. Henshaw. OverBlown: A fluid flow solver for overlapping grids, user guide. Research Report UCRL-MA-134288, Lawrence Livermore National Laboratory, 1999.

HF91. M. Hinatsu and J.H. Ferziger. Numerical computation of unsteady incompressible flow in complex geometry using a composite multigrid technique. *International Journal for Numerical Methods in Fluids*, 13:971–997, 1991.

HS03. William. D. Henshaw and Donald W. Schwendeman. An adaptive numerical scheme for high-speed reactive flow on overlapping grids. *J. Comp. Phys.*, 191:420–447, 2003.

MB94. R.C. Maple and D.M. Belk. A new approach to domain decomposition, the beggar code. In N.P. Weatherill, editor, *Numerical Grid Genertation in Computational Fluid Dynamics and Related Fields*, pages 305–314. Pineridge Press Limited, 1994.

Mea93. R. Meakin. Moving body overset grid methods for complete aircraft tiltrotor simulations. paper 93-3350, AIAA, 1993.

Mea99. Robert L. Meakin. Composite overset structured grids. In Joe F. Thompson, Bharat K. Soni, and Nigel P. Weatherill, editors, *Handbook of Grid Generation*, chapter 11, pages 1–20. CRC Press, 1999.

OY93. F. Olsson and J. Yström. Some properties of the upper convected Maxwell model for viscoelastic fluid flow. *J. Non-Newtonian Fluid Mech.*, 48:125–145, 1993.

Pet99. N. A. Petersson. Hole-cutting for three-dimensional overlapping grids. *SIAM J. Sci. Comp.*, 21:646–665, 1999.

PSM+93. D.G. Pearce, S.A. Stanley, F.W. Martin, R.J. Gomez, G.J. Le Beau, P.G. Buning, W.M. Chan, T.I. Chui, A. Wulf, and V. Akdag. Development of a large scale chimera grid system for the space shuttle launch vehicle. paper 93-0533, AIAA, 1993.

Roa98. Patrick J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, NM, 1998.

SB87. J. L. Steger and J. A. Benek. On the use of composite grid schemes in computational aerodynamics. *Computer Methods in Applied Mechanics and Engineering*, 64:301–320, 1987.

Sta77a. G. Starius. Composite mesh difference methods for elliptic and boundary value problems. *Numer. Math.*, 28:243–258, 1977.

Sta77b. G. Starius. Constructing orthogonal curvilinear meshes by solving initial value problems. *Numer. Math.*, 28:25–48, 1977.

Sta80. G. Starius. On composite mesh difference methods for hyperbolic differential equations. *Numer. Math.*, 35:241–255, 1980.

TF95. J. Y. Tu and L. Fuchs. Calculation of flows using three-dimensional overlapping grids and multigrid methods. *International Journal for Numerical Methods in Engineering*, 38:259–282, 1995.

.

A Dynamically Adaptive Arbitrary Lagrangian-Eulerian Method for Hydrodynamics

R. W. Anderson, R. B. Pember, and N. S. Elliott

Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA 94513
anderson110@llnl.gov

1 Introduction

The numerical simulation of compressible flows with shocks and material discontinuities is a computational challenge in many important application areas including inertial confinement fusion (ICF), astrophysics, and plasma physics. Lagrangian and ALE techniques have often been favored in the above application areas [3], in part due to the self-adapting nature of Lagrangian grid motion, e.g., contact discontinuities are tracked automatically, and cells are clustered into high density regions behind shocks. However, this inherent form of adaption present in Lagrangian and ALE methods, is less general and robust than a dynamically adaptive method in which the number of cells may change with time, such as the structured grid local adaptive mesh refinement (AMR) methods[1, 5, 4, 11]. The development of a hybrid algorithm combining a Lagrange based ALE with AMR requires the development of modified methods for integration of the mesh hierarchy, new interlevel solution transfer operators, and methods for application of mesh relaxation operators to an AMR mesh hierarchy.

2 Equations of Motion and the Underlying ALE Method

The governing equations of inviscid gasdynamics are discretized from the Lagrangian form:

$$\frac{D\rho}{Dt} + \rho\vec{\nabla}\cdot\vec{V} = 0 \tag{1}$$

$$\rho\frac{D\vec{V}}{Dt} + \vec{\nabla}p = 0 \tag{2}$$

$$\rho\frac{De}{Dt} + p\vec{\nabla}\cdot\vec{V} = 0. \tag{3}$$

where ρ, e, p, and $\vec{V}$ are the fluid density, internal energy, pressure, and velocity respectively, and t is time.

The ALE method employed for integration of the system (1)-(2)-(3) is of the explicit, time-marching, Lagrange plus remap type. The initial Lagrange step follows the general approach taken by Tipton [14]. It employs a predictor-corrector discretization in time, and the HEMP spatial discretization [15, 13]. The scheme employs a monotonic artificial viscosity due to Christensen [6], and a kinematic hourglass filter [10]. The two-dimensional scheme has been described extensively previously; algorithmic details as well as comparisons with more widely known Eulerian methods can be found in a recent work by Pember, et al.[12].

At the end of a Lagrange step, it is often desirable to smooth the grid to prevent excessive mesh distortion which can lead to inaccuracy and often failure of the Lagrangian algorithm. An effective smoothing algorithm can be based upon a Laplace iteration for the transformed coordinates with respect to the Cartesian coordinates of each node. This is the essence of the Winslow method which we take as a representative relaxation operator. For the adaptive method, this must be applied to a mesh hierarchy, which introduces some additional considerations into the AMR hierarchy integration algorithm.

Once the relaxed mesh has been defined, it remains to interpolate the solution from the old Lagrange grid to the relaxed grid. We cast this interpolation in terms of an apparent advection equation. This advection equation is solved using a variant of the Corner Transport Upwind (CTU) scheme [2] for use on a staggered grid. The algorithmic details of the scheme have been discussed in detail in Pember [12], et al. Our implementation uses the SAMRAI C++ AMR library [9, 16, 8].

3 The Lagrangian (L-AMR) Algorithm

We develop first the adaptive components of the Lagrangian algorithm, and then extend the ideas to the ALE context. The essence of the adaptive Lagrangian method is the introduction of new interlevel solution transfer operators. Interlevel transfer operators are required when new grids are created, for the generation of pseudo-boundary conditions on finer levels in the hierarchy, for synchronizing coarse and fine data in the hierarchy, and upon the removal of refined grids. The hierarchy advance for the Lagrangian algorithm requires no fundamental modification. However, care must be taken in applying coarse-fine boundary conditions on the moving mesh. On the fine mesh, the nodes coincident with the coarse mesh are slaved to the coarse node motion by linearly interpolation in time, and the remaining "hanging nodes" in multi-dimensions are slaved by linearly interpolating first in time, and then in space. The remaining ghost data are interpolated using the refinement operators to be described.

3.1 Staggered Mesh Refinement

The operators developed here are designed with the following properties in mind:

P1) Constant field preservation
P2) 2nd order accuracy (in smooth regions)
P3) Monotonicity
P4) Local conservation
P5) Exact inversion of refinement by coarsening

A simple way to ensure that P4 and P5 are simultaneously achieved is to maintain an exclusive r:1 correspondence, r being the refinement ratio, from fine nodes to coarse nodes, such that the local interpolation stencils on the fine mesh do not overlap. In this case inverting a locally conservative interpolation is simply a matter of summing the fine values of the conserved quantity in the stencil. This leads to a choice of odd refinement ratios to ensure this property for both cell- and node-centered quantities.

Consider a one-dimensional interpolation of some scalar density function ϕ with a known slope ϕ_0' and average value ϕ_0 over some interval Δx_0, into N arbitrary subintervals $\Delta x_k = x_{k+1} - x_k$.

An interpolation in which values are taken from the centers of the subintervals

$$\phi_k = \phi_0 + \phi_0' \left(\bar{x}_k - \frac{1}{2}\Delta x_0 \right)$$

where $\bar{x}_k = (x_k + x_{k+1})/2$, is locally conservative of $\phi\Delta x$ in the sense that

$$\sum_{k=1}^{N} \phi_k \Delta x_k = \phi_0 \Delta x_0,$$

since

$$\sum_{k=1}^{N} \phi_k \Delta x_k = \phi_0 \Delta x_0 + \phi_0' \left(\sum_{k=1}^{N} \bar{x}_k \Delta x_k - \frac{1}{2}\Delta x_0^2 \right). \tag{4}$$

In a constant field, all slopes ϕ_0' are zero, and constant fields are preserved independently of the mesh. We now have a general one-dimensional expression for interpolation that satisfies P1, P2, and P5. In order to address P3, we employ the well-known van Leer limiter for slope determination.

If we desire to prevent oscillations in the primitive variables $\phi = (\rho, u, v, E)$, where E is the total energy, the required interpolation basis to obtain property P4, local conservation, is $x = (V, \tilde{m}, \tilde{m}, m)$, where V is volume, $\tilde{m}$ is nodal mass, and m is cell mass. This interpolation generalizes to multi-dimensions by including terms for the slopes in each logical direction.

Upon closer examination of (4), there is a consistency condition for local conservation that requires that the basis itself be locally conserved, i.e.,

$$\sum_{k=1}^{N} \Delta x_k = \Delta x_0. \tag{5}$$

We have identified two potential difficulties in achieving the consistency condition and hence a conservative operator. The first is calculation of hexahedral volumes

in three dimensions. Many volume formulas for hexahedra are based on a surface triangulation of faces, which will not be consistent with a multi-linear interpolation of the mesh in the sense of (5). If instead one employs a bilinear surface model for the cell volume, then the interpolation is indeed consistent and mass conservation is retained. An efficient implementation of this exact integration is given by Dukowicz [7].

However, if we employ a cell mass interpolation, (5) is violated in the case of a nodal mass basis for velocity interpolation. In this context the neighbor-average definition of nodal mass is inconsistent with interpolation based on linear slopes of density. In multi-dimensions on a general grid, the neighbor-average definition is inconsistent even with constant slopes of density. One way to reconcile this difficulty is by interpolating an adjusted coarse velocity field

$$u_0' \sum \tilde{m}_i = u_0 \tilde{m}_0 + \sum u^* \delta m \tag{6}$$

where, borrowing from remap terminology, u^* is an "edge state" and δm is a "transport mass." If the transport masses are constructed such that

$$\sum \tilde{m}_i = \tilde{m}_0 + \sum \delta m,$$

and edge states are constructed such that all $u^* = u_0$ in a constant coarse grid velocity field, then this adjusted field both conserves momentum and preserves a constant field upon interpolation to the fine mesh. This adjustment procedure has a natural interpretation as a nodal flux of momentum due to the implied mass transport across the median mesh by the imposition of cell density slopes in the mass interpolation step. We find that an arithmetic average for the edge state is sufficient for a variety of test cases, although one can apply upwinding based on the sign of the mass flux as an alternative. A minor concession with this procedure is condition P5, a loss of precise invertibility.

3.2 Coarsening

There are two natural choices for coarsening operators as weighted sums of the conserved quantities, i.e.,

$$\rho_0 = \frac{\sum \rho_i V_i}{\sum V_i} \text{ or } \frac{\sum \rho_i V_i}{V_0}$$
$$u_0 = \frac{\sum u_i \tilde{m}_i}{\sum \tilde{m}_i} \text{ or } \frac{\sum u_i \tilde{m}_i}{\tilde{m}_0}$$
$$E_0 = \frac{\sum E_i m_i}{\sum m_i} \text{ or } \frac{\sum E_i m_i}{m_0}$$

where i varies over the refinement stencil corresponding to each coarse node. The coarse mesh is formed by selection of every r'th mesh point. The first choice is a constant field preserving construction and the second choice is conservative, but not vice versa, in general. In order to achieve a simultaneously constant field preserving

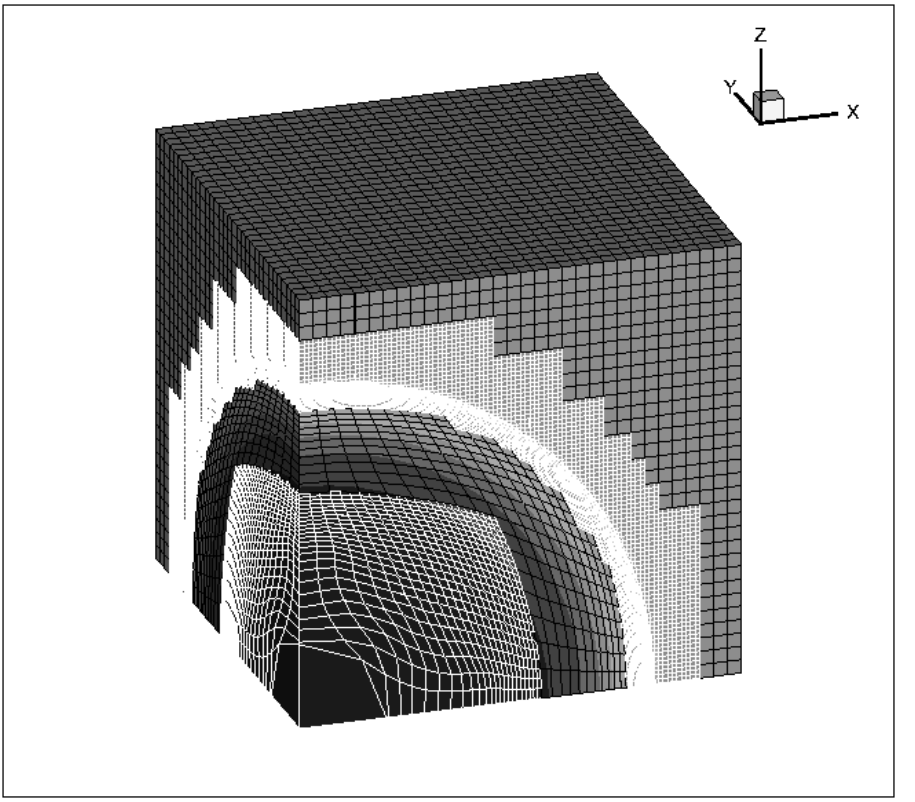


Fig. 1. L-AMR solution of Taylor-Sedov blast wave problem. Coarse level grids shown in black, fine level grids in white. Colormap is density field.

and conservative operator, one must apply a remap operation to (a copy of) the fine grid data, remapping from the fine Lagrange grid to one which is fully aligned with the underlying coarse grid. This is an analagous step to (6), and indeed preceeded and inspired the former.

A demonstration of the three-dimensional L-AMR algorithm for the Taylor-Sedov blast wave is shown in Figure 1. This solution was computed using two mesh refinement levels. The coarsest level is shown in black and the fine level in white. The refined regions capture the outgoing shock as well as the region of strong expansion near the origin. The speedup for this case over a grid using fine mesh everywhere is about 11.

4 The Arbitrary Lagrangian-Eulerian (ALE-AMR) Algorithm

The introduction of a Winslow-type relaxation operator introduces some additional requirements for the ALE-AMR method. The relevant feature of the equipotential

type methods is that they are derived from elliptic equations which intrinsically exhibit globally coupled solutions. The consequence for an AMR mesh hierarchy is that coarse meshes may not be relaxed independently of finer meshes; they instead require a solution method which enforces the required coupling between mesh levels. Thus one cannot simply use a composite Lagrange plus remap operator directly in the previously described L-AMR method to arrive at a well-behaved ALE-AMR method.

Instead the hierarchy advance algorithm is modified to include relaxation iterations *only when all finer levels have advanced to a given simulation time*. We define a "level grid" g_θ as the set of all nodes on L_θ, and a "composite grid" g^c_θ as

$$g^c_\theta = g_\theta \setminus I(g_{\theta+1}) \cup g^c_{\theta+1}. \tag{7}$$

where I is a restriction operator of nodal injection, and the definition $g^{\,c}_{\theta_{max}} = g_{\theta_{max}}$ closes the recursion. An ALE-AMR hierarchy advance algorithm is then

repeat
  construct interpolated $\theta - 1$ and/or domain boundary conditions
  advance L_θ Lagrangian to $t = \min(t + \Delta t, t_{\theta-1})$
  if $\theta < \theta_{max}$ **then**
    recurse with $\theta = \theta + 1$
    repeat
      relax g^c_θ
    until g^c_θ is sufficiently smooth
    remap levels θ to θ_{max}
    synchronize levels θ through θ_{max}
    optionally regrid levels $\theta + 1$ to θ_{max}
  end if
until $t = t_{\theta-1}$

The logical diagram in Fig. 2 visualizes the process for a 3-level hierarchy.

5 Numerical Example

The utility of the algorithm is demonstrated on a three-dimensional Richtmyer-Meshkov instability in a converging geometry; this serves as a proof-of-principle calculation for inertial confinement fusion (ICF) applications. The initial condition consists of an incoming spherical shock impinging upon a high density spherical shell with a single mode perturbation on its outer surface. For efficiency, the domain is constrained by cutting planes to approximately three wavelengths, as shown in Figure 3. As the shock impinges on the high density fluid, the classical bubbles and spikes instability occurs. Later times show the growth of complex flow features, as shown in Figure 4. ALE-AMR enables much greater resolution on the detailed features of the instability than a comparable ALE method, and the mapped grid capabil-

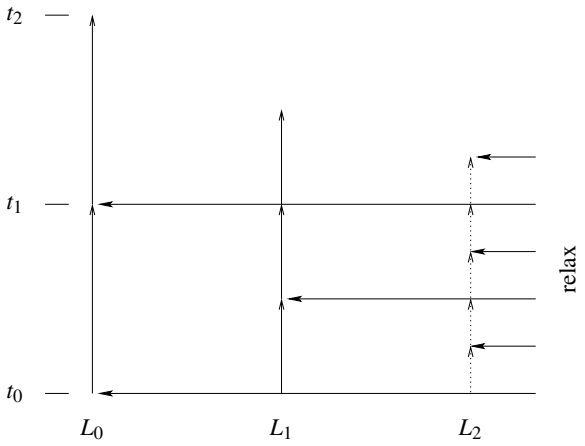


Fig. 2. ALE-AMR hierarchy integration logical diagram, shown for a 3-level hierarchy. Vertical lines are Lagrange steps, horizontal lines are relaxation operations.

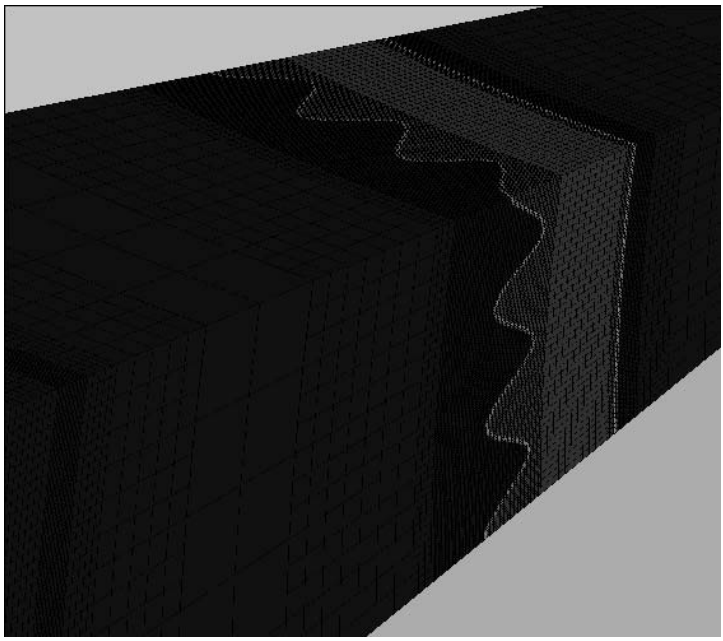


Fig. 3. AMR grid hierarchy for initial condition of Richtmyer-Meshkov instability calculation. Four levels of refinement. Colormap of density.

ity greatly reduces the "mesh imprinting" errors that tend to feed spurious anisotropic instability modes in this type of calculation on Cartesians grids.

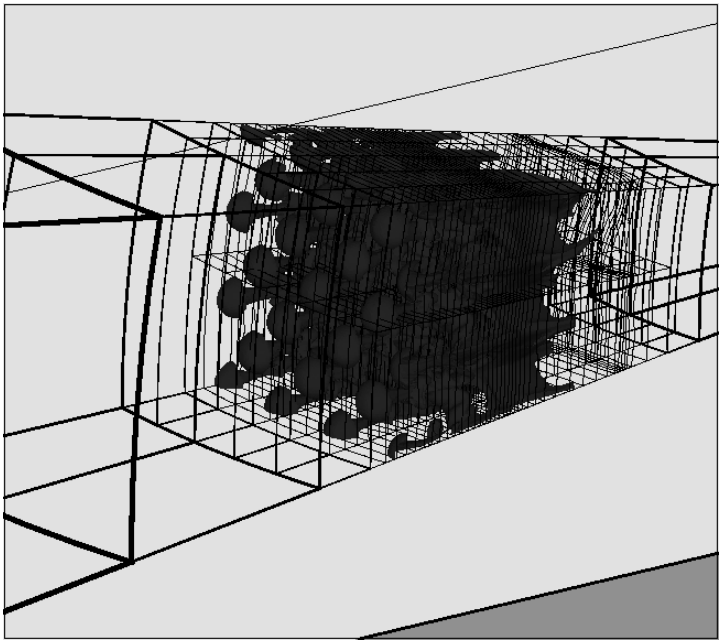


Fig. 4. Later time instability growth. Isosurface of density. Box outlines shown in light to heavy outlines, from finest to coarsest level.

6 Conclusion

The hybridization of staggered grid ALE and AMR on structured meshes is accomplished with the development of new interlevel transfer operators, modifications to the hierarchy advance algorithm, and methods for applying elliptic relaxation operators to a time- and space-refined mesh hierarchy. The advantageous features of both types of methods are retained, and the result is a powerful combination for the class of applications for which ALE methods are well-suited.

7 Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

1. J.B Bell, M.J. Berger, J.S. Saltzman, and M. Welcome. Three dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15:127–138, 1994.

2. J.B. Bell, P. Colella, J.A. Trangenstein, and M. Welcome. Adaptive mesh refinement on moving quadrilateral grids. In *Proceedings, AIAA 9th Computational Fluid Dynamics Conference*. Buffalo, New York, June 14-16, 1989, p.471-579.
3. B. J. Benson. An efficient, accurate, simple ALE method for nonlinear finite element programs. *Comp. Meth. Appl. Mech. Eng.*, 72:205–350, 1989.
4. M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.
5. M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.
6. R.B. Christensen. Godunov methods on a staggered mesh — An improved artificial viscosity. Technical Report UCRL-JC-105269, Lawrence Livermore National Laboratory, 1990.
7. J.K. Dukowicz. Efficient volume computation for three-dimensional hexahedral cells. *Journal of Computational Physics*, 74(2):493–496, February 1988.
8. Richard Hornung, Noah Elliott, Steve Smith, Andy Wissink, Brian Gunney, and David Hysom. SAMRAI home page. http://www.llnl.gov/CASC/SAMRAI, 2003.
9. Richard Hornung and Scott Kohn. Managing application complexity in the samrai object-oriented framework. *Concurrency and Computation: Practice and Experience*, 14:347–368, 2002. Also LLNL technical report UCRL–JC–141749.
10. L.G. Margolin and J.J. Pyun. A method for treating hourglass patterns. Technical Report LA-UR-87-439, Los Alamos National Laboratory, 1987.
11. R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutchfield, and M. L. Welcome. An adaptive Cartesian grid method for unsteady compressible flow in complex geometries. *J. Comput. Phys.*, 120:278–304, 1995.
12. Richard Pember and Robert Anderson. Comparison of direct eulerian godunov and lagrange plus remap artificial viscosity schemes for compressible flow. Technical Report AIAA Paper 2001-2644, 2001.
13. R.W. Sharp. HEMP advection model. Technical Report UCID-17809, Lawrence Livermore National Laboratory, 1978.
14. R.E. Tipton. Unpublished report, Lawrence Livermore National Laboratory, 1990.
15. M. L. Wilkins. Calculation of elastic-plastic flow. *Meth. Comp. Phys.*, 3:211–263, 1964.
16. Andrew Wissink, David Hysom, and Richard Hornung. Enhancing scalability of parallel structured amr calculations. In *Proceedings of the 17th Annual ACM International Conference on Supercomputing (ICS03)*, San Francisco, CA, June 2003.

Front Tracking Algorithm Using Adaptively Refined Meshes

Zhiliang Xu[1], James Glimm[12], and Xiaolin Li[1]

[1] Department of Applied Mathematics and Statistics, University at Stony Brook
xuzhi@ams.sunysb.edu, linli@ams.sunysb.edu
[2] Center for Data Intensive Computing, Brookhaven National Laboratory
glimm@ams.sunysb.edu

Summary. We propose a new algorithm which combines the front tracking with an adaptively refined Cartesian grid for solving systems of nonlinear conservation laws.

1 Introduction

The numerical simulation of time dependent complex flows which contain linear and nonlinear discontinuities is a challenge. To maintain the quality of the computed solution and to capture the solution structures of different scales, it is desirable to use adaptive grid refinement (AMR).

The AMR strategy is to expend the most computation effort in the region where it is most necessary by using more grid points here. Block-structured Cartesian grids for the refinement of complicated geometric configurations were proposed [BO84, BJ85]. The study of block-structured AMR for hydrodynamics has been developed systematically by M. Berger and P. Collela [Ber87, BC89].

The front tracking method is also an adaptive method. A front represents either a contact discontinuity, a shock wave or a rarefaction wave edge of the flow. The front represented by a lower dimensional manifold is embedded in the spatial grid. The geometry of a front is represented by a curve composed of piecewise linear segments in 2D, and a surface composed of a set of connected triangles in 3D. Driven by the flow dynamics, the front follows the flow motion and moves freely across the underlying spatial grids to track the desired flow features. Interfacial numerical diffusion across the fronts, which is a common phenomenon for the capturing method, is prevented by the tracking method. The tracking method also reduces post-shock oscillations. Two sub-processes: the front propagation and the updating of the solution on the spatial grids, complete a single computational time step. The front tracking method has been used to simulate complex fluid mixing geometries in 2D [GGMM88, CGSZ96], and in 3D [LG96, GGL98].

This paper describes a combination of these two methods. We have merged the Front Tracking code FronTier with the AMR code Overture which is based on the

Berger-Collela cell based refinement algorithm, [Ber87, BC89] and developed and maintained at LLNL.

2 The 2D AMR Front Tracking Algorithm

We describe the modules of the AMR front tracking algorithm according to the order of their execution in this section. We use block-structured uniform Cartesian grids. We use Overture to generate the AMR grid hierarchy. FronTier is responsible for updating the numerical solutions on each patch.

2.1 Initialization of the AMR grids

For an initial regular, coarse mesh filled with data by the FronTier initializer, the cells in this mesh where the estimated error is larger than some threshold are determined to be tagged. The Berger-Collela cell based refinement algorithm then generates refinement grids from these tagged cells by organizing them into a set of rectangular patches. The patches are aligned with the original mesh. The patches are properly nested, so that successive refinement levels are allowed, with the restriction that adjacent refinement levels will differ by one level only, usually representing refinement by a factor of 2 or 4 in each dimension. The fine patches are recursively added to the next coarse level patches until the desired solution accuracy is obtained. The patches with the same grid spacing belong to the same grid level. The refinement grids are organized in a hierarchy. The coarsest grid which discretizes the whole domain belongs to level 0, the next added refinement grids belong to level 1 and so on.

The error estimation provides the basis for the AMR strategy. The Overture implementation of the error estimate is based on the evaluation of a weighted sum of the first and second derivatives of one or several state variables. The estimator generally achieves a maximum value at discontinuities. In this way, the grid adaptation thus tends to refine this region, which is exactly what we desire.

We address the tracking of the contact discontinuity. We assure that the tracked front is completely embedded inside the finest grid. To ensure there are a sufficient number of the fine grid cells to cover the contact discontinuity, we use a controlling parameter which specifies the minimum distance from the front to the fine grid boundary. Cells with a distance to the front smaller than this parameter are always tagged during the refinement. Thus the finest level patches always cover a minimum neighborhood of the tracked front.

To implement the above described module, we have built a data translation interface to translate the FronTier initial state data to the Overture error estimation class and to translate the refinement grid hierarchy from Overture to FronTier.

Since the numerical computation step carried by FronTier is organized into front propagation and interior state update sub-processes, for each AMR patch, we construct a front structure and a wave structure. FronTier then calls the front propagation solver and interior finite difference solver to update the numerical solutions.

2.2 The Single Time Step Tracking Algorithm

To support dynamic tracking, a front propagation driver associated with the underlying physics and a set of geometric manipulation routines are provided [GIM81, GGL98, GGL99a]. Basic operations such as the creation and the re-meshing of the front and the untangling of self-intersecting fronts are managed by geometric manipulation routines.

The front is oriented. Each front point has two states defined, one on each side. The front propagation is accomplished by propagating each point on the front. To propagate the front point, we define the normal and tangential direction on the front. We then solve

$$U_t + n \cdot ((n \cdot \nabla)F(U)) + \tau \cdot ((\tau \cdot \nabla)F(U)) = 0, \tag{1}$$

where $n = (n_1, n_2)$ is the normal, $\tau = (-n_2, n_1)$ is the corresponding tangent. Eq. (1) is solved by dimensional splitting. Given the two side states U^n at the time t_n, we first solve

$$U_t + n \cdot ((n \cdot \nabla)F(U)) = 0, \text{ with } U(t = t_n) = U^n. \tag{2}$$

The time evolution defines a generalized Riemann problem as introduced in [CGMP86]. From the solution of this problem, we obtain the new location of the front point and the two new states on each side of the front. We denote these two states as U^{cn}. Then we solve

$$U_t + \tau \cdot ((\tau \cdot \nabla)F(U)) = 0, \text{ with } U(t = t_n) = U^{cn}, \tag{3}$$

with any finite difference solver on each side. The point propagation is complete. Since the front contained in the finest grid level is most accurate, we only maintain the front in this level. The geometric operation routines are called after point propagation to re-mesh or untangle the front if there is the need. The front in a grid communicates with neighboring grid fronts to maintain its integrity.

The last step of a single tracking step is to update the states on the spatial grids. See [CGMP86, GGL98, GGL99, GGL99a, GGL00] for a detailed discussion of the front propagation and the interior sweep.

Except for the physical boundaries, the boundary conditions on the refinement grids are imposed by use of ghost cells, which extend the patch cells by a number related to the stencil size of the finite difference scheme in each direction. Using these ghost cell state values in the usual way, the equations can be integrated in the given patch with sufficient boundary data to fill all finite difference stencils. After the updating of spatial grid solutions on the patches, solutions defined on the coarse grid cells which are covered by a finer grid and the solutions defined on the cells which are on a coarse-fine grid boundary but are not covered by any fine grid are subject to modification. At the coarse-fine grid boundaries, the refluxing procedure must be applied to ensure conservation and to reduce errors. The cell states of a fine level patch are averaged to the next coarse level patch in which this fine level patch is nested. With the front present, averaging of states from two sides of the front is not allowed.

The adaptation in the temporal dimension is not implemented at the present time. At the end of the single time step, all patches are synchronized with the same discretized time step. The discrete time increment Δt is the one calculated from the finest level grids.

2.3 Regridding the AMR grids

The dynamic evolution of the solutions requires regeneration of the refinement grid hierarchy after a specified number of time steps to capture the most significant features of the solution. As with the initialization step, FronTier gathers the states on the spatial grids, sends them to Overture, and Overture performs the regridding step according to the information from the states obtained.

The solution states defined on the old refinement grid hierarchy are interpolated to the new refinement grid hierarchy. The fronts in the old fine grid sets are assembled. Then for each of the new refinement grids, a proper portion of the assembled front is cut if it is present inside the grid. The solution is then advanced on the regridded grid hierarchy.

2.4 Parallel computation

FronTier is a fully parallelized software package running on distributed-memory systems while the current Overture implementation is not. Based on the parallelization of FronTier, we also implemented a parallel AMR tracking algorithm. The current strategy is the following: A global domain is divided into rectangular subdomains in FronTier. Each subdomain is assigned to a single processor. In each subdomain, we perform the adaptive refinement. Because the Overture error estimator does not communicate through subdomain boundaries, there might not be enough refinement across the subdomain boundaries. To ensure sufficient refinement, for two adjacent subdomains, assume one has a finer refinement (say level l) patch than the other. The subdomain with less refinement also creates a level l patch with specified depth (usually 4 to 6 cells) and length to match the refined level l patch of its neighboring subdomain. In this way, the refinement is not influenced by the subdomain decomposition.

Since each subdomain has different refinement grids, the computation load is unbalanced. To balance the load on different processors, we need to distribute AMR patches across processors, namely, the processors with excessive load will send some of the patches to other processors with deficient load.

To distribute the unbalanced workload, we need to send the entire states of a patch across the processors. If two nested fine and coarse patches of two successive levels are assigned to different processors, we need to send the entire solution states of the fine patch to the coarse patch at the averaging step of the single time step. Besides these parallel communications, there are other communications associated with front tracking at a fixed grid level. The ghost cell states are filled at the end of the single time step. Except for the ghost cells on the reflected or periodic boundaries, the ghost cell portion of a given patch covers the interior of adjacent patches

at the same level or the next coarse level in which this patch is nested. If the ghost cell of a patch overlaps a cell from the interior of another same level patch, the state of this ghost cell is obtained by copying states from this overlapped cell. Otherwise, the ghost cell state is interpolated from the next coarse level patch in which the ghost cell is located. If these patches are scattered to different processors, we send these copied or interpolated states through the parallel communication. We also communicate fine-coarse boundary fluxes if the corresponding fine and coarse patches are computed in different processors. In the case of Front Tracking, it is also necessary to communicate ghost cell portions of the front. At the end of the normal and tangential front propagation and the re-meshing or the untangling of the front, the ghost cell portion of the front of a given patch is replaced by the front cut from the interior of the neighboring patches. The cut pieces of the front from the neighboring patches must be sent to this given patch either by copying, if these patches are in the same processor or by the parallel communication if these patches are in different processors.

For parallel communication, we construct a table. Each patch corresponds to an item in the table. Each item of this table records a patch id, the processor id in which this patch is computed, ids of all patches neighboring this patch at the same level, the id of the next coarse level patch embedding this given patch and the ids of the next fine level patches embedded in this patch. Each processor maintains a copy of this table. The parallel communication is determined from this table.

We are experimenting with the optimization of load balancing, which must be reconsidered in the context of front tracking, as the computational load is nonuniform, depending on the number of front points as well as the number of cells in a given patch. We use the algorithm developed by W. Crutchfield *et al* [CLB99] to determine the load distribution. This algorithm assumes the communication cost is ignorable. In our case, however, we have large and irregular communications which can not be ignored. To develop a better load balancing strategy is still a research issue.

3 Numerical Examples

We present two numerical examples illustrating our algorithm.

3.1 A spherical Richtmyer-Meshkov instability

We calculate an axisymmetric spherical RM instability. A Mach 10 imploding spherical shock strikes a spherical contact interface perturbed by 6 modes. We used 3 level refinements, with the refinement ratio of 2 and the level 0 grid of 100×100 cells on the 1×1 domain. We compare the result with the uniform grid (which is comparable to the finest level of the AMR grid) simulation. The AMR solution matches the uniform grid solution as shown by Fig. 1.

Fig. 1. Density and grid plots for the spherical RM simulation at time = 0.3. The left frame shows the density plot from AMR front tracking, and the middle frame shows the density plot from uniform grid front tracking. The right frame shows the AMR grid plots.

3.2 A simulation of spray formation in a diesel jet

The wide disparity of length scales in the diesel jet simulation problem (injection nozzle width 0.178mm, mean droplet size 10 micron, jet length during simulation 5cm) calls for AMR. In Fig. 2 we illustrate this capability. We used 5 processors, 3 level refinements with the refinement ratio of 2 and a level 0 grid of 62×250 cells on the $0.248cm \times 1cm$ domain. The left part of the frame shows the density for the simulation, and the right part of the frame shows the grid in a color plot at time 0.06, which is 60 percent of the pressure rising time. Each color in the grid plots represents a processor. The patches in the same subdomain are computed in different processors determined by the load balancing. The comparison with experiment data from Argonne National Laboratory and uniform grid simulations is conducted at Brookhaven National Laboratory.

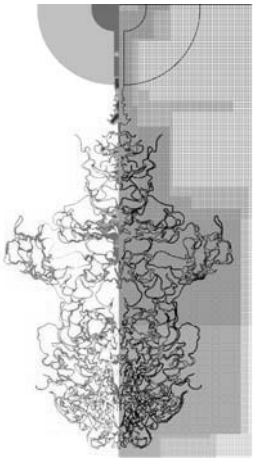


Fig. 2. Density plot of the jet simulation at t = 0.06

4 Conclusions and discussion

We have presented an AMR front tracking algorithm for calculation of time-depend flows. With this AMR front tracking, we have resolved the geometry features of various sizes and scales and maintained a sharp interface. This 2D methodology is also applicable to 3 dimensions, which is now under the development. The Berger-Collela cell based refinement algorithm gives unequal-sized grids. Dynamic load balancing is an important issue in the parallelization. The optimization of load balancing is still being investigated.

Acknowledgments. We would like to thank William B. Henshaw and Petri Fast at LLNL for helpful comments and suggestions during the code development.

References

BO84. M. Berger, J. Oliger: Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. J. Comp. Phys., **53**, 484–512 (1984)

BJ85. M. Berger, T. Jameson: Automatic Adaptive Grid Refinement for the Euler Equations. AIAA, **23**, 561–568 (1985)

Ber87. M. Berger: Adaptive Finite Difference Methods in Fluid Dynamics. In: von Karman Lecture Notes on CFD NYU/DOE report 03077-277 (1987)

BC89. M. Berger and P. Colella: Local Adaptive Mesh Refinement for Shock Hydrodynamics. J. Comp. Phys., **82**, 64–84 (1989)

GGMM88. C. L. Gardner, J. Glimm, O. McBryan, R. Menikoff, D. Sharp, and Q. Zhang: The Dynamics of Bubble Growth for Rayleigh-Taylor Unstable Interfaces. Phys. Fluids, **31**, 447–465 (1988)

CGSZ96. Y. Chen, J. Glimm, D. H. Sharp, and Q. Zhang: A Two-Phase Flow Model of the Rayleigh-Taylor Mixing Zone Phys. Fluids, **8**, 816–825 (1996)

LG96. X.-L. Li, and J. Glimm: A Numerical Study of Richtmyer-Meshkov Instability in Three Dimensions In: Proceedings of the Second Asia CFD Conference, Tokyo (1996)

GGL98. J. Glimm, J. W. Grove, X.-L. Li, K.-M. Shyue, Q. Zhang, Y. Zeng: Three Dimensional Front Tracking SIAM J. Sci. Comp., **19**, 703–727 (1998)

GIM81. J. Glimm, E. Isaacson, D. Marchesin, and O. McBryan: Front Tracking for Hyperbolic Systems. Adv. Appl. Math., **2**, 91–119 (1981)

GGL99a. J. Glimm, J. W. Grove, X.-L. Li, and D. C. Tan: Robust Computational Algorithms for Dynamic Interface Tracking in Three Dimensions. SIAM J. Sci. Comp., **21**, 2240–2256 (2000)

CGMP86. I-L. Chern, J. Glimm, O. McBryan, and B. Plohr: Front Tracking for Gas Dynamics J. Comp. Phys., **62**, 83–110 (1986)

GGL99. J. Glimm, J. W. Grove, X.-L. Li, and N. Zhao: Simple Front Tracking. In: Contemporary Mathematics, Amer. Math. Soc (1999)

GGL00. J. Glimm, J. W. Grove, X.-L. Li, W. Oh, and H. Sharp: A Critical Analysis of Rayleigh-Taylor Growth Rates. J. Comp. Phys., **169**, 652–677 (2001)

CLB99. William Crutchfield, Vincent E. Beckner, Mike Lijewski, and Charles A. Rendleman, : Parallelization of Structured, Hierarchical Adaptive Mesh Refinement Algorithms. (1999)

An accuracy study of mesh refinement on mapped grids

D. Calhoun and R. J. LeVeque

University of Washington Dept. of Applied Mathematics Seattle, WA, 98195

Summary. We test a high-resolution wave-propagation algorithm for hyperbolic conservation laws on mapped quadrilateral and hexahedral grids in the context of adaptive mesh refinement. We discuss some of the issues related to using non-Cartesian grids with AMR and study a test problem in which a grid refinement interface is fixed in space on a highly skewed portion of a mapped grid. Smooth and shock-wave solutions to the Euler equations are used to investigate the possibility that spurious reflections or other numerical errors might be generated at a grid interface.

Key words: gas dynamics, finite-volume, finite-difference, Cartesian grid, mapped grids, computational fluid dynamics, adaptive mesh refinement

1 Introduction

We study one approach to solving partial differential equations on a mapped grid in the context of AMR, using the finite volume wave-propagation algorithm described in [8] and [9] in two dimensions and in [6] in three dimensions. The application of this method to quadrilateral grids in two dimensions is discussed in [9] and recently we have implemented an extension to hexahedral grids in three dimensions. A logically rectangular computational grid is mapped to physical space via a mapping function that is applied to each corner of a grid cell. In two dimensions these corners can be connected by straight lines to obtain quadrilateral cells. In three dimensions the four corners on each face will typically not be co-planar but can be connected by ruled surfaces to obtain hexahedral physical cells. The wave-propagation algorithm is based on solving Riemann problems normal to the cell faces to obtain propagating waves. Limiters are applied to these waves and the limited waves used in "second-order" correction terms to obtain a high-resolution method.

These algorithms have been implemented with AMR in the style of Berger-Oligier and Berger-Collela [5, 2, 3] in the AMRCLAW software package as part of CLAWPACK [7, 4]. When this algorithm is applied to mapped grids, we refine grid

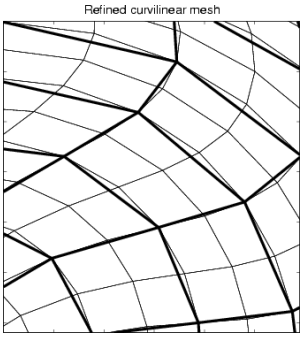


Fig. 1. Refined mapped grid shows the potential misalignment between coarse and fine grid cells. The coarse mesh is represented by thick lines and corresponding fine mesh is shown in thin lines.

cells by relying on an underlying smooth mapping function that tells us how to subdivide a coarse mesh cell. That is, the "midpoint" of two neighboring corners of a coarse cell is defined as the point along the curve, defined by our smooth mapping, passing through those two points. This eliminates the need to construct a smooth mapping through coarse grid points, as is done, for example, in the method developed in [1, 10]. Our strategy has the advantage of simplicity, particularly when extending the method to three dimensions. The disadvantage of this strategy, however, is that underlying refined cells are not necessarily contained in their corresponding coarse cell. See Figure 1 for an illustration of this. This is a potential source of error and part of our goal here is to investigate whether this simpler approach is sufficient or generates unacceptable errors.

Since refined cells are not necessarily contained within their coarse cell, one might suspect that volumes are not preserved and interpolation between coarse and fine grids might not be conservative. We address this issue by using the capacity function formulation of the equations as discussed in [8], [9]. The capacity κ_{ij} of a grid cell plays the role of a discrete Jacobian and is the ratio of the physical volume of a mesh cell to the volume in computational space. We use this in our fine to coarse interpolation. For example, in two-dimensions, a coarse value Q_{ij} is determined from its four fine grid values Q_{ij}^k, $k = 1,..4$ using the formula

$$Q_{ij} = \frac{\sum_{k=1}^{4} \kappa_{ij}^k Q_{ij}^k}{4\kappa_{ij}}. \tag{1}$$

This formula will assure that the conserved quantities are properly conserved between coarse and fine grids in spite of the fact that refined cells are not exactly aligned with their coarse grid cells. We may lose accuracy, however, unless

$$\kappa_{ij} = \frac{1}{4} \sum_{k=1}^{4} \kappa_{ij}^k. \tag{2}$$

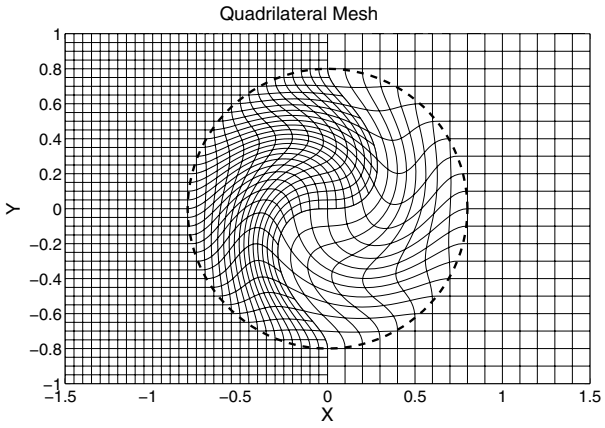


Fig. 2. Mapped partially refined grid used for two dimensional numerical studies. This plot shows every fourth grid line on a mesh with 120 coarse grid points in the x-direction. The circular region encloses the mapped area.

This is satisfied with the refinement strategy of [1, 10], but is not satisfied with our approach. Consequently, constant states are not exactly preserved between coarse and fine grids. We show that results look promising even though (2) is not strictly satisfied.

At coarse-fine boundaries, we use the conservation fix-up described in [5]. In this fix-up, we must solve Riemann problems between ghost cells of the fine grid, and the coarse cells that border the fine grid. On a quadrilateral or hexahedral mesh in which the fine grid cell edges do not line up with coarse cell edges, we use the coarse grid cell edges to determine the rotation angle necessary to define the Riemann problem required in the fix-up step.

2 Numerical results

Our main goal is to provide insight into the accuracy that can be expected when using the wave-propagation algorithm on a mapped Cartesian grid in the presence of a grid refinement interface. Normally in AMR the fine grids are constantly adapted to follow strong solution features such as shock waves. However, in complicated flow problems there are other nontrivial features such as smooth profiles or weaker shocks that are not always covered by refinement but rather move across the fine-coarse interface. To study the effect of such an interface on the solution we use a test problem in which the region of refinement is *fixed* and a solution feature moves across the interface. In order to obtain quantitative comparisons we choose a simple situation in which the true solution is a one-dimensional plane wave.

We show results for test problems on a quadrilateral grid of the type shown in Figure 2 and the hexahedral generalization shown in Figure 8. In each case the grid

is Cartesian near the edges of the domain, but a circular region of radius $R = 0.8$ about the origin is rotated by the smooth grid mapping that in two dimensions takes the form

$$\begin{aligned} X(\xi,\eta) &= \cos(\alpha(r)\theta)\xi + \sin(\alpha(r)\theta)\eta \\ Y(\xi,\eta) &= -\sin(\alpha(r)\theta)\xi + \cos(\alpha(r)\theta)\eta \end{aligned} \tag{3}$$

where the parameter θ defines the maximum skewness of the grid, and $\alpha(r)$ allows us to smoothly connect the uniform region of the grid with the rotated region. For our problems, we have defined $\alpha(r)$ as

$$\alpha(r) = \begin{cases} (1+\cos(\pi r/R))/2 & r < R \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where r is the distance to the grid origin $(0,0)$. Figure 2 shows grids corresponding to values $\theta = \pi/2$, the value used in all tests shown here. The advantages of this grid are that (1) the boundaries are straight, thereby eliminating any numerical issues that might arise at curved boundaries, (2) we can adjust the skewness of the grid with a single parameter and test how numerical errors are influenced by the skewness of the grid, and finally, (3) the initial values are concentrated in a region in which the pointwise values of the initial data and the cell averaged values are close, thereby eliminating the need to compute cell averages in cells which have been distorted by the grid mapping.

We solve the Euler equations with two sets of initial data, and in all both examples, report on the computed density. In the first example, we simulate a planar Mach 2 shock wave, and in the second example, we simulate a smooth acoustic wave with initial data

$$\begin{aligned} \rho_0(x,y) &= 1 + 0.25\exp(-5(x+1.5)^2) \\ p_0(x,y) &= \rho_0^\gamma, \\ u_0(x,y) &= \frac{2}{\gamma-1}\left(\sqrt{\gamma\rho_0^{\gamma-1}} - \sqrt{\gamma}\right) + 2. \end{aligned} \tag{5}$$

This data is chosen so that the wave is a simple wave that remains smooth over the time interval considered. In each case the solution feature is initially on the Cartesian portion of the grid and passes through the rotationally skewed portion. The true solution is purely one-dimensional. The computed density on the two- or three-dimensional mapped grid is compared with a fine grid or exact solution, and with a numerical solution to the one-dimensional equations computed using the wave-propagation algorithm on a grid with the same number of grid points as are used in the x-direction on the multidimensional grid. If the multidimensional grid were purely Cartesian ($\theta = 0$ in (3)) then the multi-dimensional results would match the one-dimensional results. On the skewed grid we do not expect the results to be as good and so this gives some basis for examining the effect of the grid mapping.

2.1 Two-dimensional results

In Figure 3 we show the two-dimensional results for the shock-wave test problem on a single (unrefined) grid with $\theta = \pi/2$. In spite of the skewed grid the contour

lines show that the numerical solution is essentially one-dimensional. The left plot in Figure 5 shows a scatter plot of the density in all grid cells, plotted against the x-coordinate of the cell center. This is compared with the exact solution based on the computed shock speed.

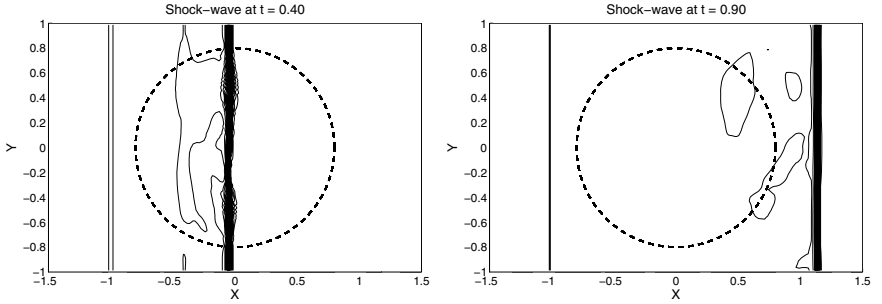


Fig. 3. Contours of the solution to shock-wave problem on a single grid at times $t = 0.4$ and $t = 0.9$. The dashed circle encloses the mapped region. The contour levels are $0.95 : 0.1 : 2.95$. The grid is 120 cells in x-direction ($dx = 0.025$).

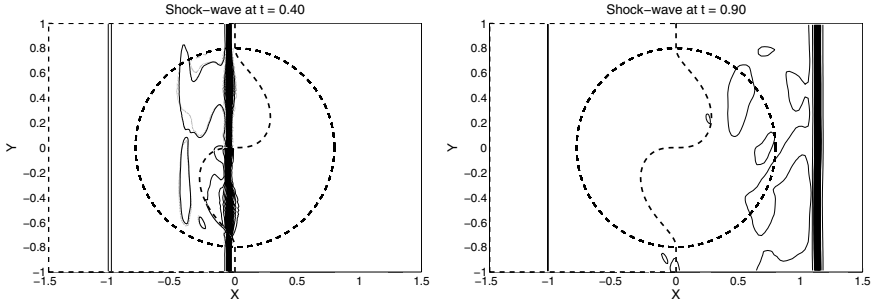


Fig. 4. Contours of the solution to shock-wave problem on a partially refined grid at times $t = 0.4$ and $t = 0.9$. The dashed circle encloses the mapped region, and the additional dashed center-line is the right edge of the refined region. The contour levels are $0.95 : 0.1 : 2.95$. The coarse grid resolution for these plots is 120 cells in x-direction ($dx = 0.025$).

We now turn to testing the behavior of the solution when the feature passes through a fine-coarse grid interface. We force a static refinement in which the left portion of the computational domain is refined by a factor of 2 relative to the right portion, as shown in Figure 2. Note that the refinement boundary is chosen to be in the region of maximum grid deformation. In this region the fine and coarse grid cells do not exactly match up, as was shown in Figure 1, and a concern is that the

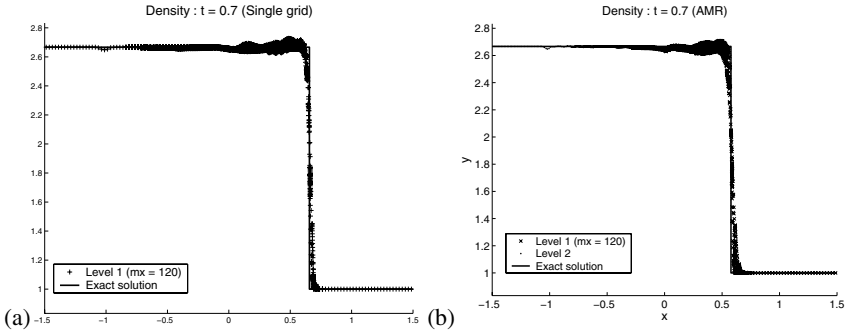


Fig. 5. Scatter plot of shock-wave solution computed on (a) a single grid and (b) after is passed through a coarse-fine interface on a partially refined grid.

interface will lead to the generation of spurious reflections or other numerical noise at the interface. Of course this can be a problem at a refinement interface even on a purely Cartesian grid, but the grid deformation and mismatch of cells heightens the concern.

Our results indicate that the wave-propagation algorithm appears to behave well in this regard, at least on the test problems used here. The two plots in Figure 4 show contour lines of the shock wave in the middle of the mapped region, and again once it has completely left the mapped and refined region. Figure 5 shows scatter plots of the solution computed on a single coarse grid (left plot) and a partially refined grid (right plot). For comparison, the exact solution, based on the computed shock speed is also shown on both plots. What we hope to observe, is that the results on the partially refined grid are no worse than what was obtained on the unrefined coarse grid. In fact we do observe this, indicating that the existence of a refined patch does not introduce substantial errors at the grid interface. We also don't expect to see substantially better results on the partially refined grid, since the shock has already left the refined region by time $t = 0.7$.

Figure 6 shows similar plots for the smooth test problem with data (5). Figure 7 shows the results of a grid refinement study on this smooth test problem. The upper curve shows the L_1 norm of the error found on a sequence of unrefined grids, indexed by the number of grid points in the x-direction. The lower curve shows corresponding results for a sequence of partially-refined grids, indexed by the number of grid points in x on the coarse grid. These results show that the method remains second order accurate in spite of the refinement boundary. The error is no larger on the partially-refined grid than on the single grid, and in fact is slightly smaller since the solution was initialized on the finer grid. This indicates that excessive errors are not being introduced at the refinement boundary.

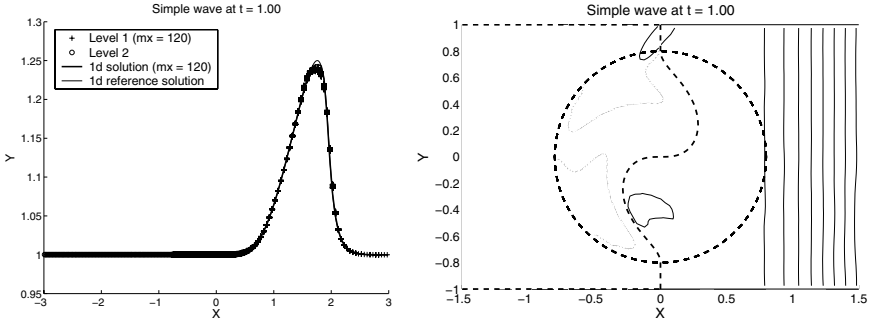


Fig. 6. Scatter plot and contour plot of simple wave solution. Both plots show the solution computed on a grid in which the left half was refined. Coarse grid resolution is 120 cells in the x-direction. ($dx = 0.05$).

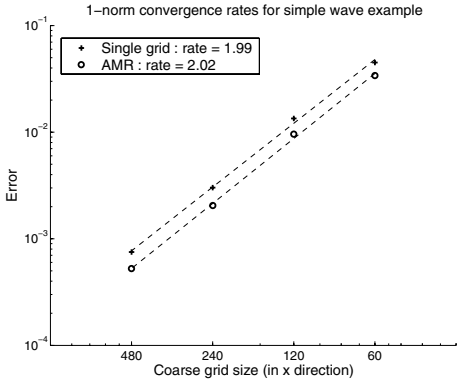


Fig. 7. Convergence results for simple wave solution. Two curves show the single grid errors (top curve) and the errors on a mesh in which the left half was refined (bottom curve).

2.2 Three-dimensional results

Tests have been performed for the three-dimensional hexahedral generalization of these test problems, with similar results. The grid used for these simulations is the three-dimensional analog of the two-dimensional grid used above. In three-dimensions, we rotate a spherical region of radius $R = 0.8$, centered at the origin, by an angle $\alpha(r)\theta$ about a vector v. For the following simulations, we set $\theta = \pi/2$ and $v = (1, 1, 1)$. The function $\alpha(r)$ is defined as in (4). The three dimensional grid is shown in Figure 8.

To determine necessary geometric quantities for each mesh cell, we approximate the hexahedral mesh by a trilinear map

$$\begin{aligned} T(\xi,\eta,\zeta) =& a_{000} + a_{100}\xi + a_{010}\eta + a_{001}\zeta + a_{110}\xi\eta + \\ & a_{101}\xi\zeta + a_{011}\eta\zeta + a_{111}\xi\eta\zeta \end{aligned} \tag{6}$$

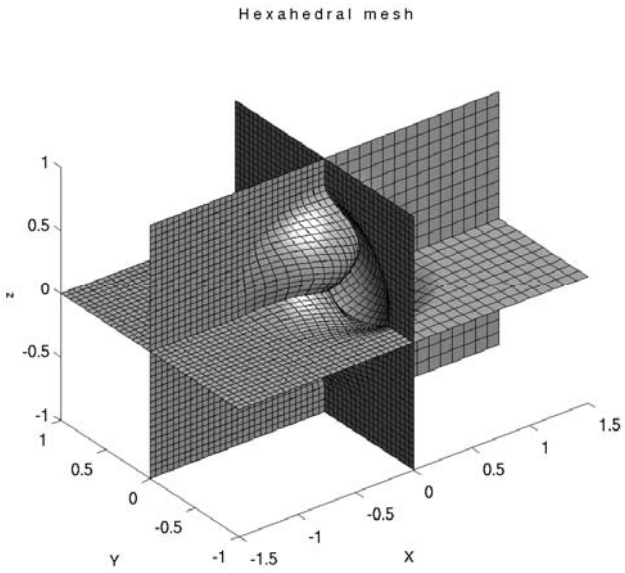


Fig. 8. Hexahedral mesh used for three-dimensional simulations. The center region has been rotated about the vector $v = (1, 1, 1)$ by a skewness factor of $\theta = \pi/2$. This plot shows every other grid line in a mesh with 60 coarse grid cells in the x-direction ($dx = 0.05$).

where $0 \leq \xi, \eta, \zeta \leq 1$. The coefficients $a_{lmn} \in \mathcal{R}^3$ are computed from the locations of the physical corners of the mesh cell. This approximation is the natural extension of two dimensional bilinear quadrilateral mesh cells to three dimensions. Using this approximation, we compute a set of normal and tangential vectors at the midpoint of each face, a surface area for each face, and the volume of the approximated mesh cell. These quantities are then used in the Riemann problems and in update formulas in a manner analogous to the what we do in two-dimensions.

To test our code, we use the same simple wave and Mach 2 shock examples that we discussed in our two dimensional test problems. In Figure 9, we show a scatter plot and a contour plot, on a $z = 0$ slice of data for the smooth-wave computed on the partially refined grid. In Figure 10 we show the results of the Mach 2 shock-wave, on a ($z = 0$) slice, at two different times, and in Figure 11, we show a scatter plot of the same shock-wave results. In both examples,the results are no worse than results computed on a single grid at the coarse grid resolution, indicating that the introduction of a refined grid and a coarse/fine boundary does not cause spurious waves or reflections. The accuracy does not appear to be as good as on the two dimensional grid, but this is due in part to the fact that we are computing the three dimensional solution on a grid with half the resolution as was used in the two-dimensional example.

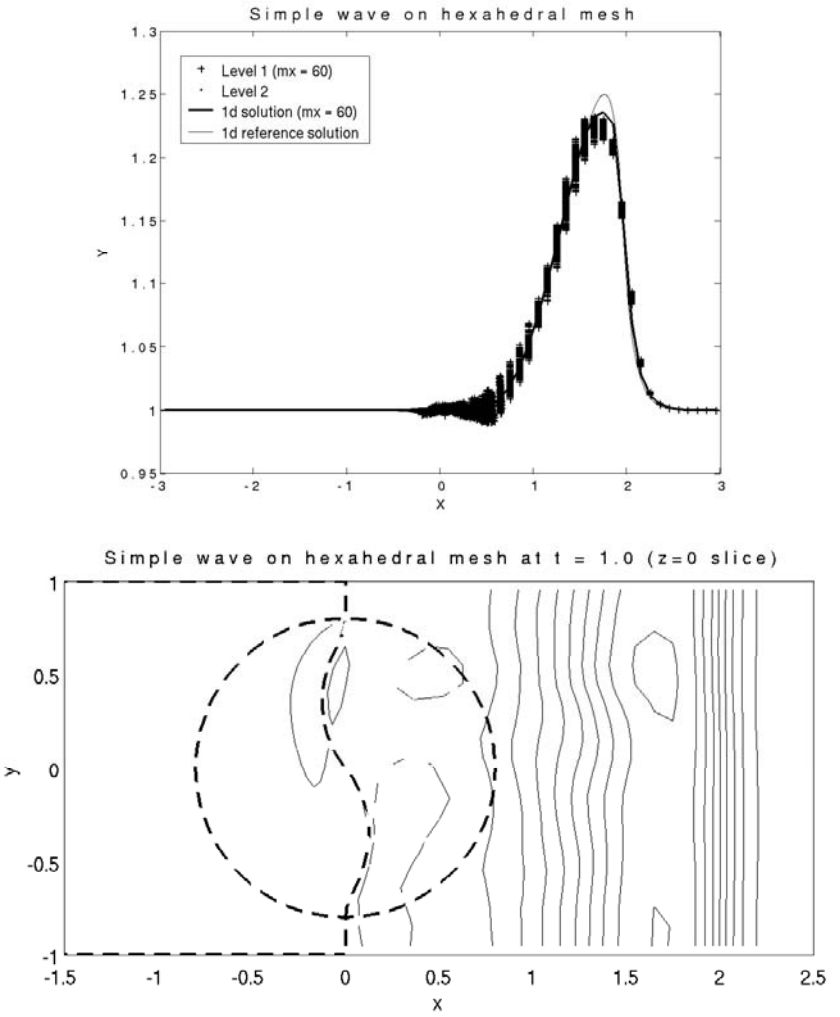


Fig. 9. Solution to simple wave problem computed on a partially refined hexahedral grid. The coarse grid resolution is 60 grid cells in the x-direction ($dx = 0.05$).

3 Conclusions

We have proposed a simple test problem to study the effect of introducing a mesh refinement boundary when solving conservation laws on a mapped grid. A static refinement interface is introduced in a highly skewed region of the grid and both a shock wave and a smooth simple wave are passed through this interface to test for spurious reflections or loss of accuracy. We have tested the wave-propagation algorithm in two and three dimensions and have observed results that are better than

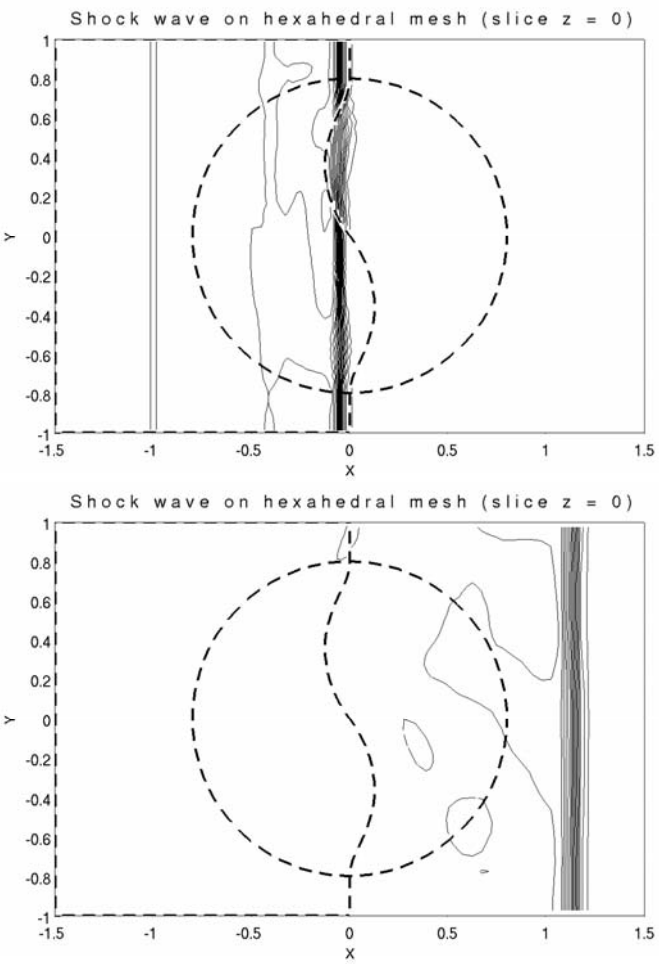


Fig. 10. Solution to shock-wave problem computed on a partially refined hexahedral grid. The coarse grid resolution is 60 grid cells in the x-direction ($dx = 0.05$).

what is obtained on an unrefined grid. This is encouraging since a refinement strategy is used in which the grid cells are not perfectly aligned between coarse and fine cells, which could potentially generate significant errors at the interface.

Acknowledgment. This work was supported in part by DOE grant DE-FC02-01ER25474 and NSF grant DMS-0106511.

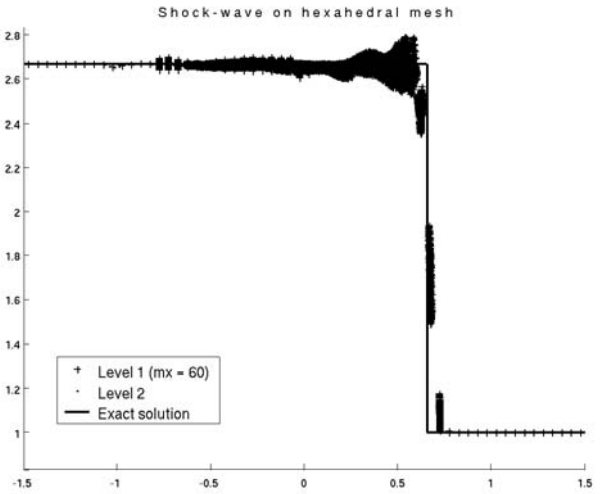


Fig. 11. Scatter plot of density in the shock-wave problem computed on a hexahedral mapped grid. The coarse grid resolution is 60 grids in the x-direction ($dx = 0.025$).

References

1. J. B. Bell, P. Colella, J. Trangenstein, and M. Welcome. Adaptive mesh refinement on moving quadrilateral grids. Technical report, Lawrence Livermore National Laboratory, April 1989.
2. M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.
3. M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.
4. M. J. Berger and R. J. LeVeque. AMRCLAW software. Available on the Web at the URL `http://www.amath.washington.edu/~claw/`.
5. M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave propagation algorithms for hyperbolic systems. *SIAM J. Num. Anal.*, 35(6):2298–2316, 1998.
6. J. O. Langseth and R. J. LeVeque. A wave-propagation method for three-dimensional hyperbolic conservation laws. *J. Comput. Phys.*, 165:126–166, 2000.
7. R. J.. LeVeque. CLAWPACK software. Available on the Web at the URL `http://www.amath.washington.edu/~claw/`.
8. R. J. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *J. Comput. Phys.*, 131:327–353, 1997.
9. R. J. LeVeque. *Finite volume methods for Hyperbolic problems*. Cambridge University Press, 2002.
10. E. Steinthorsson, D. Modiano, and P. Colella. Computations of unsteady viscous compressible flows using adaptive mesh refinement in curvilinear body-fitted grid systems. Technical Report NASA Technical Memorandum 106704, NASA, 1994.

Efficiency Gains from Time Refinement on AMR Meshes and Explicit Timestepping

L. J. Dursi[1] and M. Zingale[2]

[1] Dept. of Astronomy & Astrophysics, The University of Chicago, Chicago, IL 60637 (ljdursi@flash.uchicago.edu)
[2] Dept. of Astronomy & Astrophysics, The University of California, Santa Cruz, Santa Cruz, CA 95064 (zingale@ucolick.org)

Summary. We consider potential efficiency gains for time sub-cycling, or time refinement (TR), on Berger-Collela and oct-tree AMR meshes for explicit or local physics (such as explict hydrodynamics), where the work per block is roughly constant with level of refinement. We note that there are generally many more fine zones than there are coarse zones. We then quantify the natural result that any overall efficiency gains from reducing the amount of work on the relatively few coarse zones must necessarily be fairly small. Potential efficiency benefits from TR on these meshes are seen to be quite limited except in the case of refining a small number of points on a large mesh — in this case, the benefit can be made arbitrarily large, albeit at the expense of spatial refinement efficiency.

1 Introduction

1.1 Block-Structured AMR

Adaptive mesh refinement on rectangular grids (henceforth AMR) was introduced in [3], and improved for conservation laws in [2], henceforth BC89. In the patch-based meshes of the sort described in BC89, the patches increase in resolution by a fixed even integer factor N. One can place a finer patch anywhere in the domain of a 'parent' patch of one fewer level of refinement. A patch is not required to have only a single parent, but must be completely contained within patches of the next lowest level of refinement. Note that these meshes are non-conforming; the face of a zone in a parent patch will abut N faces in the child patch. A final restriction in the nesting of the meshes is that there must be at least one zone of the next lower level refinement about the perimeter of a patch.

Another mesh we will consider here is an oct-tree mesh (quad-tree in 2-d, binary tree in 1d), such as is implemented in the PARAMESH package [6] used in the FLASH code [5]. This oct-tree mesh is a more restrictive version of an $N = 2$ patch-based mesh as described in BC89. If a block needs additional resolution somewhere in its domain, the entire block is halved in each coordinate direction, creating 2^d children, where d is the dimensionality. Leaf blocks are defined to be those blocks

with no children, and are thus at the bottom of the tree — they are the finest-resolved blocks in their region of the domain. Frequently, only leaf blocks are evolved to compute the solution to the equations, since a refined parent block's domain is completely spanned by its children.

The only difference between the two meshing approaches of immediate interest is the resulting different refinement patterns. We will use 'patch' and 'block' interchangeably in this paper.

1.2 Time Refinement

In BC89, the timestep set by the data on the finest mesh is used to evolve that data, and data on the coarser meshes is evolved at a multiple thereof so that there is a constant ratio at each level l of Δt_l to Δx_l. The assumption here is that there is one roughly spatially constant characteristic speed throughout the entire domain, so that the maximum allowable timestep at any given resolution is directly proportional to the size of the mesh for any given block or patch. When coupled with the assumption in structured AMR of some fixed jump in refinement between levels, this makes for a very natural time evolution algorithm, shown pictured in Figure 1 for a mesh with three different levels of refinement, with resolution jumps by constant factors of N; shown is $N = 2$.

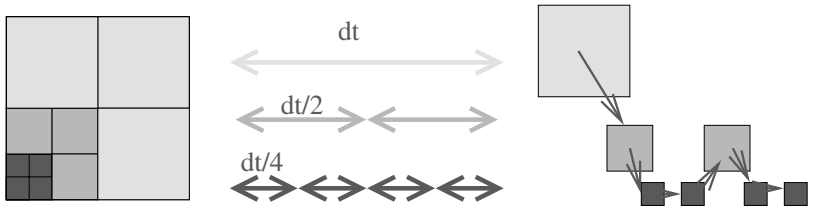


Fig. 1. A structured AMR mesh containing blocks at three different levels of refinement, showing the order of operations (far right) of an explicit time evolution algorithm. The largest block is evolved at the system timestep, and smaller blocks are subcycled at smaller timesteps. Between evolution at different levels of the mesh, time averaging and flux corrections must be done — these are not shown here.

Here the largest blocks are evolved at some system timestep dt, and smaller blocks are 'subcycled' at proportionally smaller timesteps. This defines a 'work function' for each block; the finest blocks must be evolved every sub-timestep so we take their work value to be 1 times the number of zones in the block or patch; the blocks one level of refinement 'up' need only be evolved every N sub-timesteps, so that their work value is $1/N$ times the number of zones, etc. The work function for an entire mesh is the sum of the work values of each block or patch in the mesh.

There are costs associated with this time refinement (hereafter TR). Memory is needed to store information at multiple timesteps. There are overheads from extra

copies and time-centering of fluxes. The modified time-structure of work leads to load-balance issues in parallel jobs. Further complicating parallel performance is increased communication complexity (although, it is to be pointed out, not necessarily increased communication).

Nonetheless, one might hope that these costs are outweighed by the time savings of not evolving large blocks at unnecessarily small timesteps; in the example of Figure 1, of evolving the larger blocks at timesteps of dt or $dt/2$ instead of $dt/4$. As a first step to quantify the possible benefits, we estimate the reduction in computational cost in simple cases §2. We then use the same approach to examine meshes from simulations performed with a tree-based mesh in §3. In our final section we summarize our results.

2 Simple Mesh Configurations

Here we calculate both the number of evolved blocks in a simple mesh, and a weighted sum representing the ideal amount of work done by a TR method, using the work function described in the previous section. We then calculate a work ratio, R — the amount of work that would be done by the idealized TR divided by that done with no time refinement. With no time refinement, each block must be stepped through each sub-timestep, so that the amount of work done is simply the number of blocks; thus, the work ratio is simply (TR work function)/(number of blocks). For $R = 1$, there is no reduction in work; for $R < 1$, TR reduces the amount of computational work.

One can interpret the work ratios as performance metrics for the TR, assuming that – all physics benifits from the time subcycling in proportion to the reduction in number of blocks evolved each step; the memory overhead from TR is unimportant; all larger blocks actually *can* be evolved at timesteps of larger size in proportion to their physical size; there is no single-processor overhead from TR from memory copies or flux averaging; there is no parallel overhead from increased complexity in communications; and there is no parallel from increased load-balancing issues.

2.1 Point refinement

The best case for efficiency gains for spatial refinement is clearly one isolated point of refinement. For a patched-based mesh, we imagine refinements as shown on the left of in Figure 2.

We begin with domains of length one in all directions. The completely unrefined domain is defined to be at level $l = 1$ of refinement. Consider placing increasingly fine patches at the corner, until we resolved the finest scale Δx we wished. If this requires $L - 1$ more levels of refinement, each decreasing the zone size by an integer factor N, then we have $\Delta x \sim (1/N)^{L-1}$. We will assume $\Delta x \ll 1/2$.

We consider the mesh in terms of the smallest uniform unit — for the oct-tree mesh, this is a single block, which will be of size $n_x \times n_y \times n_z$ zones. For the patch-based mesh, since the patches can be of arbitrary size (and shape), we consider zones

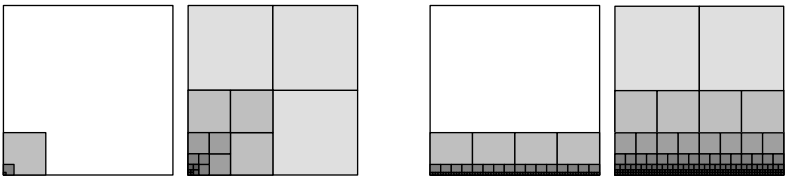


Fig. 2. Fully refining a zero-thickness point with an idealized patch-based type mesh (far left) and an oct-tree mesh (left); Fully refining an interface with a patch-based mesh (right) and oct-tree mesh (far right). For the patch-based mesh, it is assumed that a patch can be placed anywhere on existing patches, with some fixed integral increase in resolution (shown here is $N = 4$, $L = 3$). For the oct-tree mesh, N is fixed at 2, and shown is $L = 5$.

individually. (Because we are not modelling guardcell filling, we can safely ignore the fact that these zones are actually components of patches). Thus, in the results given below, an oct-tree mesh with (say) 8×8-zone blocks at a maximum refinement $L = 5$ has the same resolution as a patch-based mesh with $L = 8$.

The amount of work required by a non-TR code with only explicit or local solves will, by assumption, be the same for each block, so that $W_{\mathrm{noTR}} = N_{\mathrm{blocks}}$. The amount of work with time refinement, W_{TR}, will be a weighted sum of blocks. For the pointwise-refined patch mesh, the number of blocks will simply be $N_{\mathrm{blocks}} = L$, as there is only one block per level. The amount of TR work is

$$W_{\mathrm{TR}} = \sum_{l=1}^{L} 1 \cdot \left(\frac{1}{N}\right)^{L-l} \sim \frac{N}{N-1}. \tag{1}$$

Thus the work ratio will be

$$R = \frac{W_{\mathrm{TR}}}{W_{\mathrm{noTR}}} = \frac{W_{\mathrm{TR}}}{N_{\mathrm{blocks}}} = \frac{N}{L(N-1)}. \tag{2}$$

For ideal spatial AMR, where one can do all the refinement with only one jump, $L = 2$, and so the amount of work done by a TR algorithm is bounded from below at $1/2$ of the non-TR work. At the other limit, for a much less aggressive AMR with $N = 2$, then the work can be made an arbitrarily small fraction of the non-TR algorithm, with $R = 2/L$ — but note that this work ratio is achieved only by operating on $L/2$ times as many blocks as in the best case for spatial AMR.

The oct-tree meshes refining on a point is shown on the right of Figure 2. In this case, there are 2^d highest refined blocks in the corner, with the rest of the $2^d - 1$ surrounding blocks at the next highest refinement, surrounded by the $2^d - 1$ surrounding blocks at the next highest level of refinement, and so on.

Thus the total number of leaf blocks is

$$N_{\mathrm{blocks}} = (2^d) + \sum_{l=L-1}^{1} \left(2^d - 1\right) = 2^d(L-1) - L + 2 \tag{3}$$

Weighting them by the amount of work,

$$W_{\mathrm{TR}} = (2^d) + \sum_{l=L-1}^{1} \left(\frac{(2^d - 1)}{2^{(L-l)}} \right) \sim 2^{(d+1)} - 1 \tag{4}$$

making the work ratio

$$R = \begin{cases} 3/L & 1d \\ 7/(3L-2) & 2d \\ 15/(7L-6) & 3d \end{cases} \tag{5}$$

As with the patch-based result, this ratio goes to zero for arbitrarily large L. These results are similar to the $N = 2$ patch-based result, but TR performs better here, and the spatial refinement worse — both of these are due to the fact that the oct-tree mesh generates more intermediate-level blocks.

2.2 Planar Interface Refinement

The refinement of an interface is shown on the right of Figure 2. In the patch-based case, we continually place a grid of N-by-1 (in 2d) or N^2-by-1 (in 3d) patches along the interface, until the required resolution is achieved.

In this case, performing the same calculation as in the previous section, one obtains

$$R \approx 1 - \frac{N-1}{N^d - 1}. \tag{6}$$

Here, there is a fixed lower bound for the amount of work the TR can achieve. In the spatially-optimal large-N limit, no work is saved at all: $R \to 1$. At the other limit, for $N = 2$, in 2d, $R \to 2/3$; in 3d, $R \to 6/7$.

In the oct-tree mesh we begin with one block at the coarsest level. It must be divided into 4 in this 2D example, or, in general, 2^d. Half of these blocks will be further refined. This continues until we reach the maximum level of refinement. The work ratio one finds is

$$R = \begin{cases} 7/9 & 2d \\ 45/49 & 3d \end{cases} \tag{7}$$

In the point-refinement case of the previous subsection, a point of zero volume needed to be refined; as a result, there were the same number of blocks at each level, and thus a significant time savings could be obtained by doing less work at the coarser blocks. However, as we begin to see here, as soon as a non-trivial volume of the mesh needs to be refined, there is significantly less savings to be had.

2.3 Circular Region Refinement

The loss of efficiency gains when a non-zero fraction of the mesh must be refined is even clearer when a region, rather than an interface, is fully refined. In Fig. 3 we see the results of fully refining the interior of a quarter-circle with the center at one of the corners of the domain. Clearly, the number of finest blocks greatly outnumber intermediate or large blocks, so one might guess that there is very little efficiency gain that can be had from reducing work on the larger blocks.

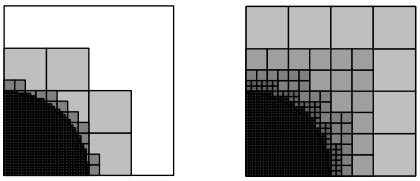


Fig. 3. Fully refining the interior of a circle, shown here with radius of 0.49 of the box size, with an idealized patch-based type mesh (left) and an oct-tree mesh (right). The patch-based mesh shown has $L = 3$ and $N = 4$. For the oct-tree mesh, N is fixed at 2, and shown is $L = 6$.

Table 1. Work ratio for a 2d Oct-tree mesh with a circular region of radius r (in units of the domain) completely refined.

	L=2	3	4	5	6	7	8
r = 0.0	0.786	0.625	0.510	0.426	0.363	0.316	0.279
0.1	0.786	0.625	0.510	0.510	0.638	0.765	0.879
0.2	0.786	0.625	0.625	0.714	0.806	0.895	0.940
0.5	0.962	0.843	0.851	0.888	0.931	0.963	0.981
0.9	1.	0.973	0.962	0.962	0.973	0.982	0.989

Because in this case the refinement pattern is complicated enough that the process must be iterated to check that each zones neighbors are no further than one level of refinement appart, we do not provide analytic work ratios. Tables 1 and 2 show the work ratios for an Oct-Tree mesh and an $N = 2$ patch-based mesh in refining a circular region of radius r. Again, the $r = 0$ results reproduce the expected point refinement, but as soon as a non-zero radius must be refined, the efficiency gains drop significantly further than in the case of only refining an interface, as more small blocks are needed to refine a region than the interface. In Table 2 we also show results for the patch based mesh with $N = 4$; we see as in previous sections that for the same resolution, increasing N (which increases the spatial efficiency of AMR) decreases the possible gains from time subcycling.

Table 2. Work ratio for a 2d patch-based mesh, $N = 2$ and $N = 4$, with a circular region of radius r (in units of the domain) completely refined.

	N=2, L= 2	3	4	5	6	7	8	N=4, L=2	3	4
r = 0.0	0.583	0.468	0.387	0.328	0.283	0.249	0.221	0.438	0.332	0.510
0.1	0.583	0.468	0.444	0.552	0.658	0.754	0.802	0.719	0.891	0.510
0.2	0.583	0.548	0.618	0.694	0.768	0.806	0.833	0.812	0.914	0.625
0.5	0.75	0.737	0.763	0.798	0.825	0.840	0.848	0.896	0.938	0.851
0.9	0.847	0.827	0.826	0.833	0.842	0.848	0.852	0.938	0.947	0.962

3 Meshes from simulations

The calculations of the previous section are for very simple refinement geometries. In this section, we apply the same work function used in §2 to the output of previous actual AMR simulations which use oct-tree based meshes for AMR. We continue to assume the same idealized performance results of the previous section.

We begin with examining results from a standard test problem, a Sedov explosion [7], as included with the FLASH code and described in [5]. In this simulation, a high pressure at a point causes a spherical shock wave to expand outwards; this is analogous to the circular region analysis of the previous section. The adaptive mesh for different stages of this simulation in 2d are shown in 4.

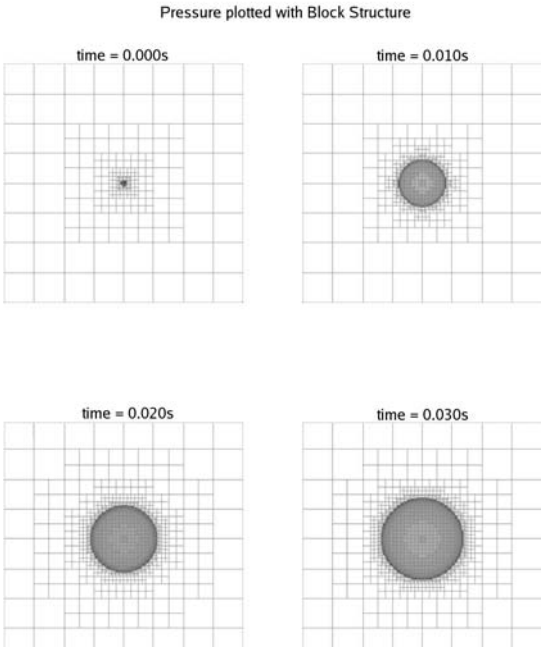


Fig. 4. The mesh of a Sedov explosion, from the FLASH setup test described in [5], with a maximum of 8 levels of refinement. Each block shown contains 8×8 zones.

Results from the meshes shown are tabulated in Table 3. The number of blocks listed in the table is the number of 'leaf' blocks – *e.g.*, the blocks that are actually evolved. Also given in the table is the work ratio (R) and the work ratio of spatial AMR to a uniform mesh at the highest resolution ($R_{\mathrm{AMR}} = W_{\mathrm{noTR}}/W_{\mathrm{uniform}}$). We include R_{AMR} to compare the relative importance of performance gains for the spatial refinement and the time subcycling.

TR provides a large performance gain initially, when there is only one point that is refined. However, consistant with previous results, immediately as the point becomes a region of non-zero measure, idealized performance gains drop to 30%–10%. Regardless of the refinement, the TR provides a very small performance enhancement compared to that of the spatial refinement.

Table 3. Results from simulations of a Sedov explosion. Listed at different evolution times are the number of leaf blocks in the mesh, the work ratio, and the work ratio for spatial AMR to uniform grid.

time	N_{blocks}	R	R_{AMR}
0.00	256	0.426	0.0156
0.01	892	0.805	0.0544
0.02	1552	0.835	0.0947
0.03	2092	0.874	0.127

The reason for the small predicted efficiency gains, consistent with the discussion of the previous section, is that there quickly become more fine blocks than coarse blocks in the simulation. By the last frame shown in Figure 4, there are no blocks being evolved at the the coarsest level of refinement, and indeed 80% of the blocks are at the highest level of refinement. Thus, even if all other blocks required zero work to evolve, we could only achieve a 20% speedup.

Next we consider an interface problem – a 2d detonation that will eventually undergo a cellular instability. These simulations are from results published in [8]. A mesh is shown in Figure 5. This corresponds almost exactly to the idealized interface problem of the previous section, but here the domain is very long in one direction, increasing the number of low-cost coarsest blocks in the domain. This change in distribution of blocks means that this problem can benefit more from TR. The numerical results are shown in Table 4.

Fig. 5. Half of the domain for the initial condition of a detonation, where the long domain is refined nowhere except at a sharp interface. The domain originally consists of a top-level mesh of 1×20 blocks. This mesh is then refined at an interface. Shown is the meshes 6, zoomed in near the interface. Not shown are 10 coarsest blocks to the right.

Here we see TR's efficiency gains actually decrease with increasing resolution, and also see a familiar pattern of TRs efficiency gains going in the opposite direction of spatial AMR efficiency gains. Even at the resolution where TRs efficiency gains are largest, they are much smaller than the improvement from using spatial AMR.

Table 4. Results from initial conditions for a 2-d detonation problem, as in Figure 5. R is less than the $7/9$ calculated in the previous section, because of the large number of extra coarsest blocks added to the domain.

Max refinement	N_{blocks}	R	R_{AMR}
4	62	0.633	0.0484
5	110	0.688	0.0215
6	206	0.727	0.0101
7	398	0.751	0.00486

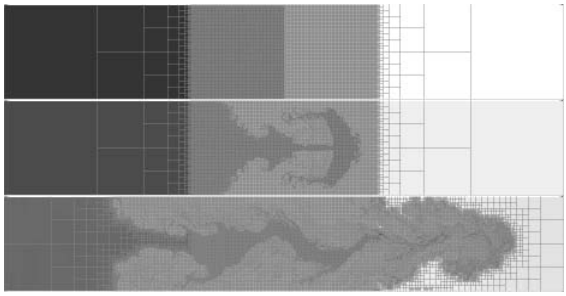


Fig. 6. Development of Rayleigh-Taylor instability at 3 epochs, from simulations presented in [4]. These are fairly high-resolution simulations, with a maximum of 8 levels of refinement on a top-level mesh with 6×1 coarsest blocks.

We next consider the development of the Rayleigh Taylor instability. (Figure 6). This is an interface problem, but in this set of simulations, the center region of the box is resolved to ensure resolution of the velocity perturbations in the region near the interface. Because this region is fully refined, many 'full cost' finest blocks are added. This decreases the scope of improvement from TR, as seen in Table 5.

Table 5. Numerical results from simulations of a Rayleigh-Taylor instability, shown in Figure 6.

time	N_{blocks}	R	R_{AMR}
0.0	33150	0.993	0.337
1.8	33150	0.993	0.337
3.6	60816	0.987	0.619

4 Conclusion

We have considered efficiency gains for time subcycling for explict or local physics. In these cases the work per block is roughly constant. Further, in most cases there

are many more fine blocks than coarse blocks — this is due to simple geometry, as a mesh that refines a significant fraction of its domain will be strongly weighted in favour of small blocks, which must be evolved at a small timestep. Thus, Any attempt to improve performance by focusing on the relatively few larger blocks can only reduce a small fraction of the work that needs to be done to evolve the system one timestep. On the other hand, in studies where only a small number of points in a large domain must be fully resolved, there may be significant efficiency gains from TR methods. Some cosmological hydrodynamical simulations [1] are examples of this situation.

We have not considered here accuracy; taking fewer timesteps may increase accuracy with some solvers, although this isn't clear for moderately time-accurate algorithms having errors of $O(\Delta t^p)$, $p > 1$; further, the coarsely refined regions which would benefit from the fewer timesteps are presumably coarsely refined because the overall solution quality is less sensitive to the error in those regions than it is to that of the highly refined parts of the domain. We also do not consider global or implicit solves, where the timestepping algorithm in Fig. 1 must be modified. Global or implicit solves will, depending on the methods used, change the amount of work done per block at different levels of refinement, which can change the results given here considerably.

We have modelled only computational cost in this work. Most of the other costs, cf. §2, work to decrease the efficiency gains of TR. One unmodelled effect that could increase the gains is the reduction of guardcell fills on large blocks. For the oct-tree mesh, where the number of zones per block is fixed, the reduction in guardcell filling work is reduced in the same way as the computational work, so that our conclusions are unchanged. For the patch-based mesh, the effect on the guardcell filling will be dependant on the shape of the refined region and the algorithm used for merging patches of the same refinement level, so that it is difficult to say anything in general.

Thus, block-structured TR significantly enhances performance of local or explicit physics solvers only under fairly narrow circumstances. In circumstances where TR is unlikely to produce much performance enhancement, the added code complexity, memory overhead, and parallel load-balancing issues may make the costs of the technique exceed its benefits.

The authors thank B. Fryxell for useful discussions with this paper, and K. Olson with his help with PARAMESH over the past years. We thank A. Calder for data from RT simulations, and F. X. Timmes for data from cellular detonation simulations. We thank T. Plewa, G. Weirs, R. Kirby, and R. Loy for suggesting this work. Support for this work was provided by the Scientific Discovery through Advanced Computing (SciDAC) program of the DOE, grant number DE-FC02-01ER41176 to the Supernova Science Center/UCSC. LJD was supported by the Department of Energy Computational Science Graduate Fellowship Program of the Office of Scientific Computing and Office of Defense Programs in the Department of Energy under contract DE-FG02-97ER25308.

The FLASH code is freely available at http://flash.uchicago.edu/.

References

1. T. Abel, G. L. Bryan, and M. L. Norman. The Formation of the First Star in the Universe. *Science*, 295:93–98, January 2002.
2. M. J. Berger and P. Collela. Local adaptive mesh refinement for shock hydrodynamics. *J. Comp. Phys.*, 82:64–84, 1989.
3. M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53:484–512, 1984.
4. A. C. Calder, et al. On validating an astrophysical simulation code. *ApJSS*, 143:201–230, 2002.
5. B. Fryxell, et al. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *ApJSS*, 131:273–334, November 2000.
6. P. MacNeice, et al. Paramesh: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126:330–354, 2000.
7. L. I. Sedov. *Similarity and Dimensional Methods in Mechanics*. Academic Press, New York, 1959.
8. F. X. Timmes, et al. On the Cellular Structure of Carbon Detonations. *ApJ*, 543:938–954, November 2000.

Using Krylov-Schwarz methods in an adaptive mesh refinement environment

Kai Germaschewski[1], Amitava Bhattacharjee[1], Rainer Grauer[2], David Keyes[3], and Barry Smith[4]

[1] Center for Magnetic Reconnection Studies, Institute for the Study of Earth, Oceans and Space, University of New Hampshire
[2] Lehrstuhl für Theoretische Physik I, Ruhr-Universität Bochum, Germany
[3] Department of Applied Physics and Applied Mathematics, Columbia University
[4] Mathematics and Computer Science Division, Argonne National Laboratory

Much of the previous work in AMR methods has concentrated on solving hyperbolic equations with explicit timestepping. However, for many problems, either due to their physical nature (e.g., incompressible flows) or for performance reasons (semi-implicit and implicit numerical methods), it becomes necessary to solve global equations.

This paper focuses on the application and performance of well-established preconditioned Krylov-Schwarz solvers in an AMR context, using a Krylov-Schwarz method to accelerate convergence while exploiting the hierarchical structure of AMR grids for multi-level preconditioning using the *fast adpative composite (FAC)* algorithm. We present an implementation that allows us to leverage the powerful supply of preconditioners and linear solvers from the PETSc library.

We apply this method to solve the three-dimensional Euler equations in the search for a finite-time singularity.

1 Introduction

The numerical integration of the three-dimensional Euler equations in order to follow a developing singularity is a highly demanding task which can be achieved only by employing a combination of sophisticated numerical methods.

In the following we describe three key ingredients which have been developed for the *Magnetic Reconnection Code* (MRC), which, while being designed with plasma flows in mind, is applicable to fluid problems as well:

- The integration scheme is a semi-implicit second-order in time and space combined Godunov/projection method.
- Adaptive mesh refinement is used to follow small-scale structures with appropriate high resolution while smooth parts of the domain are treated at lower resolu-

tions. At the same time, parallelization of the algorithms to work in a massively parallel environment is handled at this level.
- Solving the elliptic equations needed to calculate the pressure and maintain a divergence free velocity field is achieved though embracing and extending the PETSc library's Krylov-Schwarz solvers to a multi–level domain through McCormick's *fast adaptive composite (FAC)* algorithm. It should be noted that solving elliptic problems is also a prerequisite for many applications in plasma physics (divergence cleaning of the magnetic field and implicit treatment of the electron inertia in Hall-MHD).

2 Numerical procedure

2.1 Integration scheme

Looking at the incompressible Euler equations as the limiting case of compressible gas dynamics with compressibility κ going to zero reveals some insight into the challenges of numerical treatment.

Most notably, the pressure is an independent variable (related through an equation of state to the internal energy) in compressible gas dynamics, whereas it is just a function of the velocity field in the incompressible Euler equations.

This can easily be seen by taking the curl of the incompressible Euler equations, which makes the pressure term drop out completely:

$$\begin{aligned} \nabla \times (\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla p = 0) \quad , \quad \nabla \cdot \mathbf{u} = 0 \\ \implies \partial_t \omega + \mathbf{u} \cdot \nabla \omega + \omega \cdot \nabla \mathbf{u} = 0 \end{aligned} \tag{1}$$

with the vorticity $\omega = \nabla \times \mathbf{u}$.

On the other hand, taking the divergence of this equation, we find how to compute the pressure from the velocity field $\mathbf{u}$:

$$\begin{aligned} \nabla \cdot (\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} - \nabla p = 0) \quad , \quad \nabla \cdot \mathbf{u} = 0 \\ \implies \nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla^2 p \end{aligned} \tag{2}$$

The speed of sound in a compressible gas is easily determined by considering adiabatic compression and expansion and found to be $c_s = 1/\sqrt{\rho\kappa}$, with density ρ and adiabatic compressibility κ.

Explicit numerical integration of the compressible gas equations needs to resolve sound waves and is thus constrained by a CFL time-step limitation, given by (possibly modified by a constant factor)

$$c_s \Delta t / \Delta x \leq 1 \tag{3}$$

Obviously, it is not possible to simply transfer these explicit methods to the incompressible case, as the sound speed tends to infinity and thus requires the time-step Δt going to zero. In fact, in the incompressible case, a perturbation at one end of the

domain will instantaneously affect the field at the other end, whereas in an explicit scheme, the propagation speed is fundamentally limited to the stencil width (possibly multiplied by a small factor) per time-step.

This is the reason why for the incompressible equations (and equally as well for the compressible equations as long as one is interested in phenomena on time scales much slower than the speed of sound) an implicit or semi-implicit numerical treatment is needed.

In this work, we adopt a scheme based on work by Bell, Collela and Glaz[4], which is a projection method combined with a second-order accurate Godunov method. The scheme has already been successfully applied in an AMR context, e.g. in [7].

Integration is performed in the primitive variables. However, to integrate this scheme with adaptive mesh refinement (AMR), the fields which are communicated between different levels of resolution are the vorticity fields, allowing us to use only first-order interpolation as those represent the highest order derivatives occurring in the time-stepping calculations. Velocities are then calculated as the curl of a flow potential, obviously ensuring the divergence-free condition, where the flow potential is obtained from the vorticity by solving three elliptic equations. At the same time, a fourth Poisson equation is solved to evaluate the pressure using the relation (2) above.

This scheme, requiring the solution of four Poisson equations per time-step is computationally demanding, but at the same time it ensures that no numerical artifacts (like violation of the divergence-free condition) are introduced. Also, the time-step is limited only by the speed of the flow and is thus fairly large. The associated time scale needs to be resolved for accuracy reasons in any case.

The unsplit second-order Godunov scheme is described in detail in reference [4]. It uses a limiting slopes method in calculating approximations to spatial derivatives to prevent introducing new maxima or minima into the velocity field. In the actual time-stepping procedure, it exploits knowledge from treating the 1D Burgers equation, using an upwind scheme for actively transported quantities (using the solution for the Riemann problem):

$$\begin{aligned} &\partial_t u + u\partial_x u = 0 \\ \Longrightarrow\ & u_{i+1/2,j} = \begin{cases} u^L \text{ if } u^L \geq 0,\ u^L + u^R \geq 0 \\ 0 \quad \text{otherwise} \\ u^R \text{ if } u^R \leq 0,\ u^L + u^R \leq 0 \end{cases} \end{aligned}$$

For the passively transported quantities, the upwind scheme is

$$\begin{aligned} &\partial_t v + u\partial_x v = 0 \\ \Longrightarrow\ & v_{i+1/2,j} = \begin{cases} v^L & \text{if } u_{i+1/2,j} \geq 0 \\ 1/2(v^L + v^R) & \text{otherwise} \\ v^R & \text{if } u_{i+1/2,j} \leq 0 \end{cases} \end{aligned}$$

Table 1. Comparison of grid points needed with AMR to an equivalent uniform mesh simulation.

Level	# grids	# grid points
0	1	70225
1	83	146080
2	103	268666
3	153	545316
4	197	1042132
5	404	1926465
6	600	1967234

Grid points in adaptive simulation:	6976118
Grid points in non-adaptive simulation:	268730449
Ratio:	0.02

2.2 Adaptive mesh refinement

Looking for singularities, the method of adaptive mesh refinement is an ideal tool. Starting out from a uniform grid, the local truncations errors are monitored and new, refined grids are introduced to cover those regions where the error exceeds a certain threshold. In addition, buffer cells are included in the direction of anticipated movement of the small-scale structures to ensure sufficient resolution until the next refinement procedure. While not being generally useful for, e.g., turbulence studies, where small-scale structures form all over, adaptive mesh refinement proves very efficient for problems where only localized small scales occur, such as in the development of a singularity. It enables us to follow the evolving singular structures with a much higher precision than what a legacy uniform grid code could achieve.

To exemplify the efficency gain obtainable by using AMR, we show an analysis from previous work (current sheets developing in 2D MHD) in Table 1 [6]. Taking into account all the ghost cells needed for the adaptive treatment, the adaptive simulation still uses only 2 percent of the resources needed to solve the problem to the same precision on a uniform grid.

Within our integration with the PETSc [1, 2, 3] library, most of the operations applied during the numerical integration can be written in terms of a sparse matrix multiplication. An advantage here is that we can exploit PETSc's optimized linear algebra operations, which overlap communication costs for passing of non-local boundary data with computation of the processor-local data. It is of course still desirable to keep as much of the work local as possible, which is achieved by using a Hilbert-Peano curve for load-balancing. Our AMR technology uses a so-called oct-tree approach, i.e., we do not support patches of arbitrary extensions, but instead always break up one box which needs refinement into eight (four in 2D) new boxes of the same shape, which provides a slightly less effective covering but has advantages when it comes to parallelizing and load-balancing.

For visualization purposes, we show the adaptive covering with different levels of resolution for a 2D example, current sheets developing in Hall-MHD simulated

within our framework, in Figure 2b. The basic procedure of subdividing grids can clearly be seen there.

The basics of the load-balancing algorithm are visualized in Figure 2a. It shows a sample adaptive domain coverage with adaptive grids at different levels of resolution, as well as the Hilbert-Peano curve mapping those grids and the resulting distribution onto processors.

The Hilbert-Peano curve has the mathematical property that grids that are close in two or three dimensions, respectively, are on average also close on the one-dimensional curve. The one-dimensional curve is divided into equal intervals, the grids on each interval are assigned to a particular processor. In the figure, the colors denote different processors, and it can easily be seen that grids which are kept on a particular processor are generally clustered together, which means the local set of grids typically has a lot of common boundaries not requiring expensive inter-processor communication, but only few boundaries to grids on other processors.

2.3 Multi–level elliptic solvers

A major challenge for the numerical integration of the incompressible Euler equations is the need to solve elliptic equations on the grid hierarchy generated by the AMR framework. The underlying strategy employed here is a variant of the *fast adaptive composite algorithm (FAC)* developed by McCormick [9].

The derivation of FAC algorithm starts with the regular multi–grid algorithm and is best understood considering just two levels, coarse and fine.

Iterative methods, in the simplest case e.g. Gauss-Jacobi or Gauss-Seidel, are called *smoothers* in the multi–level context, since they tend to smooth out the residual – the difference between current approximation and desired right-hand side. Error components with small scales/high frequency are eliminated quickly by these methods, whereas large scales/low frequency modes converge slowly. Smoothers quickly achieve an intermediate approximation which, after removal of high-frequency modes, has only lower frequencies left – it looks *smooth*. The trouble is that from that point on convergence is slow, and one way out of this difficulty are multi-grid methods: the smooth error is restricted onto a coarser domain, where it appears as having higher frequency components, which are quickly removed using the same iterative schemes as on the coarser level. This trick can be applied recursively and will approach $O(N \log N)$, or in the case of *full multigrid* even $O(N)$ scaling.

The idea behind FAC is that the quality of the solution obtained from the coarse grid will actually be sufficient on the part of the domain where the field is smooth, whereas on the other part of the domain we have smaller scales that need to be fully resolved on the fine level. We can meet this goal by only executing the fine level iteration on a *local* subset of the domain. While this scheme gives a solution to the desired accuracy, it is not practical, since it requires us to store the fine level data on the entire domain, even in the areas where that resolution is not required.

However, it can easily be shown that the fine grid relaxation will only change the residual transferred to the coarse grid on the iterated domain plus the interface, so it

is not necessary to store the fine data everywhere but only on the refined region plus a certain border. This leads to the *bordered multi–level scheme* which is certainly practical, in particular in an AMR context where we need to take provisions to store boundary data for the time evolution in any case [9]:

> *Residuals are computed on the local fine grid including its boundary and first border; these residuals are then transferred to the coarse grid points lying ander the fine grid and at the interface; the correction is the computed and interpolated to the local fine grid including its boundary and both borders; relaxation is then performed at fine grid interior points.*

However, instead of using just a simple relaxation scheme like Gauss-Jacobi, we integrate this algorithm within the PETSc toolkit, giving us access to a large variety of preconditioners and Krylov-accelerators. At the same time, we can also exploit PETSc's optimized computational kernels and integrated parallelization support.

Effectively, the elliptic problem distributed over the grid hierarchy is mapped to a standard linear solve where the sparse matrix is determined by the stencil of the Poisson operator and the mapping of a three-dimensional AMR domain covering to a simple one-dimensional vector.

A main objective of the PETSc library is to efficiently solve this kind of distributed sparse matrix linear problem, using optimal Krylov-Schwarz solvers with multi–level preconditioning which approach $O(N)$ efficiency. The run-time behavior of those solvers is basically determined by recurring matrix-vector-products. The implementation of this important operation overcomes communication bottlenecks by overlapping the communication time needed to obtain non-local data with processor-local computation work, followed by completion of the job through computation involving the by then-arrived non-local data.

A demonstration of the adaptive elliptic solver can be seen in Figures 5 and 6, where the method described is used to solve a standard two-dimensional test problem. The problem at hand is an L-shaped domain, solving Laplace's equation $\nabla^2 u = 0$ with boundary conditions so chosen that the solution is $u = r^{2/3}\sin(2\phi/3)$. This solution has a singular derivative at the origin, making it a good problem to test a multi–level AMR domain covering, with grids of finer resolution added as one approaches the singularity at the origin.

Figure 1 shows the convergence of the solver, in spite of the asymmetry/singular behavior in the problem, we obtain a reasonably fast convergence rate.

3 Tests

To validate the new code, preliminary test runs have been performed on NERSC's seaborg IBM SP machine, using up to 1024 processors.

We use Kida's [8] initial condition for a high-symmetry periodic flow, originally designed to facilitate turbulence studies in Navier-Stokes.

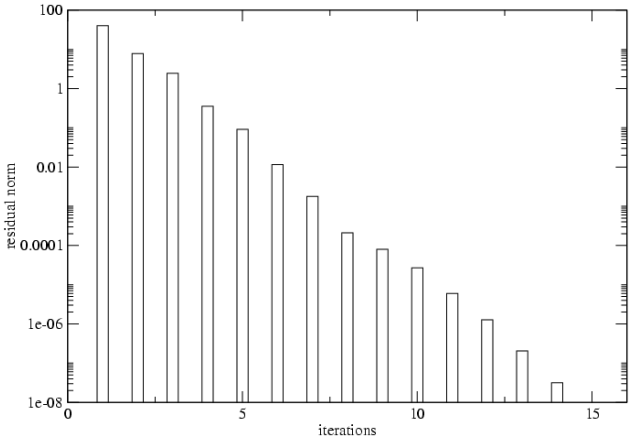


Fig. 1. Convergence of the multi-level elliptic solver for the 2-D test problem.

$$u_x(x,y,z) = \sin x(\cos 3y \cos z - \cos y \cos 3z) \tag{4}$$
$$u_y(x,y,z) = \sin y(\cos 3z \cos x - \cos z \cos 3x) \tag{5}$$
$$u_z(x,y,z) = \sin z(\cos 3x \cos y - \cos x \cos 3y) \tag{6}$$

This initial condition is known to form six vortex dipoles at the origin, which may evolve into a finite-time singularity [5, 10].

Figure 3 shows isosurfaces of the absolute value of the vorticity at times and magnitudes comparable to those given in [5] in one octant of the domain.

Figure 4 shows the y-component of the vorticity at the $y = 0$ plane evolving in time. The formation and beginning collapse of two vortex dipoles near the origin is clearly observed (the remaining 4 dipoles form in the x and z-component of the vorticity, not shown here).

For validation purposes, we compared numerical data from the pseudo-spectral code described in [5] and from the code described in [7], run with Kida's initial condition, to preliminary runs of the MRC code described in this paper and find good agreement in the vorticity isosurfaces as well as in the time evolution of maximum vorticity.

4 Conclusion

In this paper, we introduced the numerical methods used in the application of the Magnetic Reconnection Code (MRC) to solve the three-dimensional Euler equations.

We focused on presenting how the integration of Krylov-Schwarz methods with a hierarchical set of grids, obtained using block-structured adaptive mesh refinement, can provide a performant method to solve elliptic problems, as they occur in many problems in computational fluid dynamics / plasma flows.

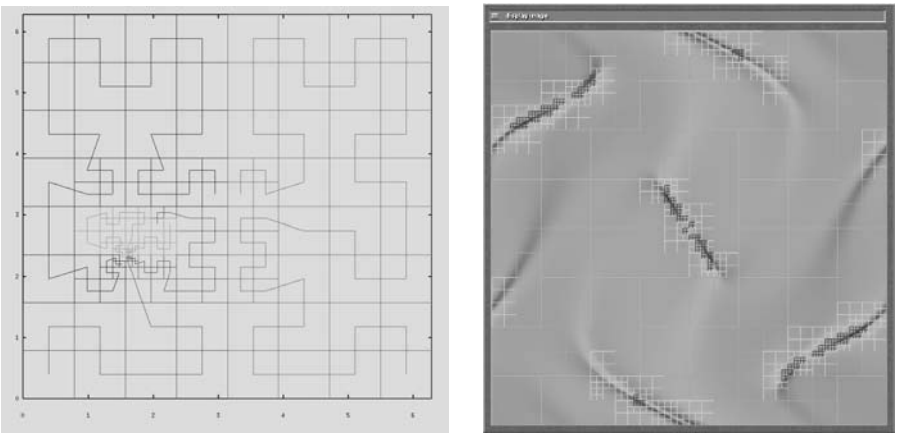


Fig. 2. a) sample adaptive domain coverage and Hilbert-Peano curve, b) Adaptive refinement of current sheets in 2D Hall-MHD.

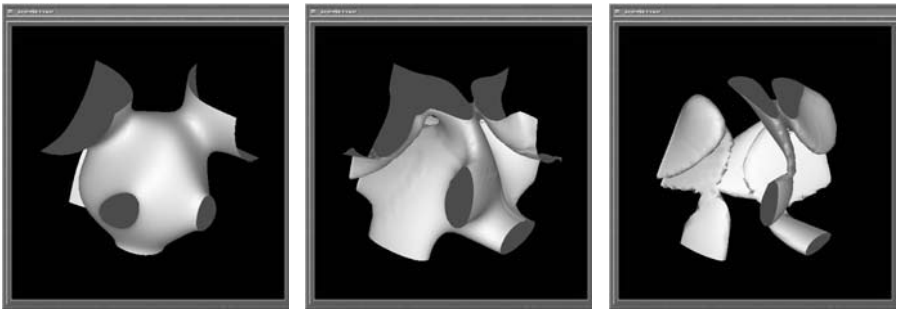


Fig. 3. Isosurfaces of the absolute vorticity value, at times 0, 0.525, 0.735, shown is one octant of the domain

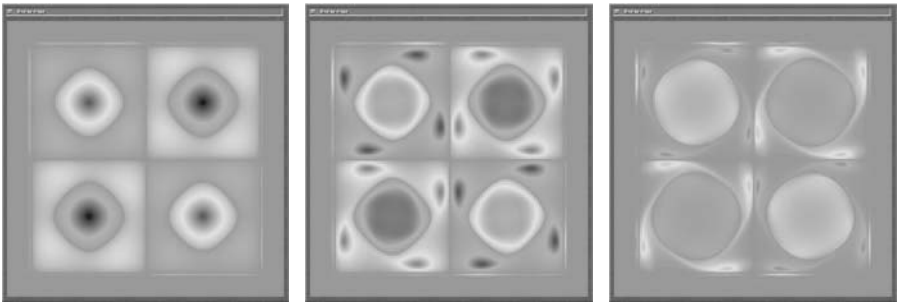


Fig. 4. y-component of the vorticity, cut through $y = 0$, at times 0, 0.5, 1.1

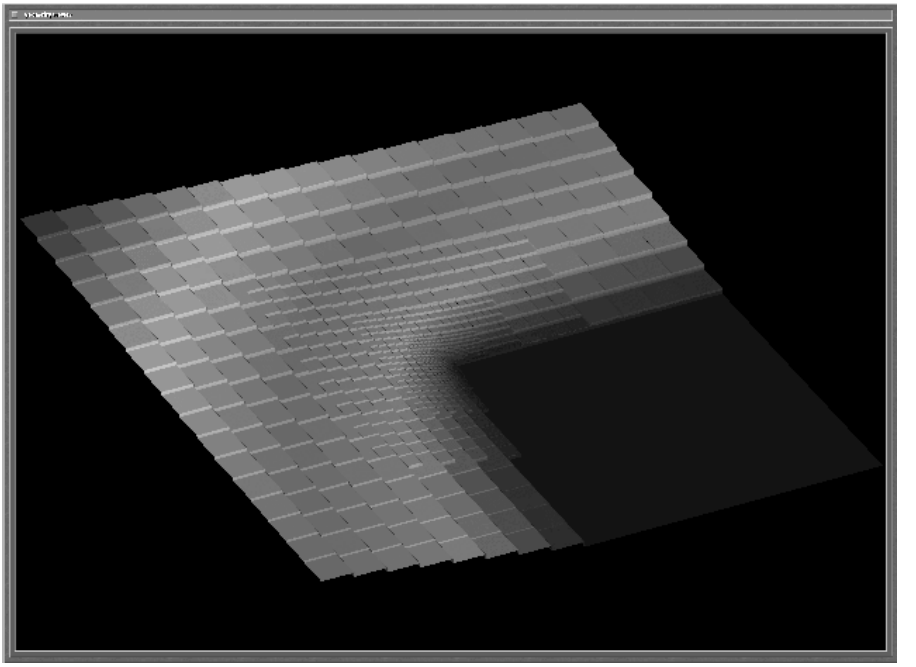


Fig. 5. 2D test problem solution for adaptive elliptic solver.

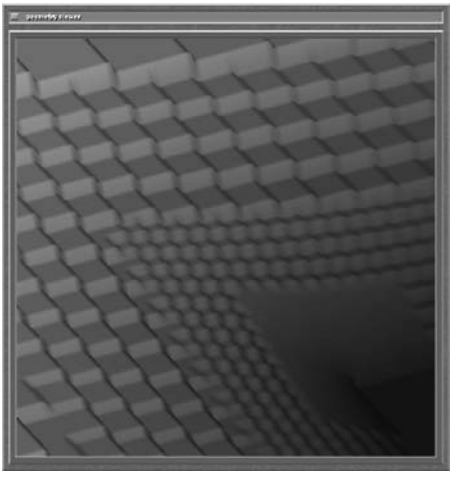


Fig. 6. Zoom into central region of Fig. 5.

Exploiting and extending existing technology, the PETSc library, enabled us to use a wide variety of well established sophisticated solver technologies and kept the implementation effort manageable.

At this point of writing, we have been able to perform only preliminary test runs of our new code. The results show promise for following an evolving singularity farther in time than has been possible previously with fixed uniform resolution.

References

1. Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matt Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc home page. http://www.mcs.anl.gov/petsc, 2001.
2. Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matt Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2002.
3. Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhauser Press, 1997.
4. J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier–Stokes equations. *J. Comput. Phys.*, 85:257–283, 1989.
5. O. N. Boratav and R. B. Pelz. Direct numerical simulation of transition to turbulence from a high-symmetry initial condition. *Phys. Fluids*, 6:2757–2784, 1994.
6. H. Friedel, R. Grauer, and C. Marliani. Adaptive mesh refinement for singular current sheets in incompressible magnetohydrodynamic flows. *J. Comput. Phys.*, 134:190–198, 1997.
7. R. Grauer, C. Marliani, and K. Germaschewski. Adaptive mesh refinement for singular solutions of the incompressible Euler equations. *Phys. Rev. Lett.*, 80:4177–4180, 1998.
8. S. Kida. Three-dimensional periodic flows with high-symmetry. *J. Phys. Soc. Jpn.*, 54:2132, 1985.
9. S.F. McCormick. *Multilevel adaptive methods for partial differential equations*, volume 6 of *Frontiers in Applied Mathematics*. SIAM, 1989.
10. R.B. Pelz. Locally self-similar, finite-time collapse in a high-symmetry vortex filament model. *Phys. Rev. E*, 55(2):1617–1626, 1997.

Dimensional Split Divergence-Free Reconstruction and Prolongation for Adaptive Mesh Refinement*

Shengtai Li[1]** and Hui Li[2]

[1] T-7, Los Alamos National Laboratory, Los Alamos, NM 87545
[2] X-1, Los Alamos National Laboratory, Los Alamos, NM 87545

Summary. A simple novel approach to preserve the divergence-free condition with adaptive mesh refinement is presented. The new approach uses only reconstructions on the coarse faces and the divergence-free condition to reconstruct the field values on the internal fine faces, and does not construct a global interpolation polynomial over a whole coarse cell. Therefore it can be easily applied to any refinement ratio. It is implemented via a directionally split approach in a directional splitting manner so that it can be applied to any kind of grids in any dimensions. Implementation is presented in the Cartesian, cylindrical and spherical geometries. It is shown by several 2D magneto-hydrodynamic simulations that such a method can keep the divergence-free error of magnetic fields at the round-off level.

1 Introduction

Adaptive and hierarchical grids provide some of the most efficient spatial discretization for multi-scale computational problems. It is of great interest to extend numerical schemes designed for a simple structured mesh to adaptive and hierarchical grids. It is critical to conserve the properties of the solutions when the mesh resolution changes. Berger and Colella [1] proposed an adaptive mesh refinement (AMR) scheme for hydrodynamics to conserve scalar quantities (e.g., mass, energy) and numerical fluxes. Additional challenges, however, are presented in physical systems satisfying the Stokes's law type of equation with the divergence-free evolution of vector fields. For example, it is important that the divergence-free condition is satisfied throughout the simulation for the velocity field in incompressible hydrodynamics and the magnetic field in magnetohydrodynamics (MHD).

The divergence-free evolution of the magnetic field is an important issue in developing a new MHD code even for a single non-adaptive grid. Brackbill and Barnes [3] have shown that the discretization error with respect to the divergence of the

*This research was performed under the auspices of the Department of Energy. It was supported by the Laboratory Directed Research and Development Program at Los Alamos.
**Email: sli@lanl.gov

magnetic field ($\nabla \cdot \mathbf{B}$) usually grows exponentially during the computations, causing an artificial force parallel to the magnetic field and destroying the correctness of the solutions. Many approaches (see Toth [11] and its references) have been proposed to handle this problem. In this paper, we consider the *constrained transport* (CT) method by Evans and Hawley [7]. CT method can keep $\nabla \cdot \mathbf{B}$ to the accuracy of machine round-off error. This approach has been combined with various shock-capturing schemes by many authors [6, 10, 2, 8]. The original CT method used a staggered grid which places the magnetic field variables in the face center and the rest in the cell-center. The divergence-free finite-difference scheme can be easily constructed for the staggered grid (see Yee [13]). Toth [11] introduced a finite-volume interpretation of the CT schemes that place all of the variables at the cell center. However, this idea is difficult to generalize to an AMR grid. In this paper, we adopted a CT approach, which is similar to the one of [2], implemented on the staggered grid.

How to preserve the divergence-free condition when the mesh is adapted remains a challenge. Balsara [4] and Toth and Roe [12] proposed a scheme to preserve this constraint exactly during the grid adaptation and time evolution. The basic idea is to construct a divergence free prolongation and restriction interpolation formula in the coarse cell being refined. This approach worked well for AMR with refinement ratio of two. However the interpolation polynomial can become quite complicated with the increase of the refinement ratio. Therefore, as suggested in [4], the procedure should be applied recursively for refinement ratio of 4 or 8. For other refinement ratios, new formulas have to be derived to account for the additional degrees of freedom needed to match the increased number of existing fine-grid faces.

In this paper, we propose a new approach, which can be implemented efficiently and uniformly for arbitrary refinement ratios and can be easily generalized to other types of orthogonal and curvilinear grid.

2 Algorithm

Before describing our algorithm, we first review briefly the algorithm of Balsara [4]. For the sake of simplicity, a 2D Cartesian grid is used to illustrate the algorithm. The generalization to 3D is straightforward. We then extend it to other types of orthogonal grids, e.g., the cylindrical and spherical coordinates. To simplify our description, we assume that each edge is split into equal-distance pieces according to the refinement ratio.

2.1 Fully divergence-free interpolation for AMR

Balsara's divergence-free prolongation can be viewed as a three-step procedure. First, the fine face components are obtained by the linear interpolation on zone faces. The component values on each face are determined by two parameters in 2D: an old coarse value and a limited slope, and three parameters in 3D: an old coarse value, and two limited slopes.

Since all the fine values on the coarse faces satisfy the divergence-free condition of the coarse cell, there are totally 4 × 2-1=7 independent parameters in 2D (or 6 × 3-1=17 independent parameters in 3D), which corresponds to seven (or 17 in 3D) independent constraints for interpolation.

Next the interpolation polynomials for B_x and B_y are constructed as

$$B_x(x,y) = a_0 + a_x x + a_y y + a_{xx}x^2 + a_{xy}xy + a_{yy}y^2, \quad (1)$$

$$B_y(x,y) = b_0 + b_x x + b_y y + b_{xx}x^2 + b_{xy}xy + b_{yy}y^2. \quad (2)$$

To match the interpolation polynomials to the linear profiles at the coarse faces, we obtain $a_{yy} = b_{xx} = 0$. The divergence-free condition $\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} = 0$ leads to

$$a_x + b_y = 0; \quad 2a_{xx} + b_{xy} = 0; \quad a_{xy} + 2b_{yy} = 0.$$

Therefore, there are only seven independent coefficients left in Eqs. (1) and (2). Evaluating the polynomials against the seven constraints yields seven independent linear equations. Inverting these linear equations yields the seven independent coefficients.

Finally the fine face values inside the coarse cell are obtained by evaluating the polynomials at specific locations.

As pointed out in [4], the linear profile on zone face *may not be sufficient* in many cases. When the mesh is readapted, the newly-created fine mesh can overlap the old fine and coarse meshes. The fine-coarse interface may not be represented by a linear profile. For the refinement ratio of 2, the linear profile is enough for 2D but not for 3D. Balsara [4] introduced a mixed derivative to resolve an extra freedom on the zone face profile in the 3D reconstruction, which results in a much more complicated interpolation polynomial. We expect that for a refinement ratio of r, we have $4r - 1$ independent coefficients in the interpolation polynomials in 2D (or $6r - 1$ in 3D). Furthermore, different refinement ratios correspond to different sets of equations, which leads to much more complexity in code development. Therefore, as pointed out in [4], Balsara's approach is intended for refinement ratio of 2. For refinement ratio of 2^n, it is used recursively.

2.2 Dimensional split reconstruction for Cartesian Grid

For the Cartesian grid, the face area is proportional to the local grid spacing in each direction. This is an important feature that can greatly simplify our algorithm. Later we will see that other types of orthogonal grids do not have this feature. For the ease of description, we will use a 2D example. Similar procedure in 3D can be deduced relatively easily.

Our approach is illustrated in sequence by plots (a), (b), (c), (d), and (e) in Fig. 1 with the following steps:

(1) We use a directional splitting approach to handle multi-dimensional problem and arbitrary refinement ratio in each dimension. In the first step, We assume the refinement happens only in x-direction. Similar to Balsara's method, we first determine the new quantities along the edges of the coarse cell according to the refinement ratio. As shown in (b), only v_1, v_2, v_3, v_4, v_5, and v_6 are to be determined. We do this

by either directly copying from the fine grids that shares the edges with the current coarse cell, or by interpolation using the coarse quantities along the edges. To achieve the second order accuracy, we use a piecewise linear profile on each face. This can be easily constructed for each coarse face if we know the slope in each direction. To preserve the monotonicity of the coarse face values, we also use a slope with an appropriate limiter as shown by Balsara [4].

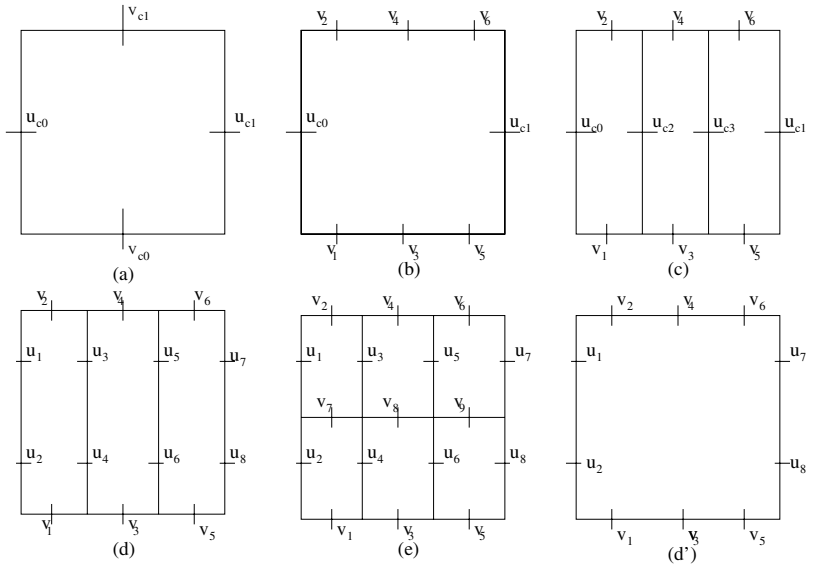


Fig. 1. Directional splitting approach for reconstruction of the divergence-free field on a fine grid. Balsara's approach is from (a) to (d') by linear interpolation or copying from the old fine grid, and then the divergence-free interpolation polynomials that match the boundary values are constructed over the whole coarse cell, and finally (e) is obtained directly from the evaluating of the polynomial at different positions.

(2) Given the quantities on the edges in (b), we can now partially construct the internal fine face values based on the divergence-free condition. For example, in (c) of Fig. 1, u_{c2} is uniquely defined by u_{c0}, v_1, v_2 as

$$u_{c2} = u_{c0} + (v_2 - v_1)\frac{\delta x}{\Delta y},$$

where δx and Δy are the spacings for the fine and coarse cells respectively. In the same manner, u_{c3} is calculated by $u_{c3} = u_{c2} + (v_4 - v_3)\frac{\delta x}{\Delta y}$. Then it is straight forward to show that the divergence-free condition is also preserved in the last cell by noting that

$$\frac{u_{c1} - u_{c3}}{\delta x} + \frac{v_6 - v_5}{\Delta y} = 0\,,$$

if

$$\frac{u_{c1}-u_{c0}}{\Delta x}+\frac{\frac{1}{3}(v_2+v_4+v_6)-\frac{1}{3}(v_1+v_3+v_5)}{\Delta y}=0\,.$$

(3) We now consider the refinement along the y-direction, which is taken as two in this example. As shown in (d), quantities u_1, u_2, u_7 and u_8 can be obtained similar to Step (1). In order to obtain quantities u_3 and u_4, we first obtain the slope along the $y-$direction at u_{c2}. This limited slope can be obtained by interpolation from the known values at the left and right faces of the coarse cell u_{c0} and u_{c1}, which yields,

$$\frac{du_{c2}}{dy}=\frac{1}{3}\frac{du_{c1}}{dy}+\frac{2}{3}\frac{du_{c0}}{dy}\quad . \tag{3}$$

The quantities u_3 and u_4 can then be calculated from u_{c2} and du_{c2}/dy. This step can be repeated to obtain u_5 and u_7.

(4) From the divergence-free condition, quantities v_7, v_8 and v_9 can be calculated.

We remark that the face values calculated in this way may not satisfy a uniform interpolation polynomial in a coarse cell. However, the reconstruction is of second order and the divergence-free condition is preserved at the center of each fine cell. Although the whole algorithm is illustrated with a 2D example, the extension to 3D is straightforward.

This approach can be easily generalized for higher than second order reconstruction. As we can see from step 1 to 4, the interpolation occurs only in the coarse faces (or intermediate coarse faces), where high order interpolation can be used without any problem. The only issue with the high order interpolation is that it should be conservative in a finite-volume sense, which involves more computations than the standard high order interpolations.

We have compared our approach with the reconstruction method of Balsara [4] for a refinement of two in each direction. If the fine face values on the coarse faces are constructed in the same way and the limited slope for the intermediate coarse face is the arithmetic averaging of the slopes on the coarse faces [as described by Eq. 3], we obtained exactly the same values for the internal fine faces for both 2D and 3D cases, which implies that our reconstruction method has the same TVD preserving property as Balsara's approach, which was claimed in [5].

The divergence-free prolongation for 3D is different from the reconstruction in Balsara [4]. The reason is that the face values of the new fine grid may come from the old fine grid, and a linear profile is not enough. Balsara [4] introduced complicated interpolation polynomials to match the known fine face values. Our results are different from Balsara's approach [4]. We may not have a closed-form expressions for the prolongation operation on a whole coarse cell. However, whenever the interpolation is needed, monotonicity-preserving linear or high-order interpolation in 1-D is used. It is efficient and TVD-preserving. We acknowledge that our approach may not be as smooth as Balsara's approach within a coarse cell.

For finite-difference and finite-volume method on an AMR grid, a number of ghost cells are usually required. When the number of ghost cells is not divisible by the refinement ratio, special treatment should be applied to maintain that the reconstruction on the coarse face matching the up-to-date values of the old fine grid, since

the fine cell faces share only a part of the coarse cell face. Otherwise, the ghost cell values may not preserve the divergence-free condition. We propose to use a virtual extended grid, where several additional zones are added. For the virtual fine grid, the number of ghost cells is divisible by the refinement ratio so that if a coarse face shares with a fine grid, it is covered wholly by the fine faces. This virtual fine grid is used only to obtain the values of fine grid whenever the grid is re-adapted. It is not used in integration.

2.3 Cylindrical and Spherical Grids

We now discuss how to implement our method in other orthogonal grids such as cylindrical and spherical coordinates with AMR. Balsara [5] extended his divergence-free reconstruction of [4] to the cylindrical and spherical structure meshes. In [5], a variable substitution and coordinate transformation approach is used to transform the divergence formula of the cylindrical and spherical geometry to the standard divergence formula of the Cartesian grid. Then the reconstruction for the Cartesian grid is applied to the new variables on the new coordinates.

In our implementation, we assume the edges of grid in any direction are split into equal-distance pieces according to the refinement ratio. This may not mean bisection of the cell (for refinement ratio of 2) in volume sense in each direction. We use (r,z,ϕ) to represent three directions for cylindrical coordinates, and use (r,θ,ϕ) for spherical coordinates. Then the divergence for a vector field $\mathbf{v}$ becomes

$$\nabla\cdot\mathbf{v}=\frac{1}{r}\left(\frac{\partial(rv_r)}{\partial r}+\frac{r\partial v_z}{\partial z}+\frac{\partial v_\phi}{\partial\phi}\right), \tag{4}$$

for cylindrical grid, and

$$\nabla\cdot\mathbf{v}=\frac{\partial(r^2v_r)}{r^2\partial r}+\frac{\partial(\sin\theta v_\theta)}{r\partial\theta}+\frac{\partial v_\phi}{r\sin\theta\partial\phi}, \tag{5}$$

for spherical grid.

The main complication in implementing AMR in these geometries is to determine where the cell centers and face centers are, because they are all weighted by the additional factors such as $1/r$ and $\sin\theta$. To be concise, we study only the spherical grid. The cylindrical grid can be deduced similarly.

Take a reduced 2-D problem (r,θ) as an example. (The full 3-D case can be derived similarly.) The divergence condition becomes

$$\nabla\cdot\mathbf{v}=\frac{1}{r^2}\frac{\partial(r^2v_r)}{\partial r}+\frac{1}{r\sin\theta}\frac{\partial(\sin\theta v_\theta)}{\partial\theta}. \tag{6}$$

Integrating (6) on cell $[r_{i+1},r_i]\times[\theta_{j-1},\theta_j]$, we obtain

$$\begin{aligned}\nabla\cdot\mathbf{v}=\frac{1}{\Delta V}\Big(&\ (v_{r_i}r_i^2-v_{r_{i-1}}r_{i-1}^2)(\cos\theta_{j-1}-\cos\theta_j)+\\&v_{\theta_j}r_{i-\frac{1}{2}}\sin\theta_j dr-v_{\theta_{j-1}}r_{i-\frac{1}{2}}\sin\theta_{j-1}dr\Big)\end{aligned} \tag{7}$$

where $\Delta V = (\frac{1}{3}r_i^3 - \frac{1}{3}r_{i-1}^3)(\cos\theta_{j-1} - \cos\theta_j)$ (where we drop $d\phi$ in our expressions) is the cell volume. Note that $r_i^2(\cos\theta_{j-1} - \cos\theta_j)$ is the area of the face at $r = r_i$, and $r_{i-\frac{1}{2}}\sin\theta_j dr$ is the area of the face at $\theta = \theta_j$, the discretization is in good agreement with the physical definition of the divergence. At the origin $r = 0$, we have only three faces, and hence we do not need to worry about the value of v_r at the origin. At $\theta = 0$ or $\theta = \pi$, we also have only three faces, and the reconstruction of v_θ there needs special treatment.

We start from the reconstruction on each coarse face. Let v_r denotes the vector field component at the cell's (θ, ϕ) face, v_θ denotes the vector field at the cell's (r, ϕ) face. A piecewise linear profile via a slope limiter for v_r and v_θ can be constructed in the same way as for Cartesian grid. Assume the limited slope for v_r in θ direction is $v_{r,\theta}$, and the linear profile is

$$v_r = v_{r_c} + v_{r,\theta}(\theta - \theta_0), \tag{8}$$

where θ_0 is the face center for the coarse face. Unlike in Cartesian

$$\theta_0 \neq \frac{1}{2}(\theta_j + \theta_{j-1})$$

in spherical coordinates. If we let v_{r_c} denote the value of v_r on the coarse face, then the following condition should be satisfied

$$\frac{1}{r(\cos\theta_{j-1} - \cos\theta_j)} \int_{\theta_{j-1}}^{\theta_j} v_r \cdot r\sin\theta d\theta = v_{r_c},$$

which yields

$$\theta_0 = \frac{\theta_{j-1}\cos\theta_{j-1} - \theta_j\cos\theta_j + \sin\theta_j - \sin\theta_{j-1}}{\cos\theta_{j-1} - \cos\theta_j}.$$

Similarly, for $v_{\theta,r}$ and linear profile

$$v_\theta = v_{\theta_c} + v_{\theta,r}(r - r_0), \tag{9}$$

where r_0 is the center of the cell face, we have

$$\int_{r_{i-1}}^{r_i} (r - r_0) r\sin\theta dr = 0,$$

which yields

$$r_0 = \frac{2(r_i^3 - r_{i-1}^3)}{3(r_i^2 - r_{i-1}^2)}.$$

We should mention that the formula for θ_0 and r_0 are also valid for the face on a fine grid. Therefore, when the mesh is refined with ratio m, face centers for the fine mesh can be calculated in the same fashion. Let $\delta r = (r_i - r_{i-1})/m$ and $r = r_i$, then the face centers for the fine grid will be

$$r_{0l} = \frac{2[(r-l\delta r)^3 - (r-(l+1)\delta r)^3]}{3[(r-l\delta r)^2 - (r-(l+1)\delta r)^2]}, \quad l = 0, 1, \ldots, m-1.$$

After all the face centers are calculated, the vector field components can be calculated by the linear profile (8) and (9). It is easy to verify that the total flux on each coarse face is conserved after interpolation.

In case of $dA_r = 0$ (e.g., $\theta = 0$ or $\theta = \pi$), we let $dA_r = \Delta r$, which is reduced to the reconstruction on a uniform 1-D grid. If the new fine grid shares face with the old fine grid, we do not need to construct the linear profile for the coarse face, and only copy the values of the old fine grid to the new grid.

For cylindrical grid, the corresponding face centers are

$$z_0 = z_{j-\frac{1}{2}} = \frac{1}{2}(z_j + z_{j-1}) \tag{10}$$

$$\phi_0 = \phi_{k-\frac{1}{2}} = \frac{1}{2}(\phi_k + \phi_{k-1}) \tag{11}$$

$$r_0 = \frac{2(r_i^3 - r_{i-1}^3)}{3(r_i^2 - r_{i-1}^2)}. \tag{12}$$

After the face reconstruction is finished, the algorithm described in section 2.2 can be used to reconstruct the internal faces step by step.

3 Numerical Experiment

We have implemented our algorithm in our MHD AMR solver [9] to preserve the divergence free condition of the magnetic field. We also implemented Balsara's electrical force correction method [4] to ensure that the restriction from the fine grid to the coarse grid preserves the divergence free condition of the magnetic field. Here we define the magnetic components on the face center while all the other fluid quantities are still defined at cell centers. A second order Roe's Riemann solver is used to advance the conservative variables.

3.1 Rotor Problem

The first example is taken from [2]. There is a dense rotating disk of fluid with density 10, angular velocity 20, and radius 0.1. The ambient fluid is at rest with density 1. The initial magnetic field is uniform with $B_x = 5/\sqrt{4\pi}$ and $B_y = 0$.

We first solved this example in a Cartesian grid with domain $[0,1] \times [0,1]$, base grid 100×100 and three-level refinement with ratios of 3 and 2. The results are shown in Fig.2. The divergence-free condition is preserved very well.

We also solved this example as a a reduced 2-D problem in (r, ϕ) with a disk domain $[0, 0.6] \times [0, 2\pi]$, where x is along $\phi = 0$ direction. Converting B_x and B_y into cylindrical coordinates, we have

$$B_r = B_x \cos\phi, \qquad (13)$$
$$B_\phi = -B_x \sin\phi,$$

is not divergence-free in our finite-volume discretization. Therefore, we adopted an approximate initialization for B_r, which is

$$B_r = B_x \frac{\sin\phi_{j+1} - \sin\phi_j}{d\phi}.$$

This problem has two challenges for preserving the divergence-free condition on an AMR grid. The first is the singular point at the origin. For the B_r component, although it is not used in our finite-volume discretization for calculating $\nabla \cdot \mathbf{B}$ (it is canceled due to the zero area at $r = 0$), it is needed in calculating the cell-centered values of B_r, which is then used in the Riemann solver. To calculate the $\frac{dB_r}{dt}$ at the origin, we use extrapolation based on the cell-centered values of $\frac{dB_r}{dt}$ at $(\phi, \frac{1}{2}dr)$ (which is calculated by our Riemann solver) and the values at $r_1 = dr$.

For obtaining the B_ϕ component at $r = 0$, the Riemann solver produces different electric-field values for different ϕ at the origin. To maintain the divergence-free condition, only one electric-field value at the origin should be used to advance the B_ϕ for all of the ϕs. We set the electric-field at the origin to be the average over the whole circle.

The next challenge is the periodic boundary condition in ϕ. Every patch that shares an edge with $\phi = 0$ or $\phi = 2\pi$ can become potentially an electric-field correction partner for a coarse patch on the other end. It is important to make sure that the correction does only once for each cell, and the B_ϕ and the electric-field components have the same values (up to the round-off error) at $\phi = 0$ and $\phi = 2\pi$.

We used the same three-level refinement with ratio 3 and 2 as in Cartesian grid. The base grid is 60×120. As suggested in [2], we output the solution at the final time $t = 0.295$. Fig. 2 shows the results.

3.2 Magnetized Jet Problem

The next example is introduced by [10]. This is a simulation of a light cylindrical MHD jet with a top-hat velocity profile. We tested this problem with computation domain $[0,1] \times [0,2]$ in the cylindrical geometry with (r,z) coordinates. The base grid is 200×200. The jet has a radius 0.125, which is about 25 base grid cells. The ambient medium has sound speed of 1, and poloidal magnetic field ($B_\phi = B_r = 0$, $B_z = 0.1$). The jet has Mach number of 20, gas density contrast $\rho_{jet}/\rho_{\text{ambient}} = 0.1$. The jet carries a helical magnetic field with $B_r = 0, B_\phi = 2B_{\text{ambient}}(r/r_{jet})$, and $B_z = B_{\text{ambient}}$.

The same numbers of refinement levels and ratios as the previous example are used. We ran our test until $t = 0.1$. Fig. 3.3-a shows the density contour plot with the refinement. Fig. 3.3-b shows the results of $\nabla \cdot \mathbf{B}$.

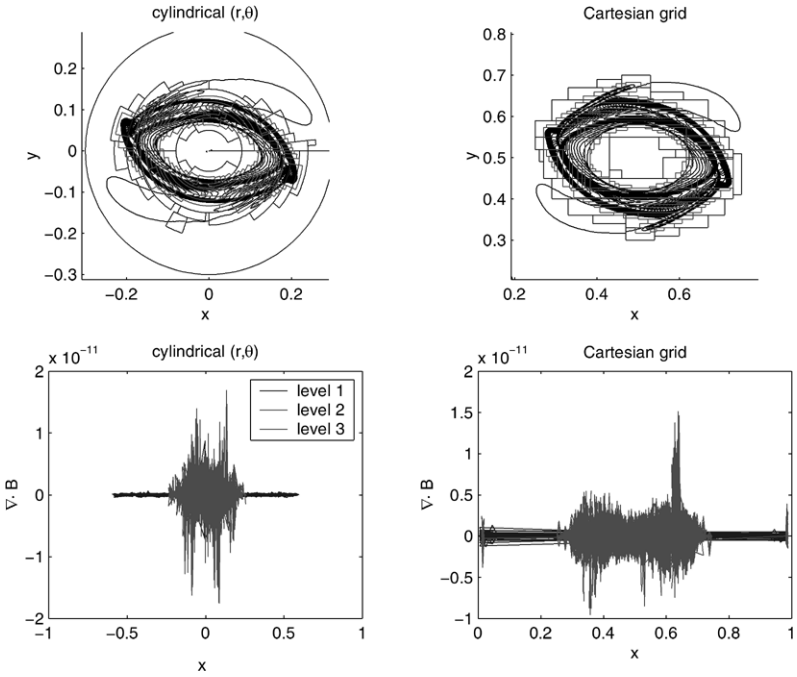


Fig. 2. The results for rotor problem at $t = 0.295$. The refinement ratios are 3 and 2 respectively. 30 contour lines between 0.532 and 10.83 are used.

3.3 Spherical Bubble Problem

The last example is a spherical hot light bubble resided in an equilibrium gas. The bubble will rise up due to the pressure imbalanced at the interface. The de-

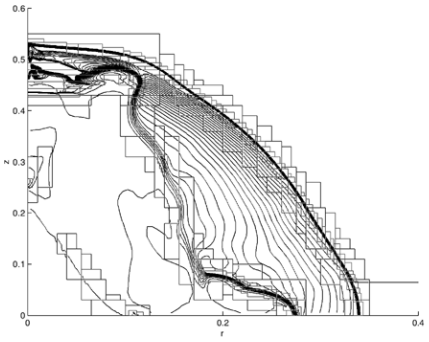


Fig. 3.3-a. The "zoomed" version of refinement and density contour plot for the jet problem on cylindrical (r, z) plane.

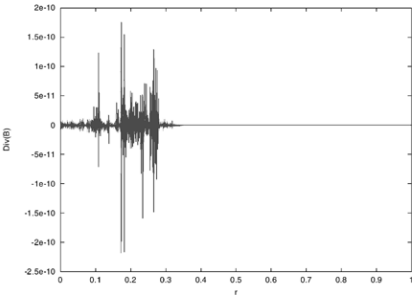


Fig. 3.3-b. The $\nabla \cdot \mathbf{B}$ for Fig. 3.3-a.

tail of the problem set up is described in [9]. The computational domain in (r,θ) is $[0.2,3.8]\times[0,\pi/3]$. The initial gas subject to the external gravity is in equilibrium state with density and pressure $\rho = p = \exp(-r)$. The spherical bubble located at center (1.1,0) with radius 0.5. The density inside the bubble is defined as $\rho = 0.1\exp(-r)$. We had solved the problem with no magnetic field and observed the Rayleigh-Taylor instability at the contact interface. The MHD test is done with a uniform magnetic field in $z = r\cos\theta$ direction with a potential $F = -bz$. The constant b controls the magnitude the magnetic field, which is defined by

$$\mathbf{B} = -\nabla F = (b\cos\theta, -b\sin\theta).$$

We tested the bubble problem with $b = 0.4$. The Rayleigh-Taylor instability at the contact interface is suppressed by the strong magnetic field.

We used a base grid of 360×120 and refinement ratio of 3. The density contour plots with refinements are shown in Fig. 3.4-a. The divergence of the magnetic field at different times is shown in Fig. 3.4-b.

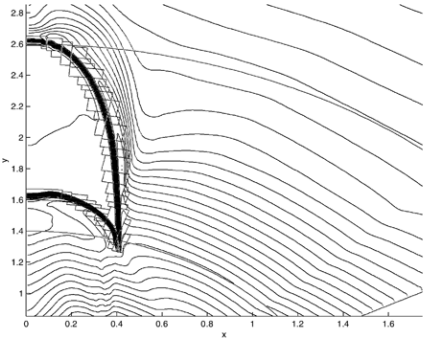


Fig. 3.4-a. The density contour plot with refinement for bubble problem with strong magnetic field $b = 0.4$. Ratio is 3.

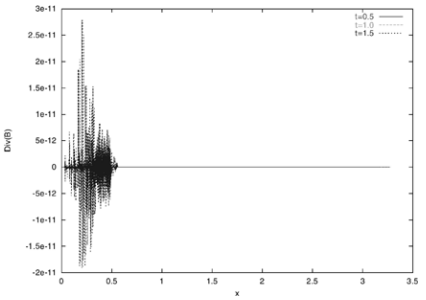


Fig. 3.4-b. The $\nabla\cdot\mathbf{B}$ at different times.

References

1. M. J. Berger and P. Colella, Local adaptive mesh refinement for shock hydrodynamics, *J. Comput. Phys.* 82 (1989), 64-84.
2. D. S. Balsara and D. S. Spicer, A staggered mesh algorithm using high order Godunov fluxes to ensure solenoidal magnetic fields in magnetohydrodynamics simulations, *J. Comput. Phys.* 149 (1999), 270-292.
3. J. U. Brackbill and D. C. Barnes, The effect of nonzero $\nabla\cdot B$ on the numerical solution of the magnetohydrodynamic equations, *J. Comput. Phys.*, 35 (1980), 426.
4. D. S. Balsara, Divergence-free adaptive mesh refinement for magnetohydrodynamics, *J. Comput. Phys.*, 174 (2001), 614-648.

5. D. S. Balsara, Second order accurate schemes for magnetohydrodynamics with divergence-free reconstruction, astro-ph/0308249, (2003).
6. W. Dai and P. R. Woodward, A simple finite difference scheme for multidimensional magnetohydrodynamics, *J. Comput. Phys.* 142 (1998), 331-369.
7. C. R. Evans and J. F. Hawley, Simulation of magnetohydrodynamic flows: A constrained transport method, *Astrophys. J.* 332 (1989), 659.
8. P. Londrillo and L. Del Zanna, High-order upwinding schemes for multidimensional magnetohydrodynamics, *Astrophysics, J.* 530 (2000), 508.
9. S. Li and H. Li, A modern code for solving magnetohydrodynamics or hydrodynamic equations, Technical Report, Los Alamos National Laboratory, 2003.
10. D. Ryu, F. Miniati, T. W. Jones, and A. Frank, A divergence-free upwinding code for multi-dimensional MHD flows, *Astrophysics. J.*, 509 (1998), 244-255.
11. G. Tóth, The $\nabla \cdot B = 0$ constraint in shock-capturing magnetohydrodynamics codes, *J. Comt. Phys.*, 161 (2000), 605-652.
12. G. Tóth and P. L. Roe, Divergence- and curl-preserving prolongation and restriction formulas, *J. Comt. Phys.*, 180 (2002), 736-750.
13. K. S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equation in an isotropic media, *IEEE Trans. Antenna Propagation* AP-14 (1966), 302

Multiresolution-based adaptive schemes for Hyperbolic Conservation Laws

Guillaume Chiavassa[1], Rosa Donat[2], and Siegfried Müller[3]

[1] LATP and EGIM, Technopôle de Château-Gombert, 13383 Marseille Cedex 13, France
guillaume.chiavassa@egim-mrs.fr
[2] Departamento de Matematica Aplicada, Universidad de Valencia, 46100 Burjassot (Valencia), Spain donat@uv.es
[3] Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 52056 Aachen, Germany mueller@igpm.rwth-aachen.de

Summary. Starting in the early nineties, wavelet and wavelet-like techniques have been successfully used to design adaptive schemes for the numerical solution of certain types of PDE. In this paper we review two representative examples of the development of such techniques for Hyperbolic Conservation Laws.

1 Introduction

Solutions to systems of Hyperbolic Conservation Laws (HCL henceforth) tend to be highly nonuniform in their spatial behavior. Large regions of smooth, slowly varying behavior are separated by highly localized transition regions of non-smooth behavior. In the context of HCLs, singularities and sharp transition regions model such important physical phenomena as the formation and evolution of shock waves.

There is a general agreement that shock related phenomena need an adequate numerical treatment that usually involves a considerable increase in computational resources. Fine grids are necessary in order to resolve adequately regions of strong variation, but global uniformity in the computational meshes necessarily implies that the solution is over-resolved in smoothness regions, usually the largest part of the computational domain.

The need to solve realistic problems has motivated the development of adaptive techniques, for which the computational effort concentrates near regions where singularities or sharp transitions occur.

Classical Adaptive Mesh Refinement (AMR) techniques [BO84, BC89] rely on a sequence of nested grids of increasing resolution and on certain error estimators that seek to determine locally whether the current resolution of the numerical solution is sufficient or a finer grid is necessary. These techniques involve a considerable effort in programming and data management but are now routinely used in realistic simulations.

In recent years, the development of the theory of wavelets has provided an additional tool to design numerical schemes that seek to adapt the computational resources to the local structure of the solution to be computed.

Working within a Galerkin framework, Liandrat and Tchamitchian [LT] developed a numerical scheme in which adaptive refinement is implemented by adding layers of successive "details" that locally increase the resolution of the approximation. Time adaptivity was incorporated in [BMP92], where the authors present a wavelet-based numerical method for hyperbolic and parabolic PDEs that adapts the space and time resolutions to the properties of the PDE and the local structure of the solution.

Being translates and dilates of a single function, wavelet bases are often too 'rigid' for certain applications. Indeed, in [LT, BMP92] only periodic problems for scalar hyperbolic and parabolic equations in one dimension are handled, mainly because of various technical difficulties related to the wavelet basis considered. Even though some of the problems related to the poor representation of boundaries by wavelet basis have been addressed in recent years (see [Coh03, Dah97] for good reviews on wavelet methods for PDEs), successful multiresolution-based adaptive techniques for HCL have followed instead the path laid out by A. Harten in his seminal work [Har95].

In the early 90's, A. Harten developed a general framework for multiresolution (MR) that exhibits a larger degree of flexibility, while retaining many of the properties associated to wavelet-decompositions. A distinctive feature of Harten's framework is that a discrete data set is always interpreted as the result of the application of a particular discretization operator on a function belonging to an appropriate functional space. This feature is well suited for computing solutions to PDEs by numerical techniques, since the values obtained are interpreted as discrete realizations of the solution on a computational mesh.

A MR decomposition of a discrete realization of a given function gives precise information on the local regularity of that function. Harten's adaptive strategy is based on the smoothness information contained on an appropriate MR decomposition of the numerical data obtained, at each time step, by an underlying numerical scheme.

In its simplest implementation the goal of the MR-based adaptive scheme is, essentially, to gain computational time while remaining within the same accuracy as the *reference* scheme, i.e. the scheme on the finest computational mesh for which the user is pleased with the computational results. Successful implementations of this strategy have been carried out for two-dimensional Cartesian meshes [BH95, BH97, CD01], curvilinear meshes [DGM00] and unstructured meshes [Abg97, CDKP00, BOLR01].

A more elaborate implementation has been developed in [CKMP03]. Here, the MR decomposition of the numerical data is used to reduce not only the computational cost, but also the memory requirements of the computation, while remaining within the same accuracy as the *reference* scheme. The implementation in [CKMP03] is indeed a spatial AMR technique, in which adaptive refinements are based on the smoothness information obtained from the MR decomposition of the data.

In this paper we shall review these two main directions in which Harten's methodology for the design of adaptive schemes for HCLs has evolved during the last ten years. The paper is organized as follows: In section 2 we briefly describe the essential ingredients of Harten's framework for MR, with particular attention to the interpolatory and cell-average frameworks. Section 3 describes the basic strategy of MR-based adaptive schemes. In section 3.1 we briefly review the cost-reduction implementation and in section 3.2 the fully adaptive one. We close in section 4 with a summary.

2 Data Representation and Multiscale Analysis

The numerical values obtained by a given numerical scheme are understood as approximations to a discrete realization of the solution on an underlying computational mesh. When using a sufficiently robust scheme, these numerical values reflect in one way or another the behavior of the true solution. In those regions where the solution is smooth, the discrete data displays a 'smooth discrete behavior'. At shocks and/or contact discontinuities, the discontinuous behavior of the true solution is represented by a sharp profile. In fact, the robustness of the scheme is often measured by the ability to represent a shock transition as a sharp, oscillation-free, discrete profile.

Smoothness regions can be handled with rather unsophisticated (and non-expensive) numerical techniques, while compression regions and shocks require a very specialized numerical treatment. A multiscale decomposition of the numerical solution at each time step can provide the necessary information about the local smoothness of the underlying data to allow for an adaptive computation.

In a MR representation of a discrete data set the information is encoded as a coarse realization of the given data set plus a sequence of detail coefficients of ascending resolution. The detail (scale, wavelet) coefficients represent the *difference in information* between consecutive resolution levels.

In the following we give a brief overview of the core ingredients of Harten's MR concept namely: (i) a sequence of nested discretization operators and (ii) a sequence of *consistent* reconstruction operators.

Discretization and Decimation

For the application that we have in mind, the resolution levels are specified by a sequence of computational meshes $\mathcal{G}_l = \{\Omega_{l,k}\}_{k\in J_l}$, $l = 0,\ldots,L$ on a domain Ω. The index l represents the resolution level (increasing l means more resolution). The starting point is a sequence of linear vector spaces of discrete data, V_l, $l \in \mathbb{N}_0$, related to $\mathcal{G}_l$ via a particular *discretization operator*.

The discretization operators are linear operators acting on a function space $\mathcal{D}_l : \mathcal{F} \to V_l$ ($\mathcal{F}$ is usually the solution space for the PDE). Given $u \in \mathcal{F}$, $\mathcal{D}_l u$ assigns a discrete value to each $\Omega_{l,k} \in \mathcal{G}_l$,

$$\bar{u}^l_k = (\mathcal{D}_l u)_k =: \mathcal{A}(\Omega_{l,k})u \qquad \forall \Omega_{l,k} \subset \mathcal{G}_l. \tag{1}$$

The grid *elements*, $\Omega_{l,k}$ can be either grid-points or mesh-cells (structured or unstructured). The chosen notation reflects the fact that we think of the discretization

as an *averaging operator*. For point-value discretizations, the *elements* of the grid are its nodes and the averaging is done with respect to Dirac's delta function. For cell-average discretizations, the *elements* of the mesh are the cells (quadrilateral, triangles, hexahedra) and the averaging is done with respect to the indicator function of each cell.

To obtain multiscale decompositions associated to a sequence of meshes $\{\mathcal{G}_l\}$ on a domain $\Omega \subset \mathbb{R}^d$, the associated discretization operators $\mathcal{D}_l$ have to satisfy two essential properties

1. $\mathcal{D}_l$ is onto.
2. The null spaces satisfy $\mathcal{N}(\mathcal{D}_l) \subset \mathcal{N}(\mathcal{D}_{l+1})$.

Property 2 gives the sequence $\{\mathcal{D}_l\}$ a *nested* structure. For our intended application, this property derives from the structure of the averaging operator and a nested structure in the sequence $\{\mathcal{G}_l\}$, i.e. each cell $\Omega_{l,k}$ on level l can be subdivided into subcells $\Omega_{l+1,r}$ on the finer level $l+1$ so that

$$\Omega_{l,k} = \cup_{r \in \mathcal{M}_{l,k}} \Omega_{l+1,r}. \tag{2}$$

As a consequence of properties 1 and 2 we can associate to the sequence $\{\mathcal{D}_l\}$ a sequence of *decimation operators*, by which coarse data is obtained from fine data [Har96]: If $u \in \mathcal{F}$ and $\bar{u}^m = \mathcal{D}_m u$ for each m, then $\bar{u}^l = D^l_{l+1}\bar{u}^{l+1}$ for all l. For local averaging operators we obtain

$$D^l_{l+1} : V_{l+1} \longrightarrow V_l \qquad (D^l_{l+1}\bar{u}^{l+1})_k = \sum_{r \in \mathcal{M}_{l,k}} m^{l,0}_{k,r}\bar{u}^{l+1}_r, \quad k \in J_l \tag{3}$$

where the coefficients $m^{l,0}_{k,r}$ depend *only* on the sequence $\{\mathcal{D}_l\}$.

Reconstruction and Prediction To represent the difference between discrete values on two consecutive resolution levels, a sequence of *consistent* reconstruction operators is introduced,l $\mathcal{R}_l : V_l \to \mathcal{F}$. Consistency means that

$$\mathcal{D}_l \mathcal{R}_l \bar{u}^l = \bar{u}^l \qquad \forall \bar{u}^l \in V_l. \tag{4}$$

Notice that if $\bar{u}^l = \mathcal{D}_l u$, $u \in \mathcal{F}$, then $\mathcal{R}_l\bar{u}^l$ is interpreted as an approximation to u. In general $u \neq \mathcal{R}_l\bar{u}^l$, but (4) implies that u and $R_l\bar{u}^l$ have the same discrete information on $\mathcal{G}_l$.

In our context, $\mathcal{R}_l\bar{u}^l$ is constructed as follows: For each element $\Omega_{l,k} \in \mathcal{G}_l$ we determine a polynomial $R^N_{l,k} : \Omega \to \mathbb{R}$ of degree N by imposing the *recovery conditions*

$$\mathcal{A}(\Omega_{l,r})R^N_{l,k}(\cdot, \bar{u}^l) = \bar{u}^l_r, \qquad r \in \mathcal{S}_{l,k}. \tag{5}$$

Here $\mathcal{S}_{l,k}$, the *stencil* of $R^N_{l,k}$, denotes an index set on level l corresponding to cells in a local neighborhood of $\Omega_{l,k}$ with $\Omega_{l,k} \in \mathcal{S}_{l,k}$ and such that it provides an *admissible configuration*, i.e. (5) leads to a uniquely solvable $R^N_{l,k}$.

The reconstruction operators are then defined by piecing together these polynomial reconstructions. If the elements are mesh-cells, then

$$\mathcal{R}_l \bar{u}^l(x) = R^N_{l,k}(x;\bar{u}^l), \qquad x \in \Omega_{l,k}. \tag{6}$$

Notice that consistency follows from the fact that k belongs to $\mathcal{S}_{l,k}$.

Remark The linearity of $\mathcal{D}_l$ and the recovery conditions imply that the reconstruction process has polynomial exactness of degree N, i.e.

$$R^N_{l,k}(\cdot,\bar{p}^l) = p(\cdot) \qquad \forall p \in \mathcal{P}_N \quad \text{with} \quad \bar{p}^l = \mathcal{D}_l p. \tag{7}$$

The sequences $\{\mathcal{R}_l\}$, $\{\mathcal{D}_l\}$ are used to define prediction operators, $P^l_{l+1} : V_{l+1} \to V_l$, that compute fine grid values from coarse grid values. These are defined as $P^l_{l+1} = \mathcal{D}_{l+1}\mathcal{R}_l$; taking (2) into account we have

$$\tilde{u}^{l+1}_r := (P^{l+1}_l \bar{u}^l)_r = \mathcal{A}(\Omega_{l+1,r}) R^N_{l,k}(\cdot,\bar{u}^l), \quad r \in \mathcal{M}_{l,k}, \quad l \in \mathbb{N}_0. \tag{8}$$

The prediction errors, computed as $e^{l+1}_k := \bar{u}^{l+1}_k - \tilde{u}^{l+1}_k$, $k \in J_{l+1}$, contain the information in $\bar{u}^{l+1}$ which cannot be predicted from the coarser data $\bar{u}^l = D^l_{l+1}\bar{u}^{l+1}$ by the prediction scheme P^l_{l+1}. These errors represent, hence, the difference in information between consecutive resolution levels.

Clearly, the discrete sets $\bar{u}^{l+1}$ and $(\bar{u}^l = D^l_{l+1}\bar{u}^{l+1}, e^{l+1} = \bar{u}^{l+1} - P^{l+1}_l \bar{u}^l)$ are equivalent. However there is an inherent redundancy in the information contained in e^{l+1}. In fact $D^l_{l+1} e^{l+1} \equiv 0$, see [Har96], i.e.

$$(D^l_{l+1} e^{l+1})_k = \sum_{r \in \mathcal{M}_{l,k}} m^{l,0}_{k,r} e^{l+1}_r = 0, \qquad k \in J_l \tag{9}$$

A non-redundant two scale representation of $\bar{u}^{l+1}$, can be obtained by defining the *scale coefficients*, or *detail coefficients*, as the coordinates of the prediction error expressed in a basis of $\mathcal{N}(D^l_{l+1})$, the null space of D^l_{l+1}. In [DGM00], following the guidelines of the standard theory in wavelet-type multiscale decompositions, the authors resort to the concept of *stable completions* to provide a working definition of the scale coefficients.

Notice that for each $\Omega_{l,k}$ there are $\#\mathcal{M}_{l,k}$ non-independent prediction errors. Because of (9), these can be adequately represented by $\#\mathcal{M}_{l,k} - 1$ independent quantities: the *scale coefficients*. When $\mathcal{S}_{l,k}$ is chosen independently of k, one finds that

$$d^l_k = \sum_{r \in \mathcal{M}_{l,k}} m^{l,1}_{k,r} e^{l+1}_r, \qquad k \in \mathcal{J}_{l,k} \tag{10}$$

where $\mathcal{J}_{l,k}$ is an index set associated to $\Omega_{l,k}$ and such that $\#\mathcal{J}_{l,k} = \#\mathcal{M}_{l,k} - 1$.

A one-to-one two-scale representation of a data set $\bar{u}^{l+1}$ is obtained by considering $\bar{u}^l = D^l_{l+1}\bar{u}^{l+1}$ and d^l_k in (10).

Let $\bar{u}^L = \mathcal{D}_L u$ for $u \in \mathcal{F}$, where L is the index of a sufficiently fine mesh $\mathcal{G}_L$. Repeating the previous process for the grid hierarchy $\{\mathcal{G}_l\}_{l=0}^L$ we obtain a multiscale decomposition of $\bar{u}^L$.

$$u^L \Leftrightarrow \{u^{L-1}, d^{L-1}\} \cdots \Leftrightarrow \cdots \{u^0; d^0; d^1; \cdots \quad d^{L-1}\} = M u^L$$

$$\begin{array}{ccccccccc} u^L & \to & u^{L-1} & \to & u^{L-2} & \to & \ldots & \to & u^0 \\ & \searrow & d^{L-1} & \searrow & d^{L-2} & \searrow & \ldots & \searrow & d^0 \end{array} \tag{11}$$

Let us consider $u \in \mathcal{F}$ and $\bar{u}^L = \mathcal{D}_L u$. Relations (1) and (8) lead to $\Omega_{l,k} = \cup_{r \in \mathcal{M}_{l,r}} \Omega_{l+1,r}$,

$$e_r^{l+1} = \bar{u}_r^{l+1} - \tilde{u}_r^{l+1} = \left(\mathcal{D}_{l+1}(u - R_{l,k}^N(\cdot, \bar{u}^l)) \right)_r, \qquad r \in \mathcal{M}_{l,k}. \tag{12}$$

Notice that for any $p \in \mathcal{P}_N$, (12) and (7) imply that for $r \in \mathcal{M}_{l,k} \subset J_{l+1}$

$$e_r^{l+1} = \mathcal{A}(\Omega_{l+1,r})(u - p) - \mathcal{A}(\Omega_{l+1,r}) R_{l,k}^N(\cdot, \mathcal{D}_l(u - p)). \tag{13}$$

Since smooth functions are well approximated by polynomials, the above relation shows that prediction errors are expected to be small in regions of smoothness. Because of (10) the same will hold for the scale coefficients d_k^l. It is precisely because of this relation that MR decompositions can be considered as a tool to adapt computational refinements to the local regularity of the solution.

In numerical simulations for time dependent HCLs, the starting point at each time step is a discrete data set $u^L = \{u_k^L\}_{k=1}^{J_L}$ so that each value u_k^L is linked to a particular *element* of an underlying mesh $\mathcal{G}_L$. The grid *elements* can be either mesh cells, as in standard finite volume schemes, but also mesh points as in the ENO numerical schemes considered in [SO88]. Within Harten's framework for MR, each interpretation (cell-averages or point-values) has an associated *natural* MR setting. As an example, we briefly review next the point-value MR setting.

2.1 Interpolatory MR

In the point-value framework for MR the grid elements are the mesh nodes. In what follows, it will be convenient to be a bit more specific in our description. For this, we consider a sequence of uniform nested grids on $[0,1]$ $\mathcal{G}_l = \{x_k^l\}_{k=0}^{J_l}$, obtained by recursive dyadic refinement of $\mathcal{G}_0$, which we consider as the coarsest resolution level associated to our underlying problem. Hence $\Omega_{l+1,2k} = x_{2k}^{l+1} = x_k^l = \Omega_{l,k}$ (see Fig. 2.1-left), $h_l = 2^{-l} h_0$, $J_l = 2^l J_0$.

Structured Cartesian meshes in 2D or 3D are obtained by a tensor product construction using these 1D meshes (e.g. [BH97]), see Fig. 2.1-right. The extension to unstructured meshes is conceptually straightforward within Harten's framework for MR, although its particular application requires a certain degree of familiarity with the technical aspects of using unstructured meshes for the solution of PDEs (see [AH98, Abg97] for specific details).

The transfer of information from fine to coarse is done by retaining those values attached to points in $\mathcal{G}^l \subset \mathcal{G}^{l+1}$ and discarding the rest. In the 1D case of Fig. 2.1-left, the *decimation by restriction* process acts as follows, $u_k^l = (D_{l+1}^l u^{l+1})_k = u_{2k}^{l+1}$. For the case 2D of Fig. 2.1-right we have $u_{k,m}^l = (D_{l+1}^l u^{l+1})_{k,m} = u_{2k,2m}^{l+1}$.

Decimation by restriction has a natural correspondence with the interpretation of the discrete sets as being the values of an underlying function $u(x)$ at the mesh points

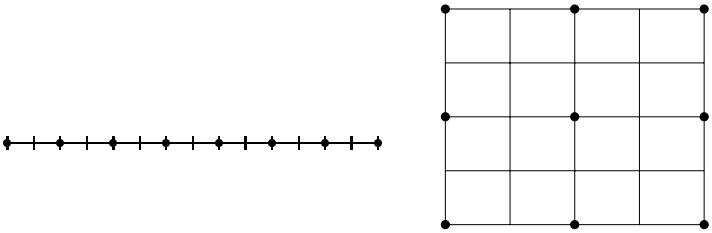


Fig. 1. +: mesh points in $\mathcal{G}^{l+1}$; •: Mesh points in $\mathcal{G}^l$. Left 1D, right 2D uniform meshes. Detail coefficents correspond to prediction errors at points in $\mathcal{G}^{l+1} - \mathcal{G}^l$.

of each grid, i.e. if $\mathcal{D}_m u = u(\mathcal{G}_m)$, $\forall m$, then $u^l = D^l_{l+1} u^{l+1} = D^l_{l+1} \mathcal{D}_{l+1} u = u(\mathcal{G}_l) = \mathcal{D}_l u$.

Given a grid element $x^l_k \in \mathcal{G}_l$, let us consider the polynomial $L^N_{l,k}$ characterized by the recovery conditions

$$L^N_{l,k}(x^l_r; u^l) = u^l_r, \qquad r = k-p, \ldots, k+p+1 \tag{14}$$

where $p \geq 0$ is a fixed integer, chosen independently of l and k. Notice that $L^N_{l,k}(x; u^l)$ is the Lagrange interpolation polynomial of degree $N = 2p+1$ based on the stencil $\mathcal{S}_{l,k} = \{x^l_r\}^{k+p+1}_{r=k-p}$ and the data attached to it.

Following (6) we have (see also Fig. 2)

$$\mathcal{R}_l u^l(x) := L^N_{l,k}(x; u^l), \qquad x \in [x^l_k, x^l_{k+1}], \quad k = 0, \ldots J_l - 1 \tag{15}$$

and following (8) we get the *prediction* operator, which obtains fine data from coarse data, $\tilde{u}^{l+1}_m = (P^{l+1}_l u^l)_m = L^N_{l,[m/2]}(x^{l+1}_m, u^l)$. Hence

$$\tilde{u}^{l+1}_{2k} = L^N_{l,k}(x^{l+1}_{2k}, u^l) = L^N_{l,k}(x^l_k, u^l) = u^l_k \tag{16}$$

$$\tilde{u}^{l+1}_{2k+1} = L^N_{l,k}(x^{l+1}_{2k+1}, u^l) = \sum_{r=1}^{p} \beta_l (u^l_{k+r} + u^l_{k-r+1}) \tag{17}$$

where the coefficients β_m depend on N (see e.g. [Har96]).

Since $u^{l+1}_{2k} = u^l_k$, (16) implies that $e^l_{2k} = 0$. The prediction error for the odd values will not be zero in general, hence in this setting the scale coefficients are defined as $d^l_k := u^{l+1}_{2k+1} - \tilde{u}^{l+1}_{2k+1}$. The invertible two-scale transformation is ($k = 0, \ldots J_l - 1$)

$$\left\{ \begin{array}{l} u^l_k = u^{l+1}_{2k} \\ d^l_k = u^{l+1}_{2k+1} - \tilde{u}^{l+1}_{2k+1} \end{array} \right\} \quad \leftrightarrow \quad \left\{ \begin{array}{l} u^{l+1}_{2k} = u^l_k \\ u^{l+1}_{2k+1} = \tilde{u}^{l+1}_{2k+1} + d^l_k \end{array} \right\}. \tag{18}$$

Notice that the two discrete sets u^{l+1} and $\{(u^l, d^l)\}$, which have exactly the same cardinality, are absolutely equivalent.

The interpretation of a multiscale decomposition within the point-value setting is straightforward: The information contents of the sequence u^L, which is understood as the point-values of $u(x)$ on the (fine) grid $\mathcal{G}_L$, is decomposed as u^0, those same

values but on a much coarser mesh plus a sequence of *scale* coefficients at each resolution level between $\mathcal{G}_0$ and $\mathcal{G}_L$. The scale coefficients d_k^l represent the difference in information between discretizations of the function at two consecutive resolution levels, or in other words, the information on level l that cannot be predicted from the coarse values one level below. In this framework the scale coefficients are simply interpolation errors.

Remark The centered-stencil construction for $L_{l,k}^N$ was used in [DD89] within the context of binary subdivision schemes, which were the basis of Donoho's construction of the Interpolatory Wavelet Transform (IWT) [Don92]. The IWT has been used to design adaptive schemes for scalar HCLs independently by Homlström [Hol99]

The relation between scale coefficients and smooth behavior. Let us assume that the discrete data $u^L = u(\mathcal{G}_L)$, where $u(x)$ is a piecewise smooth function. Figure 2 shows the piecewise polynomial interpolatory reconstruction (15) of a piecewise smooth function. We clearly observe that the quality of the approximation is degraded around the singularity, and that the region affected by the singularity is larger for higher degree polynomials.

Fig. 2. A piecewise smooth signal (solid line) and its point-values on a uniform grid (dots on solid line). Interpolatory reconstructions (Dotted lines). Left: $p = 1, N = 3$. Right $p = 2, N = 5$.

The scale coefficients in the point-value setting are simply interpolation errors, hence the relation between the behavior of the scale coefficients with respect to the regularity of u can be analyzed using elementary interpolation theory. A precise estimate of the size of the scale coefficients, can be given by expressing the error in Lagrange interpolation in its so-called *Newton form*

$$u(x) = L_{l,k}^N(x) + u[\mathcal{S}_{l,k}, x] w_{l,k}(x) \qquad x \in [x_k, x_{k+1}] \tag{19}$$

where $u[\mathcal{S}_{l,k}, x]$ denotes the $N+1$st divided difference of $u(x)$ at the points of the stencil and x and $w_{l,k}(x) = \Pi_{x_m^l \in \mathcal{S}_{l,k}}(x - x_m^l)$.

Since $d_k^l = u[\mathcal{S}_{l,k}, x_{2k+1}^{l+1}] w(x_{2k+1}^{l+1})$, the relation between the smoothness of a function and the behavior of the scale coefficients can be obtained by considering the behavior of the divided differences with respect to singularities of the function. Let us assume that $u^{(s)}$ has a jump discontinuity in the convex hull of $\mathcal{S}_{l,k}$, while $u^{(m)}$ is smooth $0 \leq m < s$. Then it is shown in [Har95] that

$$d_k^l = 2^s d_{2k+i}^{l+1}, \qquad i = 0, 1.$$

Notice that the decay rate of the scale coefficients is intimately related to the smoothness of the underlying function and the uniform coarsening of the hierarchy of meshes. In particular, coefficients belonging to a region of smooth variation of $u(x)$ should approximately satisfy

$$d_k^l \approx 2^{N+1} d_{2k+i}^{l+1}, \qquad i = 0, 1 \tag{20}$$

so that any deviation from this behavior can be interpreted as lack of smoothness. This observation, which will be revisited later, lies at the heart of Harten's heuristics in designing adaptive multiresolution schemes for HCLs.

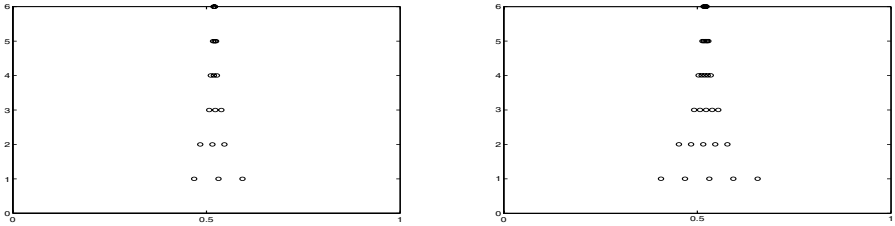


Fig. 3. Scale coefficients larger than 10^{-2} for the piecewise smooth signal of Fig. 2. Left: $p = 1, N = 3$. Right: $p = 2, N = 5$.

2.2 Discretization by cell-averages

In finite volume formulations for HCL, the discrete numerical values are interpreted as approximations to the averages of the solution over the computational cells defined by the underlying grid. In this case it is more appropriate to analyze the data within the *cell average framework*, in which the grid elements are the mesh-cells.

Given any locally integrable function $u(x) : \Omega \to \mathbb{R}$, the cell-average discretization operator is defined as

$$(\mathcal{D}_l u)_k = \mathcal{A}(\Omega_{l,k}) u := \frac{1}{|\Omega_{l,k}|} \int_{\Omega_{l,k}} u(x) dx, \qquad k = 1, \ldots, J_l. \tag{21}$$

Notice that because of the additivity of the integral, relation (3) becomes

$$\bar{u}_k^l = (D_{l+1}^l u^{l+1})_k = \frac{1}{2} \left(\bar{u}_{2k}^{l+1} + \bar{u}_{2k+1}^{l+1} \right)$$

for the 1D meshes considered in the previous section and

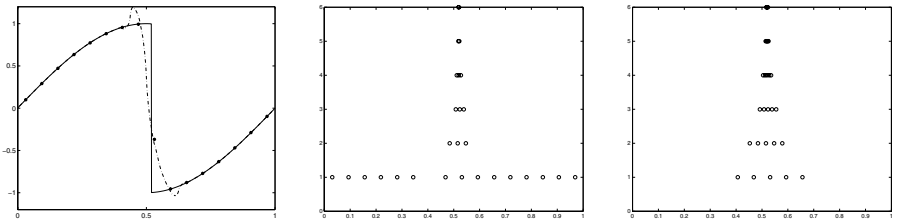
$$(D_{l+1}^l \bar{u}^{l+1})_{k,j} = \frac{1}{4} \left(\bar{u}_{2k,2j}^{l+1} + \bar{u}_{2k+1,2j}^{l+1} + \bar{u}_{2k,2j+1}^{l+1} + \bar{u}_{2k+1,2j+1}^{l+1} \right) \tag{22}$$

for the 2D (tensor-product) extension.

The remaining ingredient in the MR transformation is the reconstruction process. The choice of centered stencils of mesh cells in 1D leads to MR transformations that are closely related to the biorthogonal wavelet framework (see e.g. [Har95, Coh03]). We refer the reader to [BH97] and [Got98] for specific descriptions related to our current application.

Relation between scale coefficients and smoothness. In 1D there is a natural relation between the cell-average and the interpolatory settings which can be exploited to give a simple proof that, for a given reconstruction of degree N, the scale coefficients in the MR representation also satisfy (20).

In Fig. 4 we observe that, as in the interpolatory case, the scale coefficients 'pile up' around the location of a jump discontinuity. The relation between scale coefficients and local regularity can be extracted by employing (13) (see [BH97, Got98] and also [Coh03]).



(a) **(b)** **(c)**

Fig. 4. (a) A piecewise smooth function (Solid line) and its cell-averages on a uniform grid (dots on solid line) . Polynomial reconstruction $p = 1$ (Dotted line); (b)-(c) Detail coefficients above $\varepsilon_j = 10^{-2}/2^{7-j}$, $j = 1,\ldots,6$. (b) $p = 1$, (c) $p = 2$.

3 Multiresolution schemes within Harten's framework

We give next a brief 1D description of the basic strategy used in MR schemes for multidimensional computations. A discretization of a 1D system of conservation laws

$$u_t + F(u)_x = 0 \tag{23}$$

written as

$$U_k^{n+1} = U_k^n - \frac{\delta_t}{\delta_x} B(U^n)_k. \tag{24}$$

is said to be in conservation form if the numerical divergence $B(U^n)_k$ has the form

$$B(U^n)_k = F(U^n_{k-s-1},\ldots,U^n_{k+s}) - F(U^n_{k-s,\ldots,k+s+1}) \tag{25}$$

where the function $F(w_1,w_2,\ldots,w_{2s+1})$ is the numerical flux function.

Let M be a linear MR transformation based on a hierarchical set of computational meshes $\{\mathcal{G}_l\}_{l=0}^{L}$ on the domain of interest. Let us consider (24) on the finest mesh (the reference mesh) and denote the numerical values at time t_n as $U_{L,k}^n$. Since M is a linear transformation, we can write

$$MU_{L,k}^{n+1} = MU_{L,k}^n - \lambda_L MB_{L,k}^n \tag{26}$$

with $\lambda_L = \delta_t / h_l$ and $B_{L,k}^n = B(U_L^n)_k$

Harten's development of multiresolution schemes for HCLs was tightly linked to previous work on Essentially Non Oscillatory (ENO) schemes, a class of HRSC schemes for HCLs that obtain very good resolution properties by performing an elaborate, but very costly, computation of the numerical flux functions at cell interfaces. With the aim of reducing the cost inherent to such schemes, in [BH95, BH97] the authors concentrate on the idea of eliminating heavy flux computations wherever the multiscale analysis reveals that the solution is smooth. This cost-reduction alternative was further explored in different contexts, e.g. [Abg97, Bih96, BOLR01] and the work of the first two authors [CD01, CDM01, RCD03]. In section 3.1 we outline the implementation in [CD01].

Starting with the work in [GM99], a parallel development seeks to perform the time evolution of the numerical values only for a locally refined grid determined from the smoothness information contained in the MR decomposition of the numerical data at the beginning of the time step [Mül02, CKMP03]. In section 3.2 we describe the essential features of this development.

These are two options that lead to essentially different MR-based adaptive schemes for HCL. They both evolved from Harten's original MR-based adaptive concept and share several common ingredients:

The multiresolution transform. The choice of M is dictated by the interpretation of the numerical values. Hence, MR transformations within the cell-average framework are used for the finite-volume schemes considered in [BH95, BH97, Abg97, Bih96, BOLR01] and also in [GM99, Mül02, CKMP03].

Following the work in [SO88], many state of the art HRSC schemes in conservation form consider the numerical values as approximations to the point-values of the solution. This simplifies the implementation on Cartesian meshes in 2D and 3D and was the main motivation in [CD01] for considering the point-value framework as the appropriate multiresolution framework.

The thresholding algorithm. A key point lies in the analysis of the local regularity of *both* U^n and U^{n+1}, where the latter is of course unknown at time n, and how this information is used within the adaptive scheme.

The user introduces a *thresholding parameter* ε, which controls the difference between the reference simulation (the numerical values on the finest grid) and the outcome of the multilevel computation.

Given U_L^n and $MU_L^n = (U_0^n, d^1(U^n), \ldots d^L(U^n))$, the set of indices of *significant coefficients* is constructed as

$$\mathcal{D}_{L,\varepsilon}^n := \left\{ (l,k) \; ; \; |d_k^l(U^n)| > \varepsilon_l, \; k \in J_l, \; l \in \{0,\ldots,L-1\} \right\} \tag{27}$$

where ε_l is a level-dependent threshold value related to ε. This set identifies those locations where the prediction operator produces large errors, hence it is related to locations where data U_L^n display non-regular behavior.

In practice $\varepsilon_l = \varepsilon \ \forall l$ in the point-value framework and $\varepsilon_{L-1} = \varepsilon$, $\varepsilon_l = \varepsilon_{l+1}/2$, $l < L-1$ in the cell-average framework. This choice is motivated by the stability properties of M^{-1}. We refer the interested reader to [Har96].

The prediction step. The corresponding set for U_L^{n+1}, i.e. $\mathcal{D}_{L,\varepsilon}^{n+1}$, is also needed for the design of the adaptive strategy, but since U_L^{n+1} is not known at the beginning of the time step, we can only give an *estimation*, $\tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$, which is computed so that

$$\mathcal{D}_{L,\varepsilon}^{n} \cup \mathcal{D}_{L,\varepsilon}^{n+1} \subset \tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}, \tag{28}$$

To compute the set $\tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$, which marks the non-smooth regions of *both* U_L^n and U_L^{n+1}, all implementations of MR-based schemes known to us employ Harten's heuristic approach: In solving HCLs there are two effects that have to be taken into account: Finite speed of propagation and compressibility (i.e. convergence of characteristics which is ultimately responsible for the creation of shock waves). If a singularity is formed, the CFL condition of the underlying scheme will limit its speed of propagation. If compression mechanisms are steepening up a numerical profile, this should be detected as a loss of local regularity in the behavior of the scale coefficients.

Based on the results of section 2.1 Harten's heuristics for the computation of $\tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$ in 1D proceeds as follows:

$$\begin{aligned} \text{if } (l,k) \in \mathcal{D}_{L,\varepsilon}^{n} &\implies (l,k-i) \in \tilde{\mathcal{D}}_{L,\varepsilon}^{n+1} \quad i = -2,..,2 \\ \text{if } |d_k^l| \geq 2^{N+1}\varepsilon_l \text{ and } l < L &\implies (l+1,2k+i) \in \tilde{\mathcal{D}}_{L,\varepsilon}^{n+1} \quad i = -1,0,1 \end{aligned}$$

The first test takes into account the propagation of information (recall that the propagation of 'real' information is limited by the CFL condition). The second one aims at detecting shock formation. In a smooth region the local rate of decay of the detail coefficients is determined by the accuracy of the interpolation and the local regularity of the function. The second test measures whether the decay rate is that of a smooth function, if this is not the case, compression leading to shock formation might be taking place and the location is also marked.

To the best of our knowledge, Harten's heuristic strategy has not been rigorously verified to satisfy condition (28). In [CKMP03] the authors are able to give a theoretical justification of (28) for a slight modification of Harten's heuristics. For this modification, the reliability of the adaptive scheme is fully ensured, since (28) guarantees that no significant future feature of the solution is missed. In practice, however, Harten's heuristics seems to be sufficient.

3.1 Cost-Effective Multiresolution schemes for shock computations

State of the art HRSC schemes succeed in computing highly accurate numerical solutions (third order or higher) in regions of smoothness, while maintaining at the same

time sharp, oscillation free, numerical profiles at discontinuities. These desirable features are achieved by performing a specific, and often very expensive, computation of the numerical flux functions at each cell boundary.

When the underlying grid is uniform, the implementation of these shock capturing schemes is quite straightforward. Fine grid simulations with HRSC schemes render very precise numerical approximations to the solution of HCLs, but the heavy-duty flux computations increase the computational cost in such a way that for some HRSC schemes 2D fine mesh simulations on personal computers are out of reach simply because they cost too much.

It is common knowledge that the high-powered flux computations involved in these schemes are only strictly needed at existing singularities or when these are about to form. Hence, if *both* U^n and U^{n+1} are smooth around a specific location on the underlying computational mesh, it means that no singularity is present or will be created in the course of the computation, hence we could avoid using the numerical flux functions of the HRSC scheme in the computation of the numerical divergence at that location. On the other hand, around a discontinuity (or when a steep gradient makes it imminent), the full power of the HRSC scheme is needed, if the high-resolution properties of the scheme are to be maintained.

This observation forms the basis of the cost-effective alternative: The goal is to substitute the direct (HRSC) computation of the numerical divergence on the finest mesh by a multilevel computation based on the smoothness information obtained from the MR transformation M. The basic *cost-reduction* assumption is that interpolating numerical divergences is considerably faster than computing the necessary fluxes.

The cost-reduction alternative involves no memory savings. However, and precisely because of this feature, there is no need for special data structures and the necessary MR modules can be easily incorporated into any existing CFD code. When memory requirements are not a major concern (as it is the case in many 2D computations), this technique often provides the necessary *cost-reduction* factors to allow very fine HRSC simulations on personal computers [CD01, RCD03].

The boolean flag. Thresholding and prediction, as described in the previous section, are combined so that the set $\tilde{\mathcal{D}}_L^{n+1}$ is converted into the determination of a boolean flag whose value, 0 or 1, will determine the choice of procedure to compute the numerical divergence.

Starting from a zero value for all b_k^l and a given tolerance parameter ε, the following two tests are applied:

$$\begin{aligned} &\text{if } |d_k^l| \geq \varepsilon_l \implies b_{k-i}^l = 1 \quad i = -2,..,2 \\ &\text{if } |d_k^l| \geq 2^{N+1}\varepsilon_l \text{ and } l < L \implies b_{2k+i}^{l+1} = 1 \quad i = -1,0,1 \end{aligned}$$

In the point-value setting, the extension to 2D is straightforward (see [CD01]). For the cell-average framework see [BH95, DGM00].

The multilevel evaluation of the numerical divergence within the point-value setting. Instead of computing B_L^n with the HRSC scheme at all points in $\mathcal{G}_L$, in

[CD01] we apply the following procedure involving the boolean flags defined previously

1) Compute B_0^n (*i.e* for all points of the coarsest grid $\mathcal{G}_0$) directly with the HRSC scheme.
2) To obtain B_L^n, repeat for $l = 0$ to $L-1$:
   For each $x_{2k+1}^{l+1} \in \mathcal{G}_{l+1} - \mathcal{G}_l$:
   - if $b_k^l = 1$ the location is flagged as non-smooth and a precise computation of the numerical divergence is required: Compute $B_{l+1,2k+1}^n$ directly with the HRSC scheme.
   - if $b_k^l = 0$ the location belongs to a smooth region and a direct (expensive) computation can be avoided: Compute $B_{l+1,2k+1}^n$ using the prediction operator in M, i.e. $B_{l+1,2k+1}^n = (P_l^{l+1} B_l^n)_{2k+1}$.

The cpu gain of this algorithm lies in the fact that the cost of the prediction operator is negligible compared to the expensive HRSC evaluation.
As an example, in our implementation for the 2D Euler equations [CD01], the prediction operator is based on a 2D Lagrange polynomial interpolation of degree 3 (see [BH97]) and we use a (formally) third order HRSC described in [DM96] for which the effective ratio for the evaluation of one numerical divergence is about $1/100$.
Remark: Since the grids $\{\mathcal{G}_l\}_l$ are embedded, and the numerical values are attached to mesh points in the point-value setting, each direct (HRSC) evaluation must be done with the values of U^n on the finest grid $\mathcal{G}_L$ (they are always available in this method). This guarantees that the numerical divergence is *always* computed by the HRSC with a precision related to the *finest* spatial discretization.

This multilevel strategy has been detailed, analyzed and tested on several benchmark tests involving the 2D Euler equations. Numerical results, which are reported in a series of papers [CD01, CDM01, RCD03], indicate that the quality of the multilevel approximation (the difference between the outcome of the multilevel algorithm and the reference simulation) is directly controlled by the tolerance parameter ε used in the thresholding algorithm.

The efficiency of the multilevel scheme is, of course, problem dependent. We refer again to the aforementioned papers for a specific evaluation of the efficiency of the multilevel scheme in several situations. Here we present a specific simulation that illustrates its performance.

We have represented on Fig. 8 a numerical simulation of the interaction between a Taylor vortex and a Mach 4 shock. This interaction is difficult to handle numerically for Mach numbers larger than 2, and the use of robust HRSC methods is necessary in order to represent correctly the physics of the problem. In Fig. 8, the finest grid $\mathcal{G}_L$ has 512×256 uniformly spaced points and the coarsest $\mathcal{G}_0$ is 8×4 points (6 levels of refinement). The thresholding parameter is $\varepsilon = 5.10^{-4}$. We display a numerical Schlieren plot of the density at times $t = 0$ and $t = 0.4$. Associated to each density plot we show a display where only the points for which the numerical divergence has been evaluated by the HRSC scheme are represented.

After the interaction the vortex is highly distorted and a strong acoustic wave develops ahead of it. All these features are correctly identified by the adaptive strategy, which validates the smoothness analysis done by the wavelet coefficients. In this simulation, the percentage of points where a direct evaluation is done grows only from 3.4% to 10.6%, leading to an effective cpu time reduction of a factor **6.8** compared to a reference simulation without the multilevel strategy.

The use of the multilevel method outlined in this section, made it possible to perform in [RCD03] a detailed study of the phenomenology of these interactions on a personal computer.

3.2 Fully Adaptive Finite Volume Schemes

With the aim of reducing the computational costs with regard to both computational time *and* memory requirements but still maintaining the accuracy of the reference scheme, a modified approach was developed in [Mül02]. The main idea is to evolve in time only the cell averages of a *locally refined grid*, so that the finest (reference) mesh does not need to be available.

In this approach, the ultimate goal is to provide an algorithm that can be realized with an optimal complexity, i.e., the number of floating point operations is proportional to the number of cells in the adaptive grid. This requires new data structures supporting the local structure of the algorithm. *Hash tables* have lead to efficient implementations of this approach (see [MV00]). *Tree structures* have also been considered (see [RSTB03]).

In the following we summarize the main ingredients specific to this *fully adaptive* concept, namely, *local grid refinement* and *evolution of local cell averages*. For more specific details we refer to [Mül02].

Local Grid Refinement

The starting point is a locally refined grid characterized by the index set $\mathcal{G}^n_{L,\varepsilon} \subset \{(l,k)\,;\, k \in J_l,\ l = 0,\ldots,L\}$ such that

$$\Omega = \bigcup_{(l,k)\in\mathcal{G}^n_{L,\varepsilon}} \Omega_{l,k}.$$

that is provided with cell averages $\{U^n_{l,k}\}_{(l,k)\in\mathcal{G}_{L,\varepsilon}}$ corresponding to time step n. It is required that the set $\mathcal{G}^n_{L,\varepsilon}$ has the structure of a graded tree.

We will now summarize the six steps of the local grid refinement procedure in the context of cell averages, namely, (i) *local MR transformation*, (ii) *thresholding*, (iii) *prediction*, (iv) *grading*, (v) *local grid refinement* and (vi) *local inverse MR transformation*.

Local MR transformation. As outlined in section 2, we perform a MR analysis of the cell averages at hand which provides a new data format composed of data on a coarsest discretization level and arrays of details describing the difference information between the data on two consecutive discretization levels. For this purpose

we proceed level by level from fine to coarse as indicated in (11). Note that the two-scale transformation is performed locally only for the indices corresponding to the adaptive grid instead of the full levels. In particular, applying the local two-scale transformation can be interpreted as a successive coarsening of the grid where fine-grid cells are agglomerated to a coarse-grid cell and the difference information is stored by the detail coefficients.

Thresholding. We now apply a hard thresholding to the sequence of detail coefficients, i.e., discard all details d_k^l that fall in absolute value below a certain threshold value. Here we apply the same strategy as in section 3. For this purpose we compute the index set $\mathcal{D}_{L,\varepsilon}^n$ corresponding to the *significant details* according to (27).

Prediction. To perform the evolution step, we have to determine the adaptive grid on the *new* time level. Therefore we *predict* all significant details on time level $n+1$ that may become significant due to the evolution. In practice, we use Harten's heuristic strategy summarized in section 3 to compute the prediction set $\tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$.

Grading. In order to perform the grid adaptation procedure level by level we need that the index set of significant details corresponds to a *graded tree*, i.e., the levels of neighboring cells differ at most by one. Since the set $\tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$ is in general not graded, we have to apply in addition a grading procedure. This will slightly inflate the index set of significant details but has so far been observed not to spoil the complexity reduction of floating point operations in any significant way. In fact, from the nature of singularities occurring in flow computations one expects the distribution of significant details to exhibit at least nearly tree structure (see Figs. 3, 4).

Grid adaptation. Then we exploit the inflated set $\tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$ to determine an associated index set $\mathcal{G}_{L,\varepsilon}^{n+1}$ which characterizes the adaptive grid at the new time level. The index set $\mathcal{G}_{L,\varepsilon}^{n+1}$ is initialized by all indices of the coarsest discretization. Then, traversing through the levels from coarse to fine we proceed as follows: if $(l,k) \in \tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$ then the cell $\Omega_{l,k}$ is locally refined, i.e., the index (l,k) is removed from $\mathcal{G}_{L,\varepsilon}^{n+1}$ and the indices of the subcells on the finer level are added to $\mathcal{G}_{L,\varepsilon}^{n+1}$. Finally we obtain the locally adapted grid which naturally corresponds to the leaves of the graded tree of significant details.

Local inverse MR transformation By the previous step the grid has locally changed due to local refinement and coarsening. In order to determine the cell averages $\{U_{(l,k)}^n\}_{(l,k)\in\mathcal{G}_{L,\varepsilon}^{n+1}}$, we employ a local inverse MR transformation interrelating the local cell averages $(U_{l,k}^n)_{(l,k)\in\mathcal{G}_{L,\varepsilon}^n}^n$ and the significant details $(d_k^l)_{(l,k)\in\mathcal{D}_{L,\varepsilon}^n}$. Again we proceed level by level from coarse to fine where we locally replace a cell average on the coarse scale by the cell averages of its subcells whenever there is a significant detail associated to this coarse cell in $\tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}$. Note that the computation of these cell averages can be simultaneously determined when performing the grid adaptation.

Evolution of Local Cell Averages

The time evolution of the cell averages is now performed on the new adaptive grid determined by the index set $\mathcal{G}_{L,\varepsilon}^{n+1}$

$$U_{l,k}^{n+1} = U_{l,k}^{n} - \lambda_{l,k} B_{l,k}^{n}, \quad \lambda_{l,k} := \frac{\delta t}{|\Omega_{l,k}|} \qquad (l,k) \in \mathcal{G}_{L,\varepsilon}^{n+1}, \tag{29}$$

where $B_{l,k}^{n}$ denotes the numerical divergence of cell $\Omega_{l,k}$.

In principle, the adaptive grid could be interpreted as an unstructured grid and the numerical divergence could be computed by the local data at hand, see for instance [RSTB03]. Since the ultimate goal is the design of an adaptive scheme with an error still corresponding to the discretization of the *finest* grid, this strategy could result in a severe accuracy deficiency. Therefore we have to be more careful in the computation of the local numerical divergence. For this purpose, we assume that a reference FVS is given on the uniform finest grid $\mathcal{G}_L$ similar to (29) with the numerical divergence

$$B_{L,k}^{n} := \sum_{\Gamma_{k,r}^{L} \subset \partial\Omega_{L,k}} |\Gamma_{k,r}^{L}| \, F_{k,r}^{L,n}$$

determined by the sum of all fluxes over all cell edges of the cell $\Omega_{L,k}$. Here $\Gamma_{k,l}^{L}$ denotes the interface of the cell $\Omega_{L,k}$ to the neighbor cell $\Omega_{L,r}$ and $F_{k,r}^{L,n}$ the corresponding numerical flux. See Fig. 5 for clarification of notation. The numerical fluxes are assumed to be conservative, i.e.,

$$F_{k,r}^{L,n} = -F_{r,k}^{L,n}. \tag{30}$$

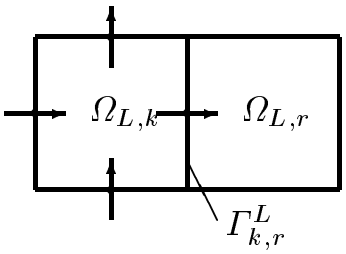


Fig. 5. Finite volume discretization.

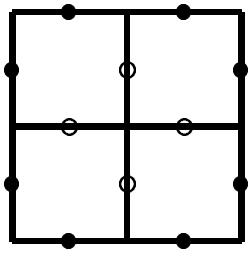


Fig. 6. Local flux balances, (•) boundary flux, (◦) internal flux .

Applying (virtually) the linear MR transformation M for cell averages similar to (26) results in local evolution equations for the cell averages on the coarser scales $l = 0, \ldots, L-1$, see (29), where the local numerical divergence is recursively defined by

$$B_{l,k}^{n} := \sum_{\Omega_{l+1,r} \subset \Omega_{l,k}} B_{l+1,r}^{n}. \tag{31}$$

This is sketched in Fig. 6 for a dyadic grid refinement. According to (31) we have to compute all fluxes marked by • and ◦. However, the internal fluxes corresponding to ◦ cancel each other out due to the conservation property (30) resulting in a significant reduction of the computational complexity. Finally, we end up with

$$B^n_{l,k} = \sum_{\Gamma^l_{k,r} \subset \partial\Omega_{l,k}} |\Gamma^l_{k,r}| F^{l,n}_{k,r} \tag{32}$$

where the local numerical fluxes are defined by

$$F^{l,n}_{k,r} := \sum_{\Gamma^{l+1}_{j,s} \subset \Gamma^l_{k,r}} \frac{|\Gamma^{l+1}_{j,s}|}{|\Gamma^l_{k,r}|} F^{l+1,n}_{j,s} = \sum_{\Gamma^L_{j,s} \subset \Gamma^l_{k,r}} \frac{|\Gamma^L_{j,s}|}{|\Gamma^l_{k,r}|} F^{L,n}_{j,s}. \tag{33}$$

Together with (29) this specifies the fully adaptive scheme. Note that we never employ the complexity of the finest grid.

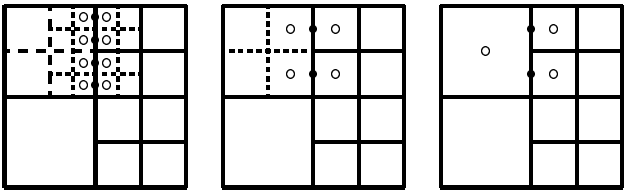


Fig. 7. Exact (left), locally structured (middle) and unstructured (right) flux computation, (•) numerical flux, (◦) cell average .

Remark. In 1D the local flux computation simplifies because there are no hanging nodes in the adaptive grid. Due to the nestedness of the grids, see Fig. 2.1, the numerical fluxes on level l coincide with the numerical fluxes on the higher scales, i.e.,

$$F^n_{l,k} = F^n_{l+1,2k} = \ldots = F^n_{L,2^{L-l}k} \equiv F\left(U^n_{L,2^{L-l}k-s,\ldots,2^{L-l}k+s+1}\right). \tag{34}$$

Since the numerical divergence on the coarser levels is recursively defined we further conclude

$$B^n_{l,k} := B^n_{l,2k} + B^n_{l,2k+1} = \sum_{i=0}^{2^{L-l}-1} B^n_{L,2^{L-l}k+i} = F^n_{L,2^{L-l}(k+1)} - F^n_{L,2^{L-l}k}.$$

Remark. According to (33) the numerical fluxes have to be computed by the data on the *finest* scale. In order to provide these data we have to perform locally an inverse two-scale transformation. In 1D this does not degrade the complexity of the algorithm but it will in higher space dimensions. For HRSC we may perform the local flux computation by means of the local data at hand instead of the data on the finest scale, see Fig. 7 (middle and right). In practice, this does not affect the accuracy but preserves the computational complexity. In this case, the number of flux computations is proportional to the number of significant detail coefficients $\#\mathcal{D}^n_{L,\varepsilon}$ or $\#\mathcal{G}^n_{L,\varepsilon}$, respectively.

Remark. In Harten's original approach [Har95, BH97], the complexity is not reduced. To see this, we consider Fig. 6. Here expensive fluxes based on higher–order upwind discretizations are computed at • and, in addition, cheap finite difference flux approximations have to be computed at ◦.

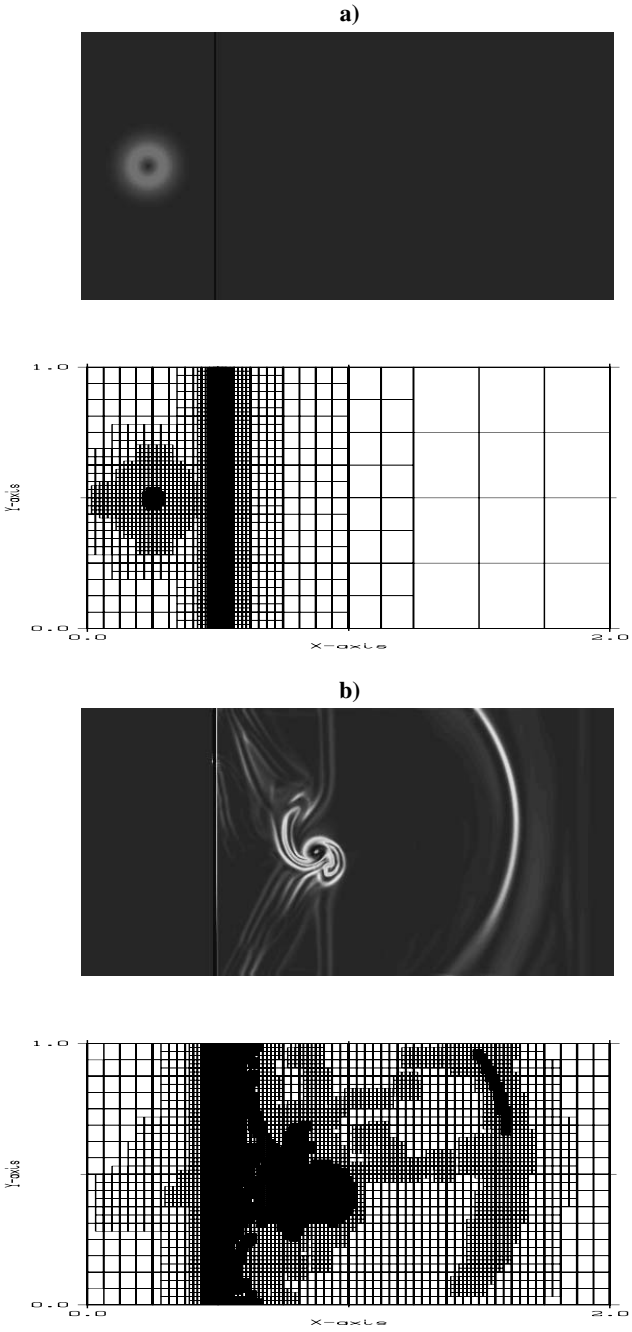


Fig. 8. *Schlieren pictures of the density field of a Mach 4 shock-vortex interaction and associated multilevel grids. a) time* $t = 0$*, b) time* $t = 0.4$*.*

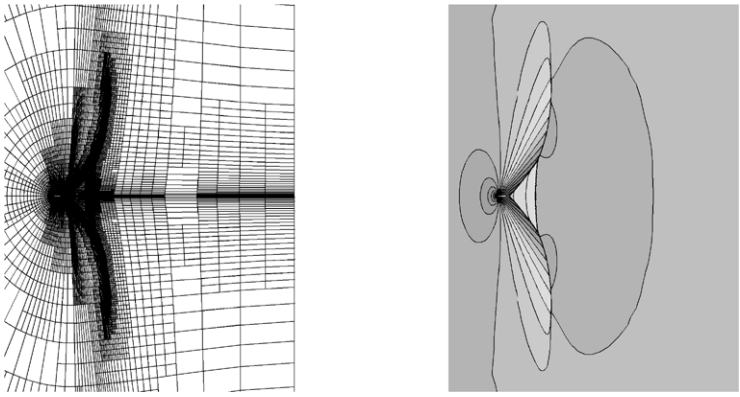


Fig. 9. Total view of NACA0012 airfoil, $M = 0.95$, $\alpha = 0.0°$. *Left*: Computational grid. *Right*: Pressure distribution, $M_{min} = 0.0$, $M_{max} = 1.45$, $\Delta M = 0.05$.

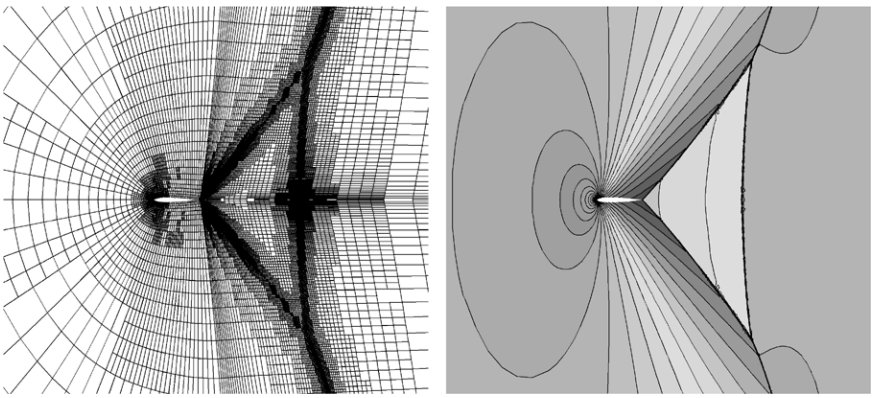


Fig. 10. Detail of Fig. 9.

Data Structures, Error Control and Real World Applications

In the previous section we outlined an adaptive MR scheme with an optimal complexity in the sense that the number of operations is proportional to the number of unknowns, i.e., $\#\mathcal{D}_{L,\varepsilon}$ and $\#\mathcal{G}_{L,\varepsilon}$, respectively. In order to design an *optimal* code, i.e., the memory requirements and the CPU time are proportional to the complexity of the adaptive algorithm, it turns out that the choice of the *data structures* and the *memory management* has a significant influence on the performance of the computation. In particular, the design of appropriate data structures crucially depends on the underlying adaptive algorithm, i.e., the data structures have to be adapted to the algorithmic requirements and should not be designed independently. For our purposes the concept of *hashing* turns out to be an efficient tool. To this end, we developed the template library `igpm_t_lib`, see [MV00], from which we derive the appropriate data structures for the realization of the adaptive code.

Concerning the quality of the computation we are aiming at the accuracy of the reference scheme. For this purpose, the local data provided by the adaptive scheme are projected onto the finest mesh applying the inverse MR transformation where the non-significant details are put to zero. The ideal strategy would be to determine the threshold value ε such that the *discretization error* of the reference scheme, i.e., difference between exact solution and reference scheme, and the *perturbation error*, i.e., the difference between the reference scheme and the adaptive scheme, are balanced. For scalar conservation laws this concept was rigorously verified, see [CKMP03].

By now the new adaptive MR concept has been applied by several groups with great success to different real world applications, e.g., 2D/3D–steady state computations of compressible fluid flow around air wings modeled by the Euler and Navier–Stokes equations, respectively, as well as fluid-structure interactions on block–structured curvilinear grid patches [BLM03], non–stationary shock–bubble interactions on 2D Cartesian grids for Euler equations [Mül02], backward–facing step on 2D triangulations [CKP02] and simulation of a flame ball modeled by reaction–diffusion equations on 3D Cartesian grids [RSTB03].

In Figs. 9 and 10 we show a numerical simulation of the transonic flow over a NACA0012 airfoil at $M_\infty = 0.95$, $\alpha = 0^\circ$. The flow pattern downstream of the trailing edge is characterized by a complex shock configuration frequently referred to as *fish-tail*. The oblique shocks extend about 10 to 12 chord lengths into the flow domain. The steady–state computation was carried out using an implicit local–time stepping using the QUADFLOW solver [BLM03]. The final adaptive grid consists of 55084 cells which provides a very high resolution over the complete extent of the shocks. Such a high shock resolution is not feasible using standard structured grids. Discretization of the shock region only by a uniform structured mesh equals about $29.5 \cdot 10^6$ grid cells. A uniform discretization of the complete flow domain would result in about 10^8 cells.

4 Conclusion

Harten's early developments on the use of MR techniques for numerical computations involving HCLs were presented as an attractive alternative to the adaptive grid methodology. The schemes currently in use show that the MR decomposition of the numerical data at each time step provides an adequate tool to adapt the computational resources to the nature of the data.

Cost-effective MR-based schemes provide an easy to use adaptive tool that has been succesfully used to investigate the behavior of new HRSC schemes. Fully adaptive MR-based schemes become a novel AMR technique, where the refinement criteria is based on the smoothness information contained in the MR representation of the data.

Acknowledgements: Research supported by EU financed networks no. HPRN-CT-2002-00282 and no. HPRN-CT-2002-00286. The second author acknowledges partial support from Spanish MCYT-BFM2001-2814. The third author acknowledges funding from the Deutsche Forschungsgemeinschaft in the Collaborative Research Centre SFB 401 "Flow Modulation and Fluid-Structure Interaction at Airplane Wings" of the RWTH Aachen, University of Technology, Aachen, Germany.

References

Abg97. R. Abgrall. Multiresolution analysis on unstructured meshes: Applications to CFD. In Chetverushkin et al., editor, *Experimentation, modelling and computation in flow, turbulence and combustion*. John Wiley & Sons, 1997.

AH98. R. Abgrall and A. Harten. Multiresolution representation in unstructured meshes. *SIAM J. Num. Anal.*, 35-6:2128–2146, 1998.

BC89. M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Comput. Phys.*, 82:64–84, 1989.

BH95. B. Bihari and A. Harten. Application of generalized wavelets: An adaptive multiresolution scheme. *J. Comp. Appl. Math*, 61:275–321, 1995.

BH97. B. Bihari and A. Harten. Multiresolution schemes for the numerical solution of 2–D conservation laws I. *SIAM J. Sci. Comput.*, 18(2):315–354, 1997.

Bih96. B. Bihari. Multiresolution schemes for conservation laws with viscosity. *J. Comp. Phys.*, 123:207–225, 1996.

BLM03. F. Bramkamp, Ph. Lamby, and S. Müller. An adaptive multiscale finite volume solver for unsteady and steady state flow computations. 2003. To appear in J. Comp. Phys.

BMP92. E. Bacry, S. Mallat, and G. Papanicolaou. A wavelet based space-time adaptive numerical method for partial differential equations. *Mathematical Modeling and Numerical Analysis*, 26(7):793, 1992.

BO84. M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. of Comput. Phys.*, 53:484–512, 1984.

BOLR01. B.L. Bihari, D.K. Ota, Z. Liu, and S.V. Ramakrishnan. The multiresolution method on general unstructured meshes. *AIAA*, (2001-2553), 2001.

CD01. G. Chiavassa and R. Donat. Point value multiresolution for 2D compressible flows. *SIAM J. Sci. Comput.*, 23(3):805–823, 2001.

CDKP00. A. Cohen, N. Dyn, S.M. Kaber, and M. Postel. Multiresolution finite volume schemes on triangles. *J. Comp. Phys.*, 161:264–286, 2000.

CDM01. G. Chiavassa, R. Donat, and A. Marquina. Fine-mesh numerical simulations for 2d Riemann problems with a multilevel scheme. In H. Freistühler and G. Warnecke, editors, *Hyperbolic Problems: Theory, Numerics, Applications*, pages 247–256. Birkhäuser, 2001.

CKMP03. A. Cohen, S.M. Kaber, S. Müller, and M. Postel. Fully Adaptive Multiresolution Finite Volume Schemes for Conservation Laws. *Math. Comp.*, 72(241):183–225, 2003.

CKP02. A. Cohen, S.M. Kaber, , and M. Postel. Multiresolution Analysis on Triangles: Application to Gas Dynamics. In G. Warnecke and H. Freistühler, editors, *Hyperbolic Problems: Theory, Numerics, Applications*, pages 257–266. Birkhäuser, 2002.

Coh03. A. Cohen. *Numerical Analysis of Wavelet Methods*. Studies in Mathematics and its Applications, 32. North-Holland-Elsevier, Amsterdam, 2003.

Dah97. W. Dahmen. Wavelet and multiscale methods for operator equations. *Acta Numerica*, 6:55–228, 1997.

DD89. G. Deslauries and S. Dubuc. Symmetric iterative interpolation processes. *Constructive Approximation*, 5:49–68, 1989.

DGM00. W. Dahmen, B. Gottschlich–Müller, and S. Müller. Multiresolution schemes for conservation laws. *Numer. Math.*, 88(3):399–443, 2000.

DM96. R. Donat and A. Marquina. Capturing shock reflections: An improved flux formula. *J. Comp. Phys.*, 125(1):42–58, 1996.

Don92. D. Donoho. Interpolating wavelet transforms. Technical Report 408, Dept. of Statistics, Stanford University, 1992.

GM99. B. Gottschlich–Müller and S. Müller. Adaptive finite volume schemes for conservation laws based on local multiresolution techniques. In M. Fey and R. Jeltsch, editors, *Hyperbolic Problems: Theory, Numerics, Applications*. Birkhäuser, 1999.

Got98. B. Gottschlich–Müller. *On Multiscale Concepts for Multidimensional Conservation Laws*. PhD thesis, RWTH Aachen, October 1998.

Har95. A. Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. *Comm. Pure Appl. Math.*, 48(12):1305–1342, 1995.

Har96. A. Harten. Multiresolution representation of data: A general framework. *SIAM J. Numer. Anal.*, 33(3):1205–1256, 1996.

Hol99. M. Holmström. Solving hyperbolic pdes using interpolatory wavelets. *SIAM J. Sci. Comput.*, 21-2:405–420, 1999.

LT. J. Liandrat and P. Tchamitchian. Resolution of the 1d regularized burgers equation using a spatial wavelet approximation. ICASE Report 90-83.

Mül02. S. Müller. *Adaptive Multiscale Schemes for Conservation Laws*, volume 27 of *Lecture Notes on Computational Science and Engineering*. Springer, 2002.

MV00. S. Müller and A. Voss. A Manual for the Template Class Library `igpm_t_lib`. IGPM–Report 197, RWTH Aachen, 2000.

RCD03. A. Rault, G. Chiavassa, and R. Donat. Shock-vortex interactions at high mach numbers. *J. Scientific Computing*, 19:347–371, 2003.

RSTB03. O. Roussel, K. Schneider, A. Tsigulin, and H. Bockhorn. A conservative fully adaptive multiresolution algorithm for parabolic PDEs. *J. Comp. Phys.*, 188(2):493–523, 2003.

SO88. C.-W. Shu and S. Osher. Efficient implementation of essentially non–oscillatory shock capturing schemes I. *J. Comp. Phys.*, 77:439–471, 1988.

Multiresolution adaptive space refinement in geophysical fluid dynamics simulation

Aimé Fournier[1], Gregory Beylkin[2], and Vani Cheruvu[2]

[1] Department of Meteorology, College of Computer, Mathematical, and Physical Sciences, University of Maryland at College Park fournier@ucar.edu
[2] Department of Applied Mathematics, University of Colorado at Boulder

Summary. We review part of the methodology for multiresolution adaptive solution of PDEs introduced by Alpert et al. (2002) in 1D (§2.1), and introduce a 2D generalization and implementation (§2.2). This methodology is similar to the spectral-element method (SEM, e.g., Fournier et al. 2004) in that it combines *spectral* accuracy with *finite-element* efficiency, but is not exactly SEM. We present 2D dynamical test cases (§2.3) that exhibit decreasing range of active scales (Heat Eq.), or else increasing range due to strong nonlinearities (Burgers Eq.). We conclude by showing that our methodology adapts to such evolving phenomena in these PDEs (§3), thereby saving computational cost, while preserving a high preselected representation accuracy per time step.

1 Introduction

By now the effectiveness of adaptive mesh refinement (AMR) is well known, for numerically solving certain PDEs in which only parts of the domain contain intense activity on which computational effort must be focused. Standard AMR methods are relatively low-order in space, an accuracy limitation that could become serious when modeling phenomena such as geophysical fluid dynamics (GFD), that involve strongly nonlinearly interacting structures across very many scales. Moreover, GFD applications such as weather/climate prediction, turbulence etc. call for *both* adaptivity and high-order accuracy. In this paper we address both requirements.

2 Method

2.1 Multiresolution spatial representation in 1D

We briefly describe the 1D spatial-discretization method in this section. For a more complete discussion and motivation see [ABGV02].

Basis functions

The representation of all dependent variables $u = u(x)$ in the domain $x \in [0,1]$ is similar to that in a type of structured-AMR codes in one respect, namely that $[0,1[$ is subdivided into dyadic intervals $\mathbb{I}_{k,\ell} \equiv [k/2^\ell, (k+1)/2^\ell[$ for $k \in \{0,\dots 2^\ell - 1\} \equiv \mathbb{K}_\ell$. However, instead of the usual single-point-value or cell-averaged approximations of u within any $\mathbb{I}_{k,\ell}$, greater accuracy is afforded as follows. Expand u in the piecewise-polynomial, orthonormal interpolating basis

$$\phi_{j,k,\ell}(x) \equiv \begin{cases} \sqrt{2}^\ell \phi_j(2^\ell x - k), & x \in \mathbb{I}_{k,\ell}, \\ 0, & \text{otherwise}, \end{cases}$$

constructed from the normalized Lagrange interpolating polynomial

$$\phi_j(\xi) \equiv \frac{1}{\sqrt{w_j}} \prod_{j' \neq j} \frac{\xi - \xi_{j'}}{\xi_j - \xi_{j'}} \in \mathbb{P}_p, \qquad j \in \{0,\dots p\}, \xi \in [0,1],$$

where ξ_j denotes the Gauss quadrature nodes, which are the $N \equiv p+1$ roots of the mapped Legendre polynomial $\mathrm{L}'_N(\xi) \equiv \mathrm{L}_N(2\xi - 1)$ and correspond to weights $w_j \xleftarrow{\xi_j \leftarrow \xi} (N \mathrm{L}'_p \frac{\mathrm{d}}{\mathrm{d}\xi} \mathrm{L}'_N)^{-1}$. The basis *interpolates* on *nodes* $x_{j,k,\ell} \equiv (\xi_j + k)/2^\ell \in \mathbb{I}_{k,\ell}$: $\phi_{j,k,\ell}(x_{j',k',\ell}) = \sqrt{2^\ell / w_j}\, \delta_{j,j'} \delta_{k,k'}$. The basis is also *orthonormal* with respect to *location* $x_{j,k,\ell}$: $\langle \phi_{j,k,\ell} \phi_{j',k',\ell} \rangle = \delta_{j,j'} \delta_{k,k'}$, where

$$\langle u \rangle \equiv \int_0^1 u(\xi) \mathrm{d}\xi \xleftarrow{u \in \mathbb{P}_{2N-1}} \sum_{j=0}^{p} u(\xi_j) w_j.$$

This approach is similar to that of the SEM, except that the element-boundary Gauss-Lobatto nodes $x = k/2^\ell$ are omitted. Thus the space $\mathbb{V}_{k,\ell}$ of polynomials on $\mathbb{I}_{k,\ell}$ has precisely the same dimension N as do its "left" and "right" child spaces $\mathbb{V}_{2k,\ell+1}$ and $\mathbb{V}_{2k+1,\ell+1}$, whose union properly contains it:

$$\operatorname*{span}_{j=0}^{p} \phi_{j,k,\ell} = \mathbb{V}_{k,\ell} \subsetneq \bigcup_{b=0}^{1} \mathbb{V}_{2k+b,\ell+1}. \tag{1}$$

As discussed by [ABGV02], these functions lead to a form of the standard *multiresolution analysis*, with the usual ladder structure

$$\bigcup_{k \in \mathbb{K}_\ell} \mathbb{V}_{k,\ell} \equiv \mathbb{V}_\ell \subsetneq \mathbb{V}_{\ell+1}$$

of function spaces. To simplify notation, we embed the local node index j in matrix notation; thus the familiar two-scale refinement relation for scaling functions becomes

$${}^{\mathrm{t}}(\phi_{0,k,\ell}, \dots \phi_{p,k,\ell}) \equiv \phi_{k,\ell} = \sum_{b=0}^{1} \mathsf{H}_b \phi_{2k+b,\ell+1}, \tag{2}$$

where $H_{b,j',j} = \sqrt{w_j/2}\, \phi_{j'}(x_{j,b,1})$ are the low-pass quadrature mirror filters.

The transformation between the projection $\mathcal{P}_\ell u$ of u onto $\mathbb{V}_\ell$ and its $2^\ell N$ interpolating scaling-function coefficients $\bar{u}_{j,k,\ell} = \sqrt{w_j/2^\ell}\mathcal{P}_\ell u(x_{j,k,\ell})$ is

$$\mathcal{P}_\ell u \equiv \sum_{k\in\mathbb{K}_\ell} {}^{\mathrm{t}}\phi_{k,\ell}\bar{u}_{k,\ell},$$

$${}^{\mathrm{t}}(\bar{u}_{0,k,\ell},\ldots\bar{u}_{p,k,\ell}) \equiv \bar{u}_{k,\ell} \equiv \langle\phi_{k,\ell}\mathcal{P}_\ell u\rangle = \sum_{b=0}^{1} \mathsf{H}_b\bar{u}_{2k+b,\ell+1}. \tag{3}$$

Unlike in the usual SEM, this transform is *orthonormal*, i.e., $||\mathcal{P}_\ell u|| = ||\bar{u}_\ell||$ in the respective $\mathbb{L}^2$ norms. In practice, if one may assume $u = \mathcal{P}_L u$ for a "small enough" sampling scale 2^{-L}, then the finest coefficients are simply $\bar{u}_{j,K,L} = \sqrt{w_j/2^L}u(x_{j,K,L})$; thereafter for $\ell < L$, (3) is used iteratively to obtain larger-scale $\bar{u}_{k,\ell}$. (Note that for brevity, the whole "multiwavelet" side of the construction in [ABGV02] is omitted here.)

Dynamic adaptivity with prescribed accuracy

As in the usual structured-AMR, here *dynamic adaptivity* is accomplished by computing index sets $\mathbb{K}^\star_\ell$ wherever elements only as small as $2^{-\ell} > 2^{-L}$ are *locally* sufficient. To distinguish from finite-difference methods, we may say that this is not just "mesh refinement" but "space refinement," i.e., representation in a multiresolution function space "just large enough" to represent u with local and global relative accuracy threshold ε, as we now describe.

The multiresolution structure (1) allows an *adaptivity criterion* that measures the projection energy in $\mathbb{V}_{k,\ell}$ relative to the projection energy (which is always no less) in the twice-larger space $\cup_{b=0}^{1}\mathbb{V}_{2k+b,\ell+1}$. Starting from $\mathbb{K}^\star_{-1} \equiv \varnothing$, define $\mathbb{K}^\star_\ell$ (of size $\le 2^\ell$) iteratively by

$$\mathbb{K}^\star_\ell \equiv \left\{ k \in \mathbb{K}_\ell \;;\; \lfloor\frac{k}{2}\rfloor \notin \mathbb{K}^\star_{\ell-1} \;\;\&\;\; \sum_{b=0}^{1} ||\bar{u}_{2k+b,\ell+1}||^2 \le (1+\varepsilon^2)||\bar{u}_{k,\ell}||^2 \right\}. \tag{4}$$

By construction, $k \in \mathbb{K}^\star_\ell$ iff $2k+b \notin \mathbb{K}^\star_{\ell+1}$, so that the $\mathbb{I}_\ell \equiv \cup_{k\in\mathbb{K}^\star_\ell}\mathbb{I}_{k,\ell}$ form a disjoint cover: $\cup_{\ell=0}^{L}\mathbb{I}_\ell = [0,1[$ and $\mathbb{I}_\ell \cap \mathbb{I}_{\ell'\neq\ell} = \varnothing$. Then the adaptively *compressed* representation of u is given by only $\sum_{\ell=0}^{L}\#\mathbb{K}^\star_\ell \le 2^L$ terms:

$$\mathcal{P}_L u \approx \mathcal{P}^\star_L u \equiv \sum_{\ell=0}^{L}\sum_{k\in\mathbb{K}^\star_\ell} {}^{\mathrm{t}}\phi_{k,\ell}\bar{u}_{k,\ell}. \tag{5}$$

One may easily show that the compression error is bounded by

$$||(\mathcal{P}_L - \mathcal{P}^\star_L)u|| \le \varepsilon||\mathcal{P}_L u||. \tag{6}$$

Adaptively evaluating nonlinear terms

Interpolating bases greatly simplify calculating nonlinear terms, compared to spectral methods that use convolutions or inverse transforms. For example,

$$\overline{uv}_{j,k,\ell} \approx \sqrt{2^\ell / w_j}\,\bar{u}_{j,k,\ell}\bar{v}_{j,k,\ell}. \tag{7}$$

In practice, (7) is implemented by *oversampling*, using

$$\bar{u}_{2k+b,\ell+1} \approx {}^{\mathrm{t}}\mathsf{H}_b \bar{u}_{k,\ell}, \tag{8}$$

which is an exact reconstruction iff $u = \mathcal{P}_\ell u$. As explained in [ABGV02, §3.3.3], (8) is first applied to refine both factors of (7) until both use all the same index sets $\mathbb{K}^\star_\ell$. Then both factors are again oversampled one level, from the scale $2^{-\ell}$ sufficient to represent the individual factors, down to $2^{-\ell-1}$. This oversampling has a similar purpose to "de-aliasing" in spectral methods, i.e., to allow for multiplication to create smaller scales in the product. Finally, the result is compressed if possible, using (4) again.

Differentiation on the smallest scale

The derivative operation $u = \frac{\mathrm{d}}{\mathrm{d}x}\chi$ is approximated in $\mathbb{V}_L$ by the convolution

$$\bar{u}_{K,L} = \sum_{K'=K-1}^{K+1} \mathsf{D}_{K,K'} \bar{\chi}_{K',L} + \tau_{K,L}, \tag{9}$$

where τ is the truncation error. Each block of the $2^L N \times 2^L N$ block-tridiagonal matrix D uses the usual projection form $\langle \phi_j \frac{\mathrm{d}}{\mathrm{d}\xi}\phi_{j'}\rangle = \sqrt{w_j}\frac{\mathrm{d}}{\mathrm{d}\xi}\phi_{j'}(\xi_j)$, plus integration by parts to provide communication between neighboring intervals:

$$\begin{aligned}
\mathsf{D}_{K,K-1} &= -2^L\ c_0'\phi(0)\,{}^{\mathrm{t}}\phi(1),\\
\mathsf{D}_{K,K} &= 2^L\big(c_1\phi(1)\,{}^{\mathrm{t}}\phi(1) - c_0\phi(0)\,{}^{\mathrm{t}}\phi(0) - \langle \tfrac{\mathrm{d}}{\mathrm{d}\xi}\phi\,{}^{\mathrm{t}}\phi\rangle\big),\\
\mathsf{D}_{K,K+1} &= 2^L\ c_1'\phi(1)\,{}^{\mathrm{t}}\phi(0).
\end{aligned}$$

Note that the off-diagonal blocks are only rank-one, as in the SEM (if one accounts for the SEM $\mathbb{C}^0$ condition).

In [ABGV02, §4.3] it is explained how to use the free parameters $c_b = 1 - c_b' \in [0,1]$ to easily *incorporate boundary conditions* without losing accuracy, and without increasing condition number. The truncation error

$$\tau_{K,L} = \frac{1}{2^{(N-1/2)L}} \sum_{b=0}^{1} \Big((-1)^{(1-b)p} c_b - (-1)^{bp} c_b'\Big)\,\rho_{K+b}\phi(b) + O\left(\frac{1}{2^{(N+1/2)L}}\right)$$

is also derived there, where $\rho_K \equiv \frac{N!}{(2N)!}\,(\frac{\mathrm{d}}{\mathrm{d}x})^N u\big(K/2^L\big)$. Choosing $c_b = \frac{1}{2}$ has the effect of enforcing $\mathbb{C}^0$ continuity by averaging approximations of the off-node value $u(K/2^L)$ due to the two neighbors $\mathbb{I}_{K-b,L}$, as in the SEM. Also note that when p is even and $c_b = \frac{1}{2}$, the truncation error is of higher order.

Operations across scales

In the adaptive case, any general operation $u = \mathcal{T}\chi$ represented at scale 2^{-L} by T, is projected up to appropriate scales $2^{-\ell} > 2^{-L}$ using *projection operators* $\mathcal{P}_{k,\ell} \equiv {}^{\mathrm{t}}\phi_{k,\ell}\langle\phi_{k,\ell}\cdot\rangle$ from $\mathbb{V}_L$ to each $\mathbb{V}_{k,\ell} \subsetneq \mathbb{V}_L$. Thus, (9) generalizes to

$$\bar{u}_{k',\ell'} \approx \sum_{\ell=0}^{L} \sum_{k\in\mathbb{K}^\star_\ell} \mathsf{T}_{k',\ell',k,\ell}\bar{\chi}_{k,\ell}, \qquad k' \in \mathbb{K}^{\star}_{\ell'}{}', \tag{10}$$

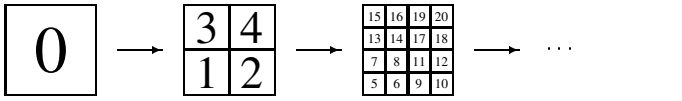
as follows. A large interval $\mathbb{I}_{k,\ell}$ contains a small interval $\mathbb{I}_{K,L}$ iff there is a Boolean vector $b \in \{0,1\}^{L-\ell}$ such that $K = 2^{L-\ell}k + \sum_{m=0}^{L-\ell-1} 2^m b_m$. The path from $K_0 := K$ to $K_{L-\ell} = k$ through a binary tree is iteratively encoded by b:

$$K_m := \lfloor K_{m-1}/2 \rfloor, \qquad b_m := K_m - 2K_{m+1}. \tag{11}$$

By iterating (2), one finds that $\mathcal{P}_{k,\ell} = {}^{\mathrm{t}}\phi_{k,\ell}\Sigma_b \mathsf{P}_b \langle\phi_{K,L}\cdot\rangle$, where $\mathsf{P}_b = \mathsf{H}_{b_0}\cdots\mathsf{H}_{b_{L-\ell-1}}$. It follows that $\mathsf{T}_{k',\ell',k,\ell} = \mathsf{P}_{b'}\mathsf{T}_{K',K}{}^{\mathrm{t}}\mathsf{P}_b$. Note that if T can possibly amplify, attenuate or translate smaller-scale structures (e.g., $\mathsf{T} = \mathsf{D}$ or e^{D} to some power), then the $\mathbb{K}^\star_\ell$ appropriate by (4) for the operand might not equal the $\mathbb{K}^\star_{\ell'}{}'$ for the result; thus in practice we may find it necessary to oversample the operand using (8) and compress the result if possible.

2.2 Formulation in 2D using tensor products

For the purpose of organizing an adaptive 2D representation, it is convenient to index 2D with a single index k as follows. Let the quad-tree subdivision sequence



show how the parent element $\mathbb{E}_0 = \mathbb{I}_{0,0} \times \mathbb{I}_{0,0}$ divides into its disjoint cover of four child elements $\mathbb{E}_{1+b+2b'} = \mathbb{I}_{b,1} \times \mathbb{I}_{b',1}$, 16 grandchild elements, etc. Then iterating in this way, for any $k > 0$, the element $\mathbb{E}_k$ has a *parent* $\mathbb{E}_{\lfloor (k-1)/4 \rfloor} \supsetneq \mathbb{E}_k$, and four *children* $\mathbb{E}_{4k+1+q} \xrightarrow{\bigcup_{q=0}^{3}} \mathbb{E}_k$.

At any given level ℓ, there are 4^ℓ elements $\mathbb{E}_k$ of area $4^{-\ell}$ for all $k \in \{i_\ell, \ldots 4i_\ell\} \equiv \mathbb{K}^2_\ell$, where $i_\ell \equiv \frac{1}{3}(4^\ell - 1)$ is the initial index. Each k encodes its own level via $\ell = \lfloor \log_4(1+3k) \rfloor$. As in (11) for b, one iterates to find a *path* $q \in \{0, \ldots 3\}^\ell$ through a quad tree from $K_0 := k$ to $K_\ell = 0$:

$$K_m := \lfloor (K_{m-1} - 1)/4 \rfloor, \qquad q_m := K_m - 4K_{m+1} - 1. \tag{12}$$

Given q, the inverse of (12) is $k = i_\ell + \sum_{m=0}^{\ell-1} 4^m q_m$. The *location* of element

$$\mathbb{E}_k = \mathbb{I}_{k_x,\ell} \times \mathbb{I}_{k_y,\ell} \tag{13}$$

is given by $\vec{k}/2^\ell$, where

$$\vec{k} = (k_x, k_y) = \sum_{m=0}^{\ell-1} 2^m (b_m, b'_m), \quad b_m := q_m \bmod 2 \quad \text{and} \quad b'_m := \lfloor \frac{q_m}{2} \rfloor. \tag{14}$$

To any $\mathbb{E}_k$ there corresponds a 2D piecewise-polynomial function space

$$\mathbb{V}_k^2 \equiv \mathbb{V}_{k_x,\ell} \otimes \mathbb{V}_{k_y,\ell}. \tag{15}$$

For $I \times I'$ and $J \times J'$ matrices A and B, define an $IJ \times I'J'$ matrix tensor product

$$(\mathsf{A} \otimes \mathsf{B})_{i+Ij,i'+I'j'} \equiv A_{i,i'} B_{j,j'}.$$

With these notations, the 2D piecewise-polynomial, orthonormal scaling-function basis of $\mathbb{V}_k^2$, and the two-scale relation generalizing (2) may be written

$$\phi_{k_x,\ell}(x) \otimes \phi_{k_y,\ell}(y) \equiv \phi_k(\vec{x}) = \sum_{q=0}^{3} \mathsf{H}_q^2 \phi_{4k+1+q}(\vec{x}),$$

where $\mathsf{H}_{b+2b'}^2 \equiv \mathsf{H}_b \otimes \mathsf{H}_{b'}$. Similarly, $\vec{\nabla}$ is represented by $(\mathsf{D} \otimes \mathsf{I}, \mathsf{I} \otimes \mathsf{D})$.

All of the other 1D formulas from §2.1 straightforwardly generalize to 2D, e.g., if $u \in \mathbb{V}_K^2$ then $\bar{u}_{\vec{j},K} = \sqrt{w_{j_x} w_{j_y}/4^L}\, u(\vec{x}_{\vec{j},K})$, where the 2D nodes are

$$\vec{x}_{\vec{j},k} \equiv (x_{j_x,k_x,\ell}, x_{j_y,k_y,\ell}) \tag{16}$$

and $\bar{u}_k = \sum_{q=0}^3 \mathsf{H}_q^2 \bar{u}_{4k+1+q}$. As in (4), the indexes for compression come from

$$\mathbb{K}_\ell^\star \equiv \left\{ k \in \mathbb{K}_\ell^2 \;;\; \lfloor \tfrac{k-1}{4} \rfloor \notin \mathbb{K}_{\ell-1}^\star \;\;\&\;\; \sum_{q=0}^{3} ||\bar{u}_{4k+1+q}||^2 \le (1+\varepsilon^2) ||\bar{u}_k||^2 \right\}, \tag{17}$$

and the 2D analogs of (5) and (6) hold. In practice, we implement and update the adaptive-length $\bar{u}$ as a hash-table data structure.

2.3 Test-case dynamical equations

We present two test cases, both spatially biperiodic initial-value problems that illustrate dynamic adaptation to changing or interacting scales. For the 2D heat or diffusion equation

$$\partial_t u = \nabla^2 u \tag{18}$$

we introduce a "rotated array of peaks" initial condition

$$u(0,\vec{x}) = \sum_{\vec{n} \in \mathbb{Z}^2} \mathrm{e}^{-a_1|n_x| - a_2|n_y| + \mathrm{i}\vec{n}\cdot\vec{x}'} = \frac{\sinh a_1}{\cosh a_1 - \cos x'} \frac{\sinh a_2}{\cosh a_2 - \cos y'}, \tag{19}$$

where $a \in \,]0,\infty[^2$ controls the width and height of the peaks, $\vec{x}' \equiv \left(\begin{smallmatrix} l_x & l_y \\ -l_y & l_x \end{smallmatrix}\right) \cdot (\vec{x} - \vec{x}_\mathrm{o})$ is a dilated, rotated, offset coordinate, $\vec{l} \in (2\pi\mathbb{Z})^2$ controls the dilation by $l = |\vec{l}|$ and rotation by $\arctan \frac{l_y}{l_x}$, and $\vec{x}_\mathrm{o} \in \mathbb{E}_0$ is the offset of the peaks.

The 2D Burgers equation includes diffusion and nonlinear advection:

$$\begin{aligned}\partial_t \vec{u} &= \nu\nabla^2\vec{u} - \vec{u}\cdot\vec{\nabla}\vec{u} \\ &= \vec{\nabla}(\nu\vec{\nabla}\cdot\vec{u} - \tfrac{1}{2}u^2) - (\nu\vec{\nabla} - \vec{u})\times(\vec{\nabla}\times\vec{u}),\end{aligned} \tag{20}$$

where ν^{-1} is the Reynolds number. For irrotational solutions $\vec{u} = \vec{\nabla}\chi$, (20) implies (up to an irrelevant, uniform term) the scalar dynamics

$$\partial_t\chi = \nu\nabla^2\chi - \tfrac{1}{2}|\vec{\nabla}\chi|^2. \tag{21}$$

Choosing a generalization of the traditional initial condition $u_x \to \sin x$,

$$\chi(0,\vec{x}) = -\frac{1}{l_x}\cos x', \tag{22}$$

implies a steepening front in u_x perpendicular to $\vec{l}$, with $\vec{l}\times\vec{u} = \vec{0}$ for all t.

2.4 Time marching algorithm

Problems (18-19) and (21-22) were solved using the following algorithm:

1. Pre-select accuracy ε, degree p and smallest element length 2^{-L}
2. Use (17) to construct an adaptive representation (5) in 2D of (19) or (22), yielding an initial scale range $2^{-L} \le 2^{-\ell_{\max}} \le 2^{-\ell_{\min}} \le 1$
3. For stability, choose $(\Delta t)^{-1}$ $\propto$ the largest-magnitude eigenvalue of $^{t}\mathsf{DD}$ ($\times\nu$ for Burgers), reduced by $\times 2^{\ell_{\max}-L}$ as in [ABGV02, Eq. (4.4)]
4. For (21) but not (18), oversample the current solution using (8) in 2D, replacing $\bar{u}_k$ at scale $2^{-\ell}$ by $\bar{u}_{4k+1+q} := {}^{t}\mathsf{H}_q^2\bar{u}_k$ at scale $2^{-\ell-1}$
5. Advance the current oversampled adaptive solution, with dynamics governed by the projection in 2D of (18) or (21), from t to $t+\Delta t$ using either quality-controlled fourth-order Runge-Kutta (which updates Δt) [PFTV88] for (21), or else forward Euler for (18)
6. If possible, compress via (17) and update $\ell_{\min}$ and $\ell_{\max}$ and hence, Δt
7. If not done go to 4

Alpert et al. [ABGV02] used a similar algorithm in 1D, but with fixed Δt and the exact linear part (ELP) time-evolution scheme that has superior stability and other qualities [BKV98]. We intend to implement ELP in 2D and compare with the more standard schemes used in the present paper.

3 Preliminary results and discussion

We present 3D surface plots of our numerical solutions of (18) and (21) in Figs. 1 and 2, accompanied by Table 1. It is evident that the methods presented by [ABGV02] in 1D and generalized here to 2D achieve qualitatively correct solutions to both test cases. In subsequent work we intend to report errors between the adaptive solutions and analytic or very-high-uniform-resolution solutions. Quantitatively we observe that this method affords relatively high order, while saving computation cost in proportion to the number ($4^{\ell_{\max}} - K_{\text{tot}}$ in Table 1) of small-scale elements that were replaced by larger-scale ones, and while preserving relative accuracy ε (6) at each time step.

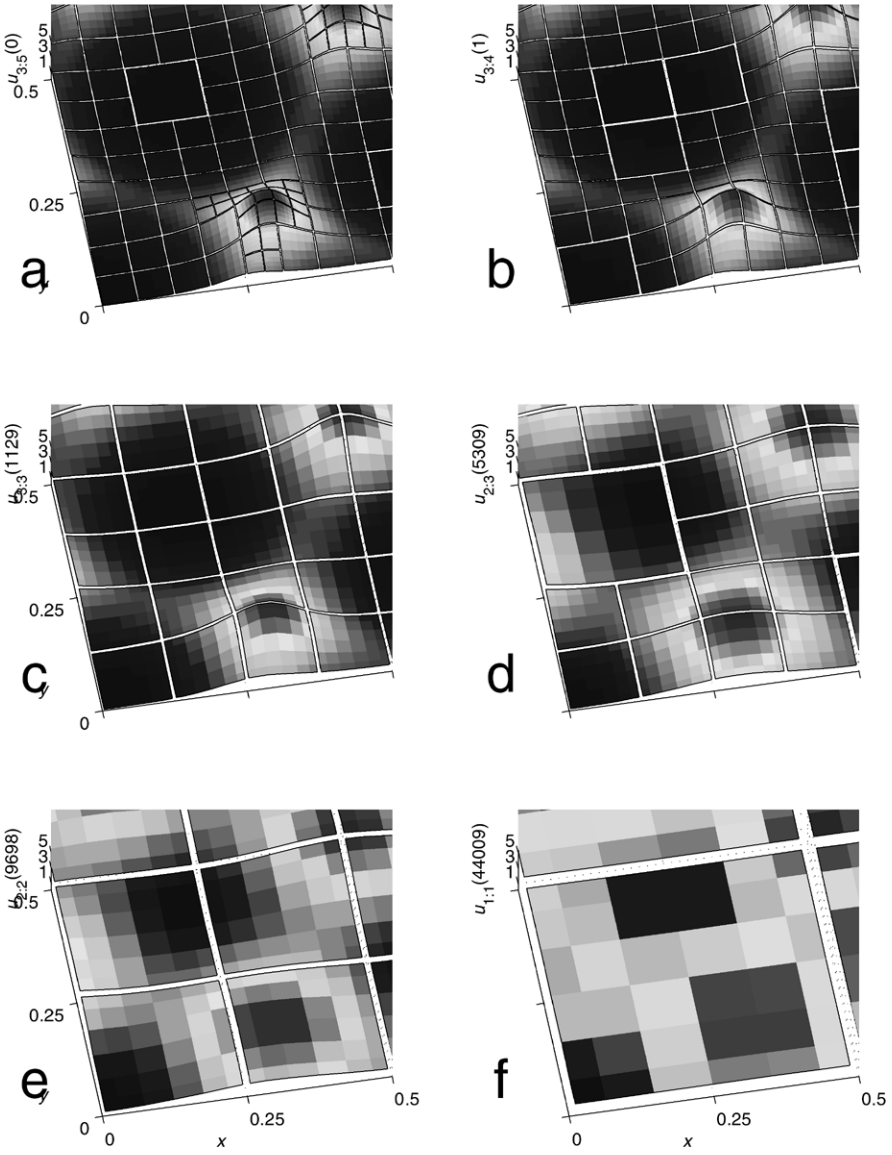


Fig. 1. Surface plots of $u(t,x,y)$ for $(x,y) \in [0,\frac{1}{2}]^2$, from solving the Heat Eq. (18-19), on adaptive piecewise-polynomial spaces $\mathbb{V}_k^2$ (15) corresponding to elements $\mathbb{E}_k$ (13) that cover the biperiodic domain $[0,1]^2$. Each $\mathbb{E}_k$ contains 7^2 Gauss quadrature nodes $\vec{x}_{\vec{j},k}$ (16). The plots (a-f) correspond to times t in the rows of Table 1. Each vertical plot axis is labeled $u_{\ell_{\min}:\ell_{\max}}(n)$ to indicate the smallest and largest element edges $2^{-\ell_{\max}}$ and $2^{-\ell_{\min}}$ at time step n. In (19) were set $e^{-a_1} = e^{-a_2} = \frac{2}{5}$, $\vec{l} = 2\pi(1,2)$ and $\vec{x}_o = (\frac{1}{2},\frac{1}{2})$. Observe that in (a) the peaks (red) generate locally small-scale representations, that evolve with time to locally larger scales.

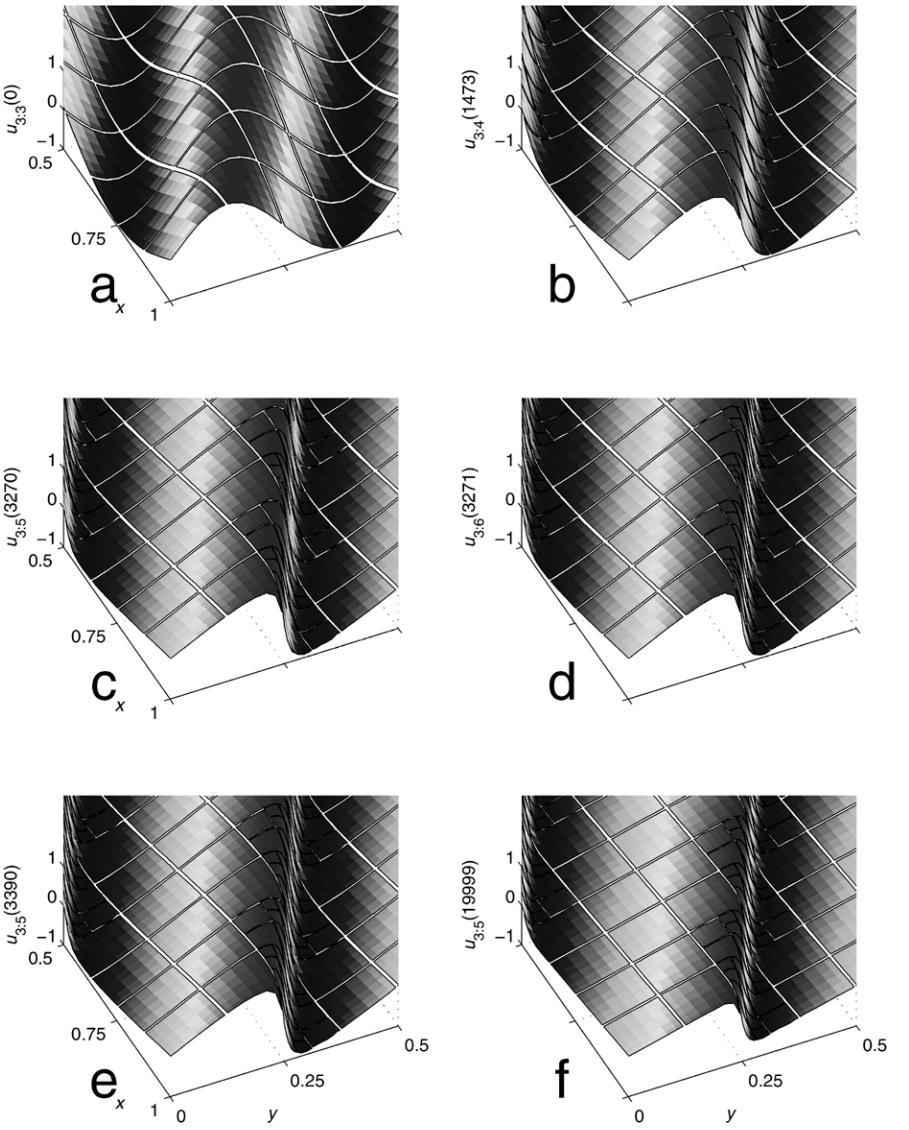


Fig. 2. As in Fig. 1 but for $u_x = \partial_x \chi$ from the Burgers Eq. (21-22), zoomed to view $(x,y) \in [\frac{1}{2},1] \times [0,\frac{1}{2}]$. Observe the formation over time of steep fronts along the line family $\vec{l} \cdot \vec{x} \in (2\mathbb{Z}+1)\pi$, e.g., the line $1-x = 2y - \frac{1}{2}$ is prominent in this view. This behavior is no more than the rotation of 1D Burgers dynamics into 2D, since if $U(t,x)$ solves the 1D Burgers Eq., then $\vec{u}(t,\vec{x}) = \vec{l}U(l^2 t, \vec{l} \cdot \vec{x})$ solves (20).

Table 1. Data for solutions of (18) and (21). Each row corresponds to a plot in Figs. 1 and 2 at time step n. The total number of elements is the sum $K_{\text{tot}} = \sum_{\ell=\ell_{\min}}^{\ell_{\max}} \#\mathbb{K}_\ell^\star$ of numbers of elements adapted to all scales $2^{-\ell}$, and $p = 6$ always.

	Heat $\varepsilon = 10^{-8}$, $\Delta t = 3.7 \times 10^{-7}$				Burgers $\varepsilon = 10^{-10}$, $\nu = 10^{-2}$					
	n	$\ell_{\min}$	$\ell_{\max}$	K_{tot}	n	Δt_n (10^{-5})	$t = \sum_n \Delta t_n$ (10^{-2})	$\ell_{\min}$	$\ell_{\max}$	K_{tot}
(a)	0	3	5	328	0	3.7	0.	3	3	64
(b)	1	3	4	208	1473	2.2	5.01	3	4	112
(c)	1129	3	3	64	3270	1.9	7.90	3	5	160
(d)	5309	2	3	52	3271	1.9	7.91	3	6	256
(e)	9698	2	2	16	3390	1.4	8.05	3	5	208
(f)	44009	1	1	4	19999	0.7	21.8	3	5	208

Acknowledgments

Thanks to Bob Cramer for his hash-table formulation of 2D adaption, and for initial implementations of (12) and (14). This material is based on work supported by the National Science Foundation, Grant 0083048. We acknowledge the National Center for Atmospheric Research, which is sponsored by the NSF, for computing time. NCAR is operated by the University Corporation for Atmospheric Research under sponsorship of the NSF. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

ABGV02. B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi. Adaptive solution of partial differential equations in multiwavelet bases. *J. Comput. Phys.*, 182(1):149–190, October 2002.

BKV98. G. Beylkin, J. M. Keiser, and L. Vozovoi. A new class of time discretization schemes for the solution of nonlinear pdes. *J. Comput. Phys.*, 147:362–387, 1998.

FTT03. Aimé Fournier, Mark A. Taylor, and Joseph J. Tribbia. The Spectral Element Atmosphere Model (SEAM): High-resolution parallel computation and localized resolution of regional dynamics. *Mon. Wea. Rev.*, 132:726–748, 2004.

PFTV88. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1988.

Anisotropic mesh adaptivity in CFD

Stefano Micheletti and Simona Perotto

MOX, Modeling and Scientific Computing, Department of Mathematics, Politecnico of Milano, via Bonardi 9, I-20133 Milano, Italy stefano.micheletti@mate.polimi.it, simona.perotto@mate.polimi.it

1 Why anisotropy?

The straightforward answer to this question could be: *because anisotropy is everywhere!* Actually, when numerically solving a problem in Computational Fluid Dynamics (CFD), or in some other areas, there are many instances where the solution shows *directional features* such as great variations along certain directions with less significant changes along other ones, e.g. boundary and internal layers, singularities or shocks. A typical example is provided by the solution of an advection-diffusion problem, as shown in Fig. 1 [17, 18]. On the computational domain $\Omega = (0, 1)^2$, a

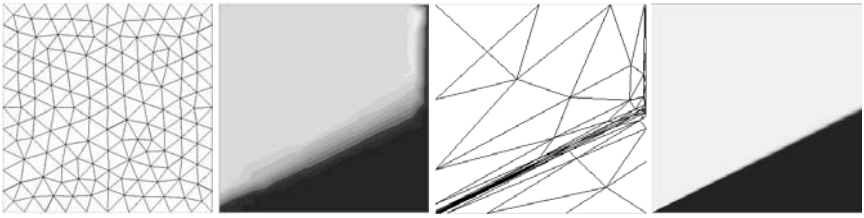


Fig. 1. Isotropy (left) versus Anisotropy (right): meshes and contour lines

standard advective-diffusive problem is solved for the scalar u, in the presence of a convective field $\mathbf{a} = (2, 1)^T$ and with a diffusivity $\mu = 10^{-4}$, completed with Dirichlet boundary conditions, i.e. $u = 1$ on the left and top sides and $u = 0$ on the remaining ones. The solution u exhibits an internal and a boundary layer of thickness $O(10^{-2})$ and $O(10^{-4})$, respectively. As Fig. 1 shows, the isotropic mesh consists of more elements than the corresponding anisotropic one (312 versus 64 triangles). In the latter case a correct orientation and deformation of the mesh elements (longest edges parallel to the boundary layers) yields a great reduction of the number of triangles. Moreover, in the anisotropic case the layers are captured more sharply.

This simple but remarkable example highlights the "leitmotiv" of an anisotropic analysis: for a fixed solution accuracy, reduce the number of degrees of freedom

involved in the approximation of the problem at hand by better orienting the mesh elements according to some suitable features of the solution, or vice versa, given a constraint on the number of elements, find the mesh maximizing the accuracy of the numerical solution.

Going back to the utility of anisotropy, we observe that things may not be so straightforward, of course. While anisotropy is proved to be superior in terms of effectiveness for the most accurate computations in many cases, yet, there are some instances where a structured Adaptive Mesh Refinement (AMR) procedure turns out to be more simple to carry out, especially in view of an implementation in a parallel environment. Moreover, in the unstructured case, the main drawback of the anisotropic approach compared to the isotropic one, is the more complex analysis required to fully describe the element dimensions and orientation. Though, this heavier burden is the strength of the method. For other approaches in the anisotropic context, see e.g. [1, 7, 8, 15, 19].

The outline of the article is the following. In Sect. 2 we introduce the anisotropic framework by recalling some anisotropic interpolation error estimates, representing the main tool used in the a posteriori error analysis addressed in Sect. 3. This analysis is discussed in the case of a general differential operator, moving from the adjoint theory for goal-oriented error control, and it is then detailed for the advection-diffusion-reaction and the Stokes problems. Finally, in Sect. 4 the effectiveness of the anisotropic philosophy is assessed on some numerical test cases.

2 The anisotropic setting for FEM

Let $\Omega \subset \mathbb{R}^2$ be a polygonal domain and, for any $0 < h \leq 1$, let $\{\mathcal{T}_h\}_h$ be a family of conforming triangulations of $\overline{\Omega}$ into triangles K of diameter $h_K \leq h$.
Following the idea proposed in [10], in order to derive the additional information for the geometrical description of the mesh triangles, we move from the standard affine transformation $T_K : \widehat{K} \to K$, with $K = M_K(\widehat{K}) + \mathbf{b}_K$, $M_K \in \mathbb{R}^{2\times 2}$ and $\mathbf{b}_K \in \mathbb{R}^2$, from the reference triangle $\widehat{K}$ into K, where $\widehat{K}$ can be, e.g., the right triangle $(0,0),(1,0),(0,1)$ or the equilateral one $(-1/2,0),(1/2,0),(0,\sqrt{3}/2)$ (see Fig. 2). Let $M_K = B_K Z_K$ be the polar decomposition of the invertible matrix M_K, with B_K and Z_K a symmetric positive definite and an orthogonal matrix, respectively. Then

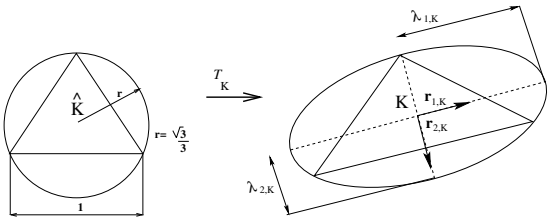


Fig. 2. The map T_K

we factorize the matrix B_K in terms of its eigenvalues $\lambda_{i,K}$ (with $\lambda_{1,K} \geq \lambda_{2,K}$) and eigenvectors $\mathbf{r}_{i,K}$, for $i = 1, 2$, as $B_K = R_K^T \Lambda_K R_K$, where $\Lambda_K = \mathrm{diag}(\lambda_{1,K}, \lambda_{2,K})$ and $R_K = [\mathbf{r}_{1,K}, \mathbf{r}_{2,K}]^T$. As Fig. 2 shows, the eigenvectors $\mathbf{r}_{i,K}$ provide the directions of the semi-axes of the ellipse circumscribed to the element K, while the eigenvalues $\lambda_{i,K}$ measure the length of such semi-axes. Thus, the shape and orientation of each triangle K is completely described by the quantities $\mathbf{r}_{i,K}$ and $\lambda_{i,K}$. The deformation of K with respect to $\widehat{K}$ can be measured by the so-called *stretching factor* $s_K = \lambda_{1,K}/\lambda_{2,K} (\geq 1)$, being $s_{\widehat{K}} = 1$.

2.1 Functional framework

Throughout, we use a standard notation to denote the Sobolev spaces of functions with Lebesgue-measurable derivatives, and their norms [16]. In more detail, let $W^{k,p}(\Omega)$ be the Sobolev space of functions for which th p-th power of the absolute value of their distributional derivatives of order up to $k \geq 0$ is Lebesgue-measurable, with $1 \leq p < \infty$. For $p = 2$ we let $H^k(\Omega) = W^{k,2}(\Omega)$. In particular, $L^2(\Omega)$ is the space of square-integrable functions with norm $\|\cdot\|_{L^2(\Omega)}$ and scalar product $(\cdot,\cdot)$, while for the space $H^k(\Omega)$ we denote by $\|\cdot\|_{H^k(\Omega)}$ and $|\cdot|_{H^k(\Omega)}$ the corresponding norm and seminorm, respectively. When the norms or seminorms are referred to some subspace S of Ω, they are written as $\|\cdot\|_{L^2(S)}$, $\|\cdot\|_{H^k(S)}$ and $|\cdot|_{H^k(S)}$, while the scalar product is denoted by $(\cdot,\cdot)_S$. We also recall that $L^\infty(\Omega)$ is the space of bounded functions a.e., while $W^{1,\infty}(\Omega) \subset L^\infty(\Omega)$ is such that also the first derivatives are bounded a.e. Finally, $C^0(\overline{\Omega})$ denotes the space of continuous functions on $\overline{\Omega}$.

2.2 Anisotropic interpolation error estimates

The starting point for the a posteriori analysis in Sect. 3 has been the derivation of suitable anisotropic interpolation error estimates [10, 12, 17].
We have proved estimates for both the Lagrange and the Clément-like interpolants [5, 6] to take into account different regularity of the function to be interpolated. Denoting by W_h the finite element space of continuous affine functions, let $\Pi_h : C^0(\overline{\Omega}) \to W_h$ and $I_h : L^2(\Omega) \to W_h$ be the Lagrange and Clément linear interpolants, respectively and let their restrictions to each element $K \in \mathcal{T}_h$ be Π_K and I_K. Then we have:

Proposition 1. *Let $v \in H^2(K)$, for any $K \in \mathcal{T}_h$. Then there exist two constants $C_1 = C_1(\widehat{K})$ and $C_2 = C_2(\widehat{K})$ such that*

$$\|v - \Pi_K(v)\|_{L^2(K)} \leq C_1 \left[\sum_{i,j=1}^{2} \lambda_{i,K}^2 \lambda_{j,K}^2 L_K^{i,j}(v) \right]^{1/2}, \tag{1}$$

$$|v - \Pi_K(v)|_{H^1(K)} \leq C_2 \lambda_{2,K}^{-1} \left[\sum_{i,j=1}^{2} \lambda_{i,K}^2 \lambda_{j,K}^2 L_K^{i,j}(v) \right]^{1/2}, \tag{2}$$

where

$$L_K^{i,j}(v) = \int_K \left(\mathbf{r}_{i,K}^T H_K(v)\, \mathbf{r}_{j,K}\right)^2 d\mathbf{x}, \quad \textit{with } i,j = 1,2, \tag{3}$$

and $H_K(v)$ is the Hessian matrix associated with v.

Proposition 2. *Let $v \in H^1(\Omega)$. Then there exist two constants $C_3 = C_3(M, \widehat{C})$ and $C_4 = C_4(M, \widehat{C})$ such that, for any $K \in \mathcal{T}_h$,*

$$\|v - I_K(v)\|_{L^2(K)} \le C_3 \left[\sum_{i=1}^{2} \lambda_{i,K}^2 \left(\mathbf{r}_{i,K}^T G_K(v) \mathbf{r}_{i,K}\right)\right]^{1/2}, \tag{4}$$

$$|v - I_K(v)|_{H^1(K)} \le C_4\, \lambda_{2,K}^{-1} \left[\sum_{i=1}^{2} \lambda_{i,K}^2 \left(\mathbf{r}_{i,K}^T G_K(v) \mathbf{r}_{i,K}\right)\right]^{1/2}, \tag{5}$$

where $G_K(v) \in \mathbb{R}^{2\times 2}$ is the symmetric positive semi-definite matrix with entries $(G_K(v))_{i,j} = \int_{\Delta_K} \partial v/\partial x_i\, \partial v/\partial x_j\, d\mathbf{x}$ with $\mathbf{x} = (x_1, x_2)^T \in K$, Δ_K is the patch of all the elements sharing a vertex with K, and $M \in \mathbb{N}$ and $\widehat{C} > 0$ are the constants defined through the relations

$$\mathrm{card}(\Delta_K) \le M \quad \textit{and} \quad \mathrm{diam}(\Delta_{\widehat{K}}) \le \widehat{C}, \tag{6}$$

with $\Delta_{\widehat{K}} = T_K^{-1}(\Delta_K)$.

Remark 1. Requirements (6) demand the cardinality of any patch Δ_K as well as the diameter of the reference patch $\Delta_{\widehat{K}}$ to be uniformly bounded independently of the geometry of the mesh. In particular, the latter inequality rules out some too distorted reference patches (see Fig. 1.1 in [17]).

A comparison of the inequalities in Proposition 1 and 2 with the corresponding isotropic results shows that the anisotropic estimates are more complex. For instance, let us consider the isotropic estimate corresponding to (1), given by

$$\|v - \Pi_K(v)\|_{L^2(K)} \le C_1^*\, h_K^2\, |v|_{H^2(K)}, \tag{7}$$

with $C_1^* = C_1^*(\widehat{K})$. From a dimensional viewpoint, we have both in (1) and in (7) the square of the spacing parameters (i.e. h_K in the isotropic case, $\lambda_{1,K}$, $\lambda_{2,K}$ in the anisotropic one). On the other hand, the H^2-seminorm of v in (7) is replaced by a suitable sum of the $L_K^{i,j}(v)$ quantities in (1). We claim that the information provided by the seminorm $|v|_{H^2(K)}$ has been split along the directions $\mathbf{r}_{1,K}$ and $\mathbf{r}_{2,K}$ via the quantities $L_K^{i,j}(v)$ representing squared L^2-norms of directional second-order derivatives of v. As anticipated in Sect. 1, we are replacing the "lumped" isotropic results with more "distributed" ones. The pay-off of such a framework is that we are able to finely tune the adapted meshes in terms of shape and orientation of the elements.
Finally, in view of the a posteriori analysis of Sect. 3, we have also derived anisotropic estimates for the L^2-norm of the interpolation error on the edges e of the triangulation $\mathcal{T}_h$ (see [12] for the details).

3 Anisotropic a posteriori error analysis

Moving from the a posteriori dual-based approach developed in [3], we aim to control suitable linear continuous functionals $J(\cdot)$ of the discretization error e_h associated with the considered finite element approximation. In CFD, examples of $J(\cdot)$ are the lift and drag around bodies in external flows or mean and local values, while in structural mechanics the torsion moment, the stress value or the surface tension are typical goal-quantities. The leading idea of our a posteriori analysis has been to combine the advantages deriving from an error functional control with the richness of information provided by an anisotropic framework.

Let us sketch the procedure used to derive the anisotropic a posteriori error estimator for a general differential problem

$$L(u) = f \quad \text{in } \Omega. \tag{8}$$

In the subsections below we particularize such a procedure to standard model problems in CFD. We refer to [11, 12, 13] for a detailed description of such an approach. First, let us introduce the weak form associated with (8): find $u \in V$ such that

$$a(u, v) = F(v) \quad \text{for any} \quad v \in V, \tag{9}$$

where V is a suitable functional space accounting for the boundary conditions completing the problem at hand, and $a(\cdot,\cdot)$ and $F(\cdot)$ are the bilinear and linear forms corresponding to the differential operator L and the source term f in (8), respectively. The discrete form associated with (9) is obtained by projection onto the space $V_h \subset V$ of continuous piecewise linear finite elements which yields: find $u_h \in V_h$ such that

$$a(u_h, v_h) = F(v_h) \quad \text{for any} \quad v_h \in V_h. \tag{10}$$

As shown in Sects. 3.1 and 3.2, the forms $a(\cdot,\cdot)$ and $F(\cdot)$ have to be suitably stabilized in the case of strong advective/reactive terms, or of the Stokes problems in order to guarantee the absence of spurious oscillations or the well-posedness of the problem, respectively.
By suitably combining the weak form, with $v = v_h$, with the discrete one, we get the well-known Galerkin orthogonality property

$$a(e_h, v_h) = 0 \quad \text{for any} \quad v_h \in V_h, \tag{11}$$

stating the orthogonality of the discretization error $e_h = u - u_h$ with respect to the discrete space V_h.
Let us introduce now the dual problem associated with (9): find $z \in V$ such that

$$a^*(z, \varphi) = J(\varphi) \quad \text{for any} \quad \varphi \in V, \tag{12}$$

where J is a linear continuous functional to be suitably chosen according to the physical quantity to control and $a^*(\cdot,\cdot)$ is the adjoint form to $a(\cdot,\cdot)$, defined by the relation

$a^*(z,\varphi) = a(\varphi, z)$, for any $\varphi \in V$.

We are now in a position to estimate the discretization error associated with the goal quantity, i.e. $J(e_h)$. With this aim, let us first choose in (12) $\varphi = e_h$. Then by exploiting the property of the adjoint form $a^*(\cdot,\cdot)$ and the Galerkin orthogonality property (11) with $v_h = z_h$, we get

$$J(e_h) = a^*(z, e_h) = a(e_h, z) = a(e_h, z - z_h) = F(z - z_h) - a(u_h, z - z_h), \tag{13}$$

where the last equality is due to the weak form (9) with $v = z - z_h$. So far no explicit choice has been made for z_h. Usually, z_h is identified with a suitable interpolant of the dual solution z, according to the regularity of the latter. An integration by parts of the right-hand side of (13) together with a suitable use of anisotropic interpolation error estimates such as those cited in Sect. 2.2, lead to an a posteriori error estimate of the general form

$$|J(e_h)| \leq C \sum_{K \in \mathcal{T}_h} \rho_K(u_h)\, \omega_K(z), \tag{14}$$

where $\rho_K(u_h) = f - L(u_h)$ is the residual associated with the the primal problem (8) and $\omega_K(z)$, which gathers the anisotropic information, depends on the dual solution and weights the residual term. Notice that $\rho_K(u_h)$ measures the error related to the approximation u_h, while the term $\omega_K(z)$ takes into account the propagation of such an error driven by the functional $J(\cdot)$ to control. The terms $\rho_K(u_h)$ and $\omega_K(z)$ in (14) depend on the particular differential problem (8). In the subsections below we explicitly provide two examples of the estimator (14) by considering some standard problems.

3.1 The advection-diffusion-reaction problem

We address the standard scalar advection-diffusion-reaction problem with mixed boundary conditions

$$\begin{cases} L(u) = -\mu \Delta u + \mathbf{a} \cdot \nabla u + \alpha u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma_D, \\ \mu \dfrac{\partial u}{\partial n} = g & \text{on } \Gamma_N \end{cases} \tag{15}$$

where Γ_D and Γ_N are suitable measurable nonoverlapping partitions of the boundary $\partial\Omega$ of Ω with $\Gamma_D \neq \varnothing$ and such that $\partial\Omega = \overline{\Gamma}_D \cup \overline{\Gamma}_N$; the source $f \in L^2(\Omega)$, the diffusivity $\mu \in \mathbb{R}^+$, the advective field $\mathbf{a} \in (W^{1,\infty}(\Omega))^2$, with $\nabla \cdot \mathbf{a} = 0$, the reaction coefficient $\alpha \in L^\infty(\Omega)$ with $\alpha \geq 0$ a.e. in Ω, and $g \in L^2(\Gamma_N)$ are given data, while $\partial u/\partial n = \nabla u \cdot \mathbf{n}$ is the normal derivative of u, $\mathbf{n}$ being the unit outward normal to $\partial\Omega$. As we are interested in advection-reaction dominated problems, we have to discretize (15) by means of a suitable stabilized scheme. The discrete form (10) is thus replaced by the stabilized one

$$a_\tau(u_h, v_h) = F_\tau(v_h) \quad \text{for any} \quad v_h \in V_h. \tag{16}$$

For instance, by choosing a streamline-diffusion scheme [9], the stabilized forms $a_\tau : V \times V \to \mathbb{R}$ and $F_\tau : V \to \mathbb{R}$ for smooth enough functions u and v, are defined as

$$\begin{aligned} a_\tau(u,v) &= \int_\Omega \mu \nabla u \cdot \nabla v \, d\mathbf{x} + \int_\Omega (\mathbf{a} \cdot \nabla u + \alpha u) v \, d\mathbf{x} \\ &\quad + \sum_{K \in \mathcal{T}_h} \int_K \tau_K \left(-\mu \Delta u + \mathbf{a} \cdot \nabla u + \alpha u\right) (\mathbf{a} \cdot \nabla v) \, d\mathbf{x}, \\ F_\tau(v) &= \int_\Omega f v \, d\mathbf{x} + \int_{\Gamma_N} g v \, ds + \sum_{K \in \mathcal{T}_h} \int_K \tau_K f (\mathbf{a} \cdot \nabla v) \, d\mathbf{x}, \end{aligned}$$

where the coefficients τ_K are elementwise stabilizing parameters for which several proposals can be found in the literature (see, e.g., [2, 3, 4, 17]).
Following the procedure described above, we can derive an anisotropic a posteriori error estimator for (15) which can be cast in the form (14) [13]. Let us define the element interior and boundary residuals given by $r_K = (f + \mu \Delta u_h - \mathbf{a} \cdot \nabla u_h - \alpha u_h)|_K$ and

$$j_e = \begin{cases} 0 & \text{if } e \in \Gamma_D, \\ -2 \left(\mu \dfrac{\partial u_h}{\partial n_K} - g \right) & \text{if } e \in \Gamma_N, \\ -\mu \left[\dfrac{\partial u_h}{\partial n_K} \right]_e & \text{if } e \in \mathcal{E}_h^{\text{int}}, \end{cases} \tag{17}$$

respectively. Here $\partial u_h / \partial n_K = \nabla u_h \cdot \mathbf{n}_K$ is the normal derivative of u_h, $\mathbf{n}_K$ is the unit outward normal to ∂K, $\mathcal{E}_h^{\text{int}}$ denotes the set of the internal edges of the skeleton $\mathcal{E}_h$ of the triangulation $\mathcal{T}_h$, and $[\partial u_h / \partial n_K]_e$ stands for the jump of the normal derivative of u_h over the edge $e \subset \partial K$. Then the residual $\rho_K(u_h)$ is given by

$$\rho_K(u_h) = \|r_K(u_h)\|_{L^2(K)} \left(1 + \frac{\tau_K}{\lambda_{2,K}} \|\mathbf{a}\|_{L^\infty(K)} \right) + \frac{1}{2 \lambda_{2,K}^{1/2}} \|j_e\|_{L^2(\partial K)}. \tag{18}$$

Concerning the weight $\omega_K(z)$, by assuming an H^1-regularity for the dual solution z, we identify z_h in (13) with the Clément interpolant of z, thus obtaining

$$\omega_K(z) = \left[\sum_{i=1}^{2} \lambda_{i,K}^2 \left(\mathbf{r}_{i,K}^T G_K(z) \mathbf{r}_{i,K} \right) \right]^{1/2}. \tag{19}$$

Notice that all the anisotropic information $\lambda_{i,K}$ and $\mathbf{r}_{i,K}$ is contained in (19). The a posteriori analysis above has been applied to a more realistic problem in haemodynamics [11]. Moreover, the analysis above covers also the diffusion-reaction problem by letting $\mathbf{a} = \mathbf{0}$ in (15).

3.2 The Stokes problem

Let us consider the standard Stokes problem: seek the velocity $\mathbf{u}$ and the pressure p of an incompressible fluid, subject to mixed boundary conditions:

$$\begin{cases} -\mu\Delta\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega, \\ \nabla\cdot\mathbf{u} = 0 & \text{in } \Omega, \\ \mu(\nabla\mathbf{u})\vec{n} - p\vec{n} = \mathbf{g} & \text{on } \Gamma_N, \\ \mathbf{u} = \mathbf{0} & \text{on } \Gamma_D, \end{cases} \tag{20}$$

where Γ_D, Γ_N and $\vec{n}$ are defined as in Sect. 3.1; the source term $\mathbf{f} \in [L^2(\Omega)]^2$, the viscosity $\mu \in \mathbb{R}^+$, $\mathbf{g} \in [L^2(\Gamma_N)]^2$ are given data. Notice that the differential operator $L(u)$ in (8) is replaced by the operator $L(\mathbf{u}, p)$ given by the left-hand sides of $(20)_1$-$(20)_2$. Moreover, the weak space V in (9) is replaced by the tensor product space $W \times Q$. In order to guarantee the inf-sup condition, the discretization of the Stokes problem requires a stabilized method. By using, for instance, the Galerkin Least Squares method, the stabilized discrete form of (20) becomes: find $(\mathbf{u}_h, p_h)$ in $W_h \times Q_h$, with $W_h \subset W$ and $Q_h \subset Q$ formed by continuous piecewise linear finite elements, such that

$$a_\tau((\mathbf{u}_h, p_h), (\mathbf{v}_h, q_h)) = F_\tau(\mathbf{v}_h, q_h) \quad \text{for any} \quad (\mathbf{v}_h, q_h) \in W_h \times Q_h, \tag{21}$$

where the stabilized forms $a_\tau : [W \times Q]^2 \to \mathbb{R}$ and $F_\tau : W \times Q \to \mathbb{R}$ are given by

$$\begin{aligned} a_\tau((\mathbf{u}, p), (\mathbf{v}, q)) &= \int_\Omega \mu\nabla\mathbf{u} : \nabla\mathbf{v}\, d\mathbf{x} - \int_\Omega p\nabla\cdot\mathbf{v}\, d\mathbf{x} - \int_\Omega q\nabla\cdot\mathbf{u}\, d\mathbf{x} \\ &\quad - \sum_{K\in\mathcal{T}_h} \tau_K \int_K \nabla p\cdot\nabla q\, d\mathbf{x} \\ F_\tau(\mathbf{v}, q) &= \int_\Omega \mathbf{f}\cdot\mathbf{v}\, d\mathbf{x} + \int_{\Gamma_N} \mathbf{g}\cdot\mathbf{v}\, ds - \sum_{K\in\mathcal{T}_h} \tau_K \int_K \mathbf{f}\cdot\nabla q\, d\mathbf{x}. \end{aligned} \tag{22}$$

As we have two unknowns, we can control two continuous linear functionals, the first one $J_1(\cdot)$ associated with the discretization error $\vec{e}_u = \mathbf{u} - \mathbf{u}_h$ of the velocity and the second one $J_2(\cdot)$ related to the discretization error $e_p = p - p_h$ of the pressure. Likewise, we can define both the element interior and boundary residuals associated with the momentum equation $(20)_1$, $r_K^1(\mathbf{u}_h, p_h) = (\mathbf{f} + \mu\Delta\mathbf{u}_h - \nabla p_h)|_K$ and

$$\mathcal{J}_e = \begin{cases} \mathbf{0} & \text{if } e \in \Gamma_D, \\ 2\,(\mathbf{g} - (\mu(\nabla\mathbf{u}_h\vec{n}_K) - p_h\vec{n}_K)) & \text{if } e \in \Gamma_N, \\ -[(\mu(\nabla\mathbf{u}_h\vec{n}_K) - p_h\vec{n}_K)]_e & \text{if } e \in \mathcal{E}_h^{\text{int}}, \end{cases} \tag{23}$$

respectively, and the interior residual $r_K^2(\mathbf{u}_h) = (\nabla\cdot\mathbf{u}_h)|_K$ related to the continuity equation $(20)_2$. Estimate (14) is thus replaced by the new one

$$|J_1(\vec{e}_u) + J_2(e_p)| \leq C \sum_{K \in \mathcal{T}_h} \left(\rho_K^1(\mathbf{u}_h, p_h)\, \omega_K^1(\vec{w}) + \rho_K^2(\mathbf{u}_h, p_h)\, \omega_K^2(r)\right), \tag{24}$$

with $C = C(M, \widehat{C}, \widehat{K})$ and $(\vec{w}, r)$ the dual velocity-pressure pair, while

$$\begin{aligned}
\rho_K^1(\mathbf{u}_h, p_h) &= ||r_K^1(\mathbf{u}_h, p_h)||_{L^2(K)} + \frac{1}{2}||\mathcal{J}_e||_{L^2(\partial K)} \left(\frac{\lambda_{1,K}^2 + \lambda_{2,K}^2}{\lambda_{2,K}^3}\right)^{1/2}, \\
\rho_K^2(\mathbf{u}_h, p_h) &= ||r_K^2(\mathbf{u}_h)||_{L^2(K)} + \frac{\tau_K}{\lambda_{2,K}} ||r_K^1(\mathbf{u}_h, p_h)||_{L^2(K)}, \\
\omega_K^1(\vec{w}) &= \Big[\sum_{i,j=1}^{2} \lambda_{i,K}^2 \lambda_{j,K}^2 L_K^{i,j}(\vec{w})\Big]^{1/2}, \omega_K^2(r) = \Big[\sum_{i=1}^{2} \lambda_{i,K}^2 \Big(\mathbf{r}_{i,K}^T G_K(r) \mathbf{r}_{i,K}\Big)\Big]^{1/2},
\end{aligned} \tag{25}$$

where $L_K^{i,j}(\vec{w})$ is the straightforward extension to vector-valued functions of the term (3). We point out that (24) consists of contributions associated with the error propagation due to both the dual velocity and the dual pressure.

4 Numerical results

A typical numerical solution process of a given problem consists of an adaptive iterative procedure based on a metric-based approach. Starting from the a posteriori error estimate, a second-order tensor field, embedding the information about the mesh spacing and stretching, is defined on the actual mesh and employed for the generation of the new mesh, as described in [13]. The software BAMG [14] has been used for this purpose.

In this section we address the numerical solution of some test cases. In more detail, we consider the advection-diffusion-reaction problem (15) and we show the effectiveness of the adaptive algorithm for the construction of an "optimal " mesh, e.g., the mesh for which we have maximum accuracy for a given number of degrees of freedom.

The "glass" test case

Let us define $r = \sqrt{(x_1 - 1/2)^2 + (x_2 - 1/2)^2}$ while choosing in (15), $\Omega = (0,1)^2$, $\mu = 10^{-4}$, $f = 1$ for $1/5 < r < 1/4$ and zero elsewhere, $\mathbf{a} = (x_2 - 1/2, -(x_1 - 1/2))^T$, $\alpha = 100$ for $r < 1/5$ and zero elsewhere, and $\Gamma_N = \varnothing$. The solution u exhibits a strong internal circular layer in the region $1/5 < r < 1/4$ and a large gradient in the radial direction in the region $1/4 < r < 1$. With reference to Fig. 3, the adaptive process starts from a uniform mesh (top-left) and is stopped after two iterations, yielding the meshes at the top-center and top-right. The numerical solutions on the initial mesh, and on the other two meshes are displayed in the bottom line. The functional $J(\cdot)$ has been chosen as $J(v) = a_0(v, u)$ for any $v \in V$, where the subscript 0 refers to the nonstabilized bilinear form derived from (16). This choice allows us to control the energy norm of the discretization error, as $J(u - u_h) = a_0(u - u_h, u) = a_0(u - u_h, u - u_h)$, thanks to the Galerkin orthogonality property. All the main directional features characterizing the solution u are well captured by the anisotropic error estimator

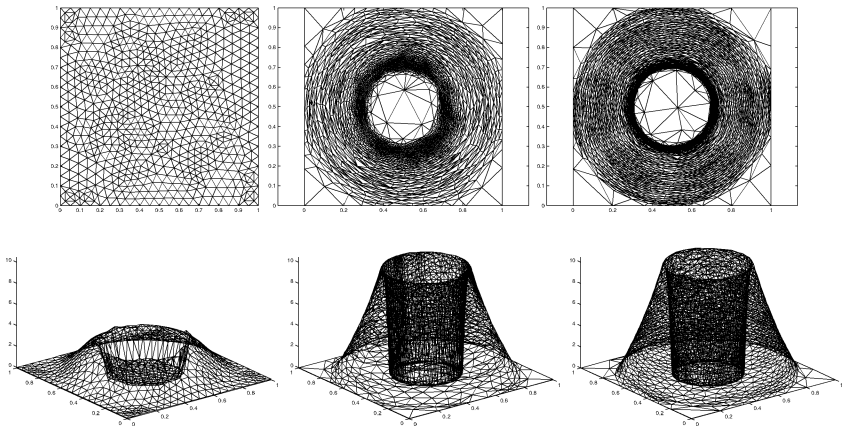


Fig. 3. Sequence of meshes (top) and corresponding solutions (bottom)

as the mesh elements are stretched along the direction of the layers. Table 1 (left) collects the information about the number of elements and of nodes for the three meshes.

Table 1. Degrees of freedom: the "glass" (left) and "channel" (right) test cases

# elements	# nodes
1312	697
4203	2123
5838	2939

# elements	# nodes
1016	561
3763	1950
3991	2083

The "channel" test case

Let Ω in (15) be the U-shaped domain given by the square $(-1,1)^2$ from which the rectangle $(-1,0) \times (-0.4,0.4)$ has been cut, and $\mu = 10^{-4}$, $f = 0$, $\mathbf{a} = (x_2, -x_1)^T$, $\alpha = 0$, and $\Gamma_N = \varnothing$. The nonhomogeneous Dirichlet datum takes the value 1 on the sides $(\{x_1 = -1\} \cap \{0.4 \le x_2 \le 1\}) \cup (\{x_2 = 0.4\} \cap \{-1 \le x_1 \le -0.5\})$. The solution shows two circular-shaped internal layers, a boundary layer near the top-left corner at $x_2 = 1$, and an outflow boundary layer at $x_2 = -0.4$. With reference to Fig. 4, the adaptive process starts from a uniform mesh (top-left) and is iterated two more times, with corresponding grids shown at top-center and top-right. The numerical solutions on the initial mesh and on the two adapted meshes are displayed in the bottom line. The functional $J(\cdot)$ has been chosen as in the previous example. Notice how all the layers are well represented on the last grid, though some oscillation is still polluting the numerical solution. Table 1 (right) summarizes the information about the number of elements and of nodes for the sequence of meshes.

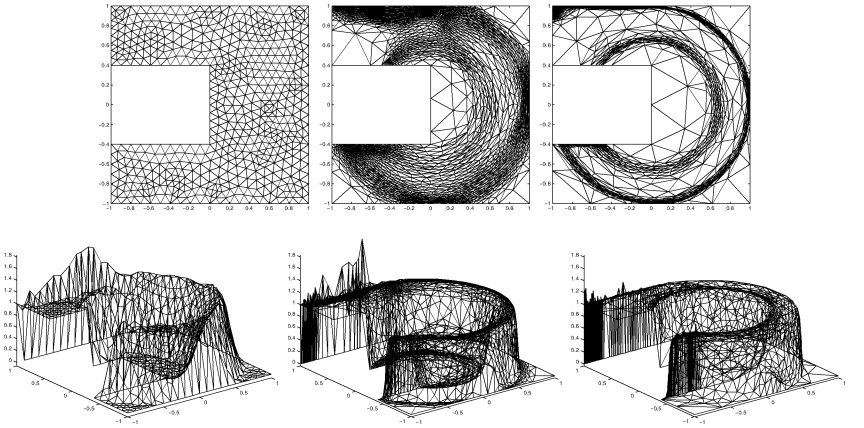


Fig. 4. Sequence of meshes (top) and corresponding solutions (bottom)

5 Acknowledgments

This work has been supported by the project MIUR 2001 “Numerical Methods in Fluid Dynamics and Electromagnetism”. We thank Prof. Luca Formaggia for useful comments and suggestions over our enduring collaboration.

References

1. Apel, T.: Anisotropic Finite Elements: Local Estimates and Applications. Book Series: Advances in Numerical Mathematics, Teubner, Stuttgart (1999)
2. Apel, L., Lube, G.: Anisotropic mesh refinement in stabilized Galerkin methods. Numer. Math., **74**, 261–282 (1996)
3. Becker, R., Rannacher, R.: An optimal control approach to a posteriori error estimation in finite element methods. Acta Numerica, **10**, 1–102 (2001)
4. Brezzi, F., Russo, A.: Choosing bubbles for advection-diffusion problems. Math. Models Methods Appl. Sci., **4**, 571–587 (1994)
5. Ciarlet, Ph.: The Finite Element Method for Elliptic Problems. North-Holland Publishing Company, Amsterdam (1978)
6. Clément, Ph.: Approximation by finite element functions using local regularization. RAIRO Anal. Numér., **2**, 77–84 (1975)
7. Courty, F., Leservoisier, D., George, P.L., Dervieux, A.: Continuous metrics and mesh optimization. Submitted for publication in Appl. Numer. Math. (2003)
8. Darmofal, D.L., Venditti, D.A.: Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. J. Comput. Phys., **187**, 22–46 (2003)
9. Eriksson, K., Johnson, C.: Adaptive streamline diffusion finite element methods for stationary convection-diffusion problems. Math. Comput., **60**, 167–188 (1993)
10. Formaggia, L., Perotto, S.: New anisotropic a priori error estimates. Numer. Math., **89**, 641–667 (2001)

11. Formaggia, L., Perotto, S., Zunino, P.: An anisotropic a-posteriori error estimate for a convection-diffusion problem. Comput. Visual. Sci., **4**, 99–104 (2001)
12. Formaggia, L., Perotto, S.: Anisotropic error estimates for elliptic problems. Numer. Math., **94**, 67–92 (2003)
13. Formaggia, L., Micheletti, S., Perotto, S.: Anisotropic mesh adaptation in Computational Fluid Dynamics: application to the advection-diffusion-reaction and the Stokes problems. Submitted for publication in Mathematics and Computers in Simulation (2003)
14. Hecht, F.: BAMG: bidimensional anisotropic mesh generator. http://www-rocq.inria.fr/gamma/cdrom/www/bamg/eng.htm (1998)
15. Kunert, G.: A Posteriori Error Estimation for Anisotropic Tetrahedral and Triangular Finite Element Meshes. Ph.D. thesis, Fakultät für Mathematik der Technischen Universität Chemnitz, Chemnitz (1999)
16. Lions, J.L., Magenes, E.: Non-Homogeneous Boundary Value Problems and Applications. Volume I, Springer-Verlag, Berlin (1972)
17. Micheletti, S., Perotto, S., Picasso, M.: Stabilized finite elements on anisotropic meshes: a priori error estimates for the advection-diffusion and Stokes problems. SIAM J. Numer. Anal., **41** No. 3, 1131–1162 (2003)
18. Picasso, M.: An anisotropic error indicator based on Zienkiewicz-Zhu error estimator: application to elliptic and parabolic problems. SIAM J. Sci. Comput., **24** No. 4, 1328–1355 (2003)
19. Siebert, K.G.: An a posteriori error estimator for anisotropic refinement. Numer. Math., **73**, 373–398 (1996)

A Posteriori Error Estimation and Mesh Adaptivity for Finite Volume and Finite Element Methods

Timothy J. Barth

NASA Ames Research Center, Moffett Field, CA 94035, USA
barth@nas.nasa.gov

Summary. Error representation formulas and *a posteriori* error estimates for numerical solutions of hyperbolic conservation laws are considered with specialized variants given for the Godunov finite volume and discontinuous Galerkin finite element methods. The error representation formulas utilize the solution of a dual problem to capture the nonlocal error behavior present in hyperbolic problems. The error representation formulas also provide a framework for understanding superconvergence properties of functionals and fundamental differences between finite element and Godunov finite volume methods. Computable error estimates are then constructed for practical implementation in computer codes. The error representation formulas and computable error estimates also suggest a straightforward strategy for mesh adaptivity which is demonstrated on numerical hyperbolic problems of interest.

Key words: *A posteriori* error estimates, error representation, Godunov finite volume methods, finite element methods, unstructured meshes.

1 Introduction

In the numerical simulation of partial differential equations, a frequently encountered objective in these simulations is the subsequent calculation of certain derived quantities of particular interest, e.g., aerodynamic lift and drag coefficients, stress intensity factors, mean temperatures, etc. The ability to estimate the error in such derived quantities (mathematically described as functionals) and modify the calculation procedure via adaptivity to efficiently decrease this error provides a systematic approach to improved reliability and efficiency of numerical simulations.

For an introduction to *a posteriori* error analysis of functionals see the articles by Becker and Rannacher [BR98], Eriksson et al. [EEHJ95], Giles et al. [GLLS97, GP99], Johnson et al. [JRB95], Prudhomme and Oden [OP99, PO99], Süli [S̈98], the collected NATO lecture notes [BD02] and the multitude of additional references contained therein. The main goal of this work is to provide a brief introduction to these general theories for nonlinear conservation laws and to relate specialized theories for the Godunov finite volume method as described in Barth and

Larson [BL02] and the discontinuous Galerkin method as described in Larson and Barth [LB99] as well as Hartmann and Houston [HH02]. Comparison of these theories exposes important differences between finite element and finite volume methods with respect to the error representation of functionals. This is due to the absence of full Galerkin orthogonality in Godunov finite volume methods, namely that element (cell) residuals in the finite element method are orthogonal to a much larger space of functions than are residuals in the finite volume method. Consequently, finite element and finite volume methods with identical rates of convergence for global error measures can have dramatically different rates of convergence for derived functionals. Another consequence is that the dual (adjoint) problem used in the error representation formula for functionals can be approximated in Godunov finite volume methods using the same order method (same space of reconstructed functions) as used in the primal problem. Attempting this same strategy using the finite element methods considered herein fails completely since by Galerkin orthogonality the estimated error is identically zero. For the finite element method, the dual problem must be approximated in a larger space of functions than used in the primal numerical method.

2 Background

Consider the following system of m first-order conservation laws in a domain $\Omega \subset \mathbf{R}^d$ with boundary Γ

$$\begin{cases} \sum_{i=1}^{d} f^i_{,x_i}(u) = 0 \ , & \text{for } x \text{ in } \Omega \\ A^-(g-u)|_\Gamma = 0 \ , & \text{for } x \text{ on } \Gamma \end{cases} \tag{1}$$

where $u(x) : \mathbf{R}^d \mapsto \mathbf{R}^m$ denotes the vector of conserved variables, $f^i(u) : \mathbf{R}^m \mapsto \mathbf{R}^m, i = 1, \ldots, d$ the flux vector components, and $A \equiv \sum_{i=1}^{d} n_i f^i_{,u}$ the flux jacobian matrix associated with a direction, n, normal to Γ. In the present discussion, only spatial derivatives are considered but more generally x could include a time coordinate without introducing any new complication in the abstract error representation formulas given below.

Let $\mathcal{K}$ be a partition of a polygonal domain Ω into non-overlapping shape regular elements (or control volumes) denoted by K. Furthermore, consider two finite-dimensional spaces of piecewise polynomials with differing degrees of interelement continuity. The first space, $\mathcal{V}_{h,p}$, is the standard finite element space of piecewise polynomials of complete degree p with C^0 continuity between elements

$$\mathcal{V}_{h,p} = \{v : v \in C^0(\Omega), v|_K \in \mathcal{P}_p(K) \ , \ \forall K \in \mathcal{K}\} \tag{2}$$

with $\mathcal{P}_p(K)$ the space of polynomials of degree $\leq p$ defined on an element K. The second space, $\mathcal{V}^{\mathrm{B}}_{h,p}$, is the mesh dependent broken space of piecewise polynomials of complete degree p in each K with no continuity between elements

$$\mathcal{V}^{\mathrm{B}}_{h,p} = \{v : v|_K \in \mathcal{P}_p(K) \ , \ \forall K \in \mathcal{K}\} \ . \tag{3}$$

Using this latter broken space, two seemly different methods are considered. The first method is the the discontinuous Galerkin (DG) finite element method introduced by

Reed and Hill [RH73] as analyzed by Johnson and Pitkäranta [JP86] and further refined for nonlinear conservation laws by Cockburn et al. [CLS89, CS97].

Discontinuous Galerkin FEM. Find $u_h \in \mathcal{V}_{h,p}^{\mathrm{B}}$ such that

$$\mathcal{B}_{\mathrm{DG}}(u_h, v) = F(v), \quad \forall v \in \mathcal{V}_{h,p}^{\mathrm{B}} \tag{4}$$

where

$$\begin{aligned}\mathcal{B}_{\mathrm{DG}}(u_h, v) - F(v) = \sum_{K\in\mathcal{K}} \Big(&- \int_K \sum_{i=1}^{d} v_{,x_i} \cdot f^i(u_h)\, dx \\ &+ \int_{\partial K\setminus\Gamma} v_- \cdot h(n; (u_h)_-, (u_h)_+)\, ds + \int_{\partial K\cap\Gamma} v_- \cdot h(n; (u_h)_-, g)\, ds \Big)\end{aligned} \tag{5}$$

where $h(n; u_-, u_+)$ is a numerical flux function such that $\sum_{i=1}^{d} n_i f^i(u) = h(n; u, u)$ and $h(n; u_-, u_+) = -h(-n; u_+, u_-)$.

The second method considered is the generalization of Godunov's original method [God59] to higher order accuracy via various forms of data reconstruction, e.g. MUSCL in [vL79], TVD in [Har83], UNO in [HOEC87], ENO in [Har89], and further generalization to unstructured meshes given in [BJ89, BF90, DOE90, Bar98, Abg94, Van93]. Recently, in Barth and Larson [BL02] the generalized Godunov finite volume method was shown equivalent to a particular Petrov-Galerkin variant of the discontinuous Galerkin method:

Higher Order Godunov FVM. Find $u_0 \in \mathcal{V}_{h,0}^{\mathrm{B}}$ such that

$$\mathcal{B}_{\mathrm{DG}}(R_p^0 u_0, v) = F(v), \quad \forall v \in \mathcal{V}_{h,0}^{\mathrm{B}}, \quad R_p^0 : \mathcal{V}_{h,0}^{\mathrm{B}} \mapsto \mathcal{V}_{h,p}^{\mathrm{B}} \tag{6}$$

where $\mathcal{B}_{\mathrm{DG}}$ is the same semilinear form used in the discontinuous Galerkin method and R_p^0 is any patchwise reconstruction operator that maps the broken space of piecewise constants to the broken space of piecewise polynomials of complete degree p.

Using either of these methods, the objective is to estimate the error in a user specified functional $M(u)$ which can be expressed as a weighted integration over the domain Ω

$$M_\psi(u) = \int_\Omega \psi \cdot N(u)\, dx$$

or a weighted integration on the boundary Γ

$$M_\psi(u) = \int_\Gamma \psi \cdot N(u)\, dx$$

for some user specified weighting function $\psi(x) : \mathbf{R}^d \mapsto \mathbf{R}^m$ and linear/nonlinear function $N(u) : \mathbf{R}^m \mapsto \mathbf{R}^m$. By an appropriate choice of $\psi(x)$ and $N(u)$, it is possible to devise functionals of practical engineering use, e.g. lift and drag forces on a body, stress intensity factors, average quantities, etc.

In the remainder of this article, the connection between local error and nonlocal cell residuals is given. This is accomplished through the introduction of a dual (adjoint) problem. Error representation formulas using these dual problems are then constructed for the following quantities of interest:

- (Finite Element) $M_\Psi(u) - M_\Psi(u_h)$ where $u_h \in \mathcal{V}_{h,p}^{\mathrm{B}}$.
- (Godunov Finite Volume) $M_\Psi(u) - M_\Psi(R_p^0 u_0)$ with $u_0 \in \mathcal{V}_{h,0}^{\mathrm{B}}$ where $R_p^0 u_0$ is the reconstructed data in the Godunov finite volume method.

With error representation formulas in hand, superconvergence properties of certain functionals is briefly examined. This identifies some distinct differences between finite element and Godunov finite volume methods. The error representation formulas suggest simplified error estimates and a strategy for mesh adaptivity which is demonstrated on numerical problems of interest.

3 Error Representation for Hyperbolic Problems

In this section, the notion of local error representation for hyperbolic problems is revisited. Consider the primal scalar hyperbolic problem

$$\begin{cases} \mathcal{L}u = f & \text{for } x \text{ in } \Omega \\ u|_\Gamma = g & \text{for } x \text{ on } \Gamma^- \end{cases} . \tag{7}$$

For illustrative purposes, let $\mathcal{L}$ denote the advection operator

$$\mathcal{L}u \equiv \lambda \cdot \nabla u \tag{8}$$

with $\lambda(x) : \mathbf{R}^d \mapsto \mathbf{R}^d$. In this particular case, the inflow boundary is defined by

$$\Gamma^- = \{x \mid x \in \Gamma \text{ and } \lambda \cdot n < 0\} \tag{9}$$

with n the exterior normal vector on Γ. Next introduce the Green's function $G(\xi;x)$ satisfying adjoint problem

$$\begin{cases} \mathcal{L}^* G(\xi;x) = \delta(x-\xi) & \text{for } x \text{ in } \Omega \\ G(\xi;x)|_\Gamma = 0 & \text{for } x \text{ on } \Gamma^+ \end{cases} \tag{10}$$

where $\mathcal{L}^*$ is the adjoint operator and Γ^+ the outflow boundary, $\Gamma^+ \equiv \Gamma \setminus \Gamma^-$. The Green's function quantifies the connection between the *local* solution error and *nonlocal* residuals. Let $(u,v)_\Omega \equiv \int_\Omega u \cdot v\, dx$ and consider the solution error at a point $\xi \in \Omega$ for $u_h \in \mathcal{V}_{h,p}$

$$\begin{aligned} (u_h - u)(\xi) &= (u_h - u, \delta(x-\xi))_\Omega \\ &= (u_h - u, \mathcal{L}^* G(\xi;x))_\Omega \\ &= (\mathcal{L}(u_h - u), G(\xi;x))_\Omega \\ &= (R(u_h), G(\xi,x))_\Omega \end{aligned} \tag{11}$$

where $R(u_h) = \mathcal{L}u_h - f$ is the numerical residual. Absent from the final right-hand side equation is the exact solution u. This latter formula reveals the dependence of pointwise error on nonlocal numerical residuals. The error at a point ξ is a Green's function weighted combination of numerical residual errors integrated over the domain.

In the PDE system generalization, the associated Green's function components can become quite complicated (and counterintuitive) so that heuristic error estimation methods are more likely to fail. For example, Fig. 1 shows Green's function components for steady ideal magnetohydrodynamic flow with a velocity V and velocity aligned magnetic induction field B. Figure 1 (left) shows isodensity contours of the numerically computed Green's function for a uniform super Alfvén MHD flow indicating how numerical errors at a point in the center of the domain depend on local residual errors occurring in an *upstream* domain of dependence associated with the streamline and the fast magnetoacoustic characteristic cone. Decreasing the velocity magnitude while keeping the magnetic field fixed in this problem eventually gives rise to slow magnetoacoustic "forward inclined" waves (not found in hydrodynamics) so that now the same error components at a point in the center of the domain depend on residual errors occurring both *upstream* along the streamline and *downstream* along a cone associated with slow magnetoacoustic waves.

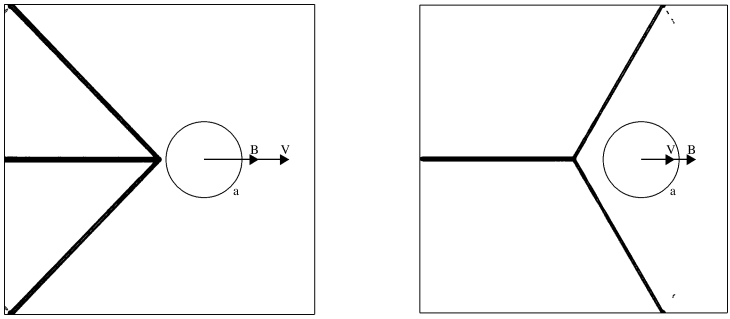


Fig. 1. Isocontours of the numerically computed Green's function density component for the 2-D steady ideal magnetohydrodynamic equations for a point ξ located at the center of the unit square domain. Streamline and fast magnetoacoustic wave front (left figure) corresponding to super Alfvén flow $|V|/c = 2, |B|/(c\sqrt{\rho}) = 1.2$. Streamline and slow magnetoacoustic forward inclined wave front (right figure), $|V|/c = .88, |B|/(c\sqrt{\rho}) = 1.2$.

4 Error Representation Formulas for Functionals

In this section, exact error representation formulas are derived for three abstract formulations with
(1) $B_{\mathrm{DG}}(\cdot,\cdot)$ a bilinear form with $M(\cdot)$ a linear functional.
(2) $\mathcal{B}_{\mathrm{DG}}(\cdot,\cdot)$ a semilinear form (nonlinear in the first argument and linear in the second argument) with $\mathcal{M}(\cdot)$ a nonlinear functional.
(3) $\mathcal{B}_{\mathrm{DG}}(R_p^0\cdot,\cdot)$ a semilinear form with $\mathcal{M}(\cdot)$ a nonlinear functional.

Remark 1. To simplify the derivation of the error representation formulas, all boundary conditions are assumed weakly enforced rather than equipping the trial space

with strong boundary condition data. This technique avoids the tedious explicit calculation of strong boundary conditions for the dual problem.

4.1 DG FEM Error Representation: The Linear Case

Let $B_{\mathrm{DG}}(\cdot,\cdot)$ denote a bilinear form and $M(\cdot)$ a linear functional. In the following derivations, π_h denotes any suitable projection operator (e.g. interpolation, L_2 projection) into $\mathcal{V}_{h,p}^{\mathrm{B}}$. Begin by introducing the primal numerical method assuming all boundary conditions are weakly enforced.

Primal numerical problem: *Find* $u_h \in \mathcal{V}_{h,p}^{\mathrm{B}}$ *such that*

$$B_{\mathrm{DG}}(u_h, v) = F(v) \quad \forall\, v \in \mathcal{V}_{h,p}^{\mathrm{B}}$$

with the Galerkin orthogonality condition

$$B_{\mathrm{DG}}(u - u_h, v) = 0 \quad \forall\, v \in \mathcal{V}_{h,p}^{\mathrm{B}} \ .$$

Next, we introduce the auxiliary dual problem utilizing infinite-dimensional trial and test spaces.

Dual problem: *Find* $\Phi \in \mathcal{V}^{\mathrm{B}}$ *such that*

$$B_{\mathrm{DG}}(v, \Phi) = M(v) \quad \forall\, v \in \mathcal{V}^{\mathrm{B}} \ .$$

An exact error representation formula for a given functional $M(\cdot)$ results from the following steps

$$\begin{aligned}
M(u) - M(u_h) &= M(u - u_h) && \text{(linearity of } M\text{)} \\
&= B_{\mathrm{DG}}(u - u_h, \Phi) && \text{(dual problem)} \\
&= B_{\mathrm{DG}}(u - u_h, \Phi - \pi_h\Phi) && \text{(orthogonality)} \\
&= B_{\mathrm{DG}}(u, \Phi - \pi_h\Phi) - B_{\mathrm{DG}}(u_h, \Phi - \pi_h\Phi) && \text{(linearity of } B\text{)} \\
&= F(\Phi - \pi_h\Phi) - B_{\mathrm{DG}}(u_h, \Phi - \pi_h\Phi) && \text{(primal problem)}
\end{aligned}$$

so in summary

$$M(u) - M(u_h) = F(\Phi - \pi_h\Phi) - B_{\mathrm{DG}}(u_h, \Phi - \pi_h\Phi) \ . \tag{12}$$

Notably absent from the right-hand side of this equation is any dependence on the exact solution u.

4.2 DG FEM Error Representation: The Nonlinear Case

Let $\mathcal{B}_{\mathrm{DG}}(\cdot,\cdot)$ denote a semilinear form and $\mathcal{M}(\cdot)$ a nonlinear functional. To cope with nonlinearity, mean-value linearization is employed

$$\begin{aligned}\mathcal{B}_{\mathrm{DG}}(u,v) &= \mathcal{B}_{\mathrm{DG}}(u_h,v) + \overline{\mathcal{B}}_{\mathrm{DG}}(u_h,u;u-u_h,v) \quad \forall\, v \in \mathcal{V}^{\mathrm{B}} \\ \mathcal{M}(u) &= \mathcal{M}(u_h) + \overline{\mathcal{M}}(u_h,u;u-u_h) \ .\end{aligned}$$

For example, if $\mathcal{B}(u,v) = (\mathcal{L}u,v)$ for some nonlinear differential operator $\mathcal{L}$ then for $v \in \mathcal{V}$

$$\begin{aligned}\mathcal{B}(u,v) &= \mathcal{B}(u_h,v) + \left(\int_0^1 \mathcal{L}_{,u}(\tilde{u}(\theta))\, d\theta\ (u-u_h), v\right) \\ &= \mathcal{B}(u_h,v) + (\overline{\mathcal{L}}_{,u}\,(u-u_h), v) \\ &= \mathcal{B}(u_h,v) + \overline{\mathcal{B}}(u_h,u;u-u_h,v).\end{aligned}$$

with $\tilde{u}(\theta) \equiv u_h + (u-u_h)\,\theta$. For brevity, the dependence of $\overline{\mathcal{B}}$ on the path integration involving the exact solution u will be notationally suppressed. We then proceed in the same fashion as in the previous example.

Primal numerical problem: *Find* $u_h \in \mathcal{V}^{\mathrm{B}}_{h,p}$ *such that*

$$\mathcal{B}_{\mathrm{DG}}(u_h,v) = F(v) \quad \forall\, v \in \mathcal{V}^{\mathrm{B}}_{h,p} \tag{13}$$

with orthogonality condition for the linearized form

$$\overline{\mathcal{B}}_{\mathrm{DG}}(u-u_h,v) = 0 \quad \forall\, v \in \mathcal{V}^{\mathrm{B}}_{h,p} \ .$$

A mean-value linearized dual problem is then introduced which utilizes infinite-dimensional trial and test spaces.

Linearized dual problem: *Find* $\Phi \in \mathcal{V}^{\mathrm{B}}$ *such that*

$$\overline{\mathcal{B}}_{\mathrm{DG}}(v,\Phi) = \overline{\mathcal{M}}(v) \quad \forall\, v \in \mathcal{V}^{\mathrm{B}} \ . \tag{14}$$

An exact error representation formula for a given nonlinear functional $\mathcal{M}(\cdot)$ then results from the following steps

$$\begin{aligned}\mathcal{M}(u) - \mathcal{M}(u_h) &= \overline{\mathcal{M}}(u-u_h) && \text{(mean-value } \overline{\mathcal{M}}\text{)} \\ &= \overline{\mathcal{B}}_{\mathrm{DG}}(u-u_h,\Phi) && \text{(dual problem)} \\ &= \overline{\mathcal{B}}_{\mathrm{DG}}(u-u_h,\Phi-\pi_h\Phi) && \text{(orthogonality)} \\ &= \mathcal{B}_{\mathrm{DG}}(u,\Phi-\pi_h\Phi) - \mathcal{B}_{\mathrm{DG}}(u_h,\Phi-\pi_h\Phi) && \text{(mean-value } \overline{\mathcal{B}}\text{)} \\ &= F(\Phi-\pi_h\Phi) - \mathcal{B}_{\mathrm{DG}}(u_h,\Phi-\pi_h\Phi), && \text{(primal problem)}\end{aligned}$$

so in summary

$$\mathcal{M}(u) - \mathcal{M}(u_h) = F(\Phi-\pi_h\Phi) - \mathcal{B}_{\mathrm{DG}}(u_h,\Phi-\pi_h\Phi) \ . \tag{15}$$

Note that although Eqns. (12) and (15) appear identical, mean-value linearization introduces a subtle right-hand side dependency on the exact solution in Eqn. (15).

4.3 Godunov FVM Error Representation

$\mathcal{B}_{\rm DG}(R_p^0\cdot,\cdot)$ and $\mathcal{M}(\cdot)$ are both assumed nonlinear. Mean-value linearizations are introduced as in the previous case

$$\begin{aligned}\mathcal{B}_{\rm DG}(u,v) &= \mathcal{B}_{\rm DG}(R_p^0 u_0, v) + \overline{\mathcal{B}}_{\rm DG}(u - R_p^0 u_0, v) \quad \forall\, v \in \mathcal{V}^{\rm B}\\ \mathcal{M}(u) &= \mathcal{M}(R_p^0 u_0) + \overline{\mathcal{M}}(u - R_p^0 u_0)\end{aligned}$$

for $u_0 \in \mathcal{V}_{h,0}^{\rm B}$ where the mean-value linearized forms again depend on a path integration involving the exact solution.

Primal Godunov FVM problem: *Find* $u_0 \in \mathcal{V}_{h,0}^{\rm B}$ *such that*

$$\mathcal{B}_{\rm DG}(R_p^0 u_0, v) = F(v) \quad \forall\, v \in \mathcal{V}_{h,0}^{\rm B} \tag{16}$$

with the orthogonality condition

$$\overline{\mathcal{B}}_{\rm DG}(u - R_p^0 u_0, v) = 0 \quad \forall\, v \in \mathcal{V}_{h,0}^{\rm B}\ .$$

Next, introduce the infinite-dimensional linearized dual problem.

Linearized dual problem: *Find* $\Phi \in \mathcal{V}^{\rm B}$ *such that*

$$\overline{\mathcal{B}}_{\rm DG}(v, \Phi) = \overline{\mathcal{M}}(v) \quad \forall\, v \in \mathcal{V}^{\rm B}\ . \tag{17}$$

An exact error representation formula for a given nonlinear functional $\mathcal{M}(\cdot)$ for the class of Godunov finite volume methods results from the following steps

$$\begin{aligned}\mathcal{M}(u) - \mathcal{M}(R_p^0 u_0) &= \overline{\mathcal{M}}(u - R_p^0 u_0) && \text{(mean-value } \overline{\mathcal{M}}\text{)}\\ &= \overline{\mathcal{B}}_{\rm DG}(u - R_p^0 u_0, \Phi) && \text{(dual problem)}\\ &= \overline{\mathcal{B}}_{\rm DG}(u - R_p^0 u_0, \Phi - \pi_0\Phi) && \text{(orthogonality)}\\ &= \mathcal{B}_{\rm DG}(u, \Phi - \pi_0\Phi) - \mathcal{B}_{\rm DG}(R_p^0 u_0, \Phi - \pi_0\Phi) && \text{(mean-value } \overline{\mathcal{B}}\text{)}\\ &= F(\Phi - \pi_0\Phi) - \mathcal{B}_{\rm DG}(R_p^0 u_0, \Phi - \pi_0\Phi), && \text{(primal problem)}\end{aligned}$$

with π_0 any projection into $\mathcal{V}_{h,0}$ thus yielding the following exact error representation formula

$$\mathcal{M}(u) - \mathcal{M}(R_p^0 u_0) = F(\Phi - \pi_0\Phi) - \mathcal{B}_{\rm DG}(R_p^0 u_0, \Phi - \pi_0\Phi)\ . \tag{18}$$

Again, it should be noted that the right-hand side has a subtle dependence on the exact solution through the mean-value linearization used in the dual problem.

4.4 Superconvergence of Functionals

In this section, convergence rates for functionals is examined. Recall the error representation formulas for the discontinuous Galerkin and Godunov finite volume methods.

Discontinuous Galerkin Finite Element

$$\mathcal{M}(u) - \mathcal{M}(u_h) = F(\Phi - \pi_h\Phi) - \mathcal{B}_{\mathrm{DG}}(u_h, \Phi - \pi_h\Phi) \tag{19}$$

Godunov Finite Volume Method

$$\mathcal{M}(u) - \mathcal{M}(R_p^0 u_0) = F(\Phi - \pi_0\Phi) - \mathcal{B}_{\mathrm{DG}}(R_p^0 u_0, \Phi - \pi_0\Phi) \ . \tag{20}$$

A notable difference between the discontinuous Galerkin method and the Godunov finite volume method comes from the orthogonality condition used in the derivation of the error representation formulas. In the finite element method, the error $u - u_h$ is Galerkin orthogonal to all test functions in $\mathcal{V}_{h,p}$ with respect to the bilinear form $\overline{\mathcal{B}}_{\mathrm{DG}}(\cdot,\cdot)$. In the finite volume method, the error $u - R_p^0 u_0$ is only orthogonal to constant test functions with respect to the same bilinear form. So even if the convergence rates for global error measures using the finite volume and finite element methods are the same, the convergence rates for functionals can be quite different. In the setting of linear advection-diffusion problems, the superconvergence theory for functionals is understood. For example, Süli and Houston in [BD02] give the convergence theory of functionals for the streamline diffusion discretization of scalar hyperbolic problems. For problems with sufficiently smooth primal and dual solutions, the basic theoretical result for streamline diffusion states that if the primal method converges at the rate $O(h^{p+1/2})$ then functionals converge at the rate $O(h^{2p+1})$. When diffusion terms are added to the PDE the convergence rate of functionals becomes $O(h^{2p})$.

The analysis of functionals for the discontinuous Galerkin method is a trivial extension of the streamline diffusion analysis. Consider the scalar advection problem given in (7-8). The well-known *a priori* theory [JP86, Joh87] for the discontinuous Galerkin method (with interior stabilization added only for theoretical analysis purposes) gives the following convergence result

$$|||u - u_h|||^2 \leq h^{2s+1}|u|^2_{H^{s+1}(\Omega)}$$

where

$$|||v||| = \sum_{K\in\mathcal{K}} \left(h\|\mathcal{L}v\|^2_K + \|v\|^2_{\partial K^-\cap\Gamma^-} + \frac{1}{2}\|v\|^2_{\partial K^+\cap\Gamma^+} + \frac{1}{2}\|[v]^+_-\|^2_{\partial K^-\setminus\Gamma} \right)$$

for $0 \leq s \leq p$. The discontinuous Galerkin method for (7-8) then reduces to the following problem:

Find $u_h \in \mathcal{V}^B_{h,p}$ *such that* $\forall v \in \mathcal{V}_{h,p}$

$$\sum_{K\in\mathcal{K}} (\mathcal{L}u_h - f, v + \delta_h \mathcal{L}v)_K + \langle (\lambda\cdot n)^- [u]^+_-, v_-\rangle_{\partial K\setminus\Gamma} + \langle (\lambda\cdot n)^- (g-u), v_-\rangle_{\partial K\cap\Gamma} = 0$$

where δ_h is $O(h)$. Using the *a priori* theory for the discontinuous Galerkin method together with standard approximation theory, terms in the error representation formula are readily estimated

$$|M(u_h) - M(u)|_{\mathrm{DG}} = \mathrm{I} + \mathrm{II} + \mathrm{III} + \mathrm{IV}$$

with

$$\begin{aligned}
\mathrm{I} &= |\sum_{K\in\mathcal{K}} (\mathcal{L}u_h - f, \Phi - \pi_h\Phi)_K| \le Ch^s |u|_{H^{s+1}(\Omega)} \times h^{s+1} |\Phi|_{H^{s+1}(\Omega)} \\
\mathrm{II} &= |\sum_{K\in\mathcal{K}} (\mathcal{L}u_h - f, \delta_h \mathcal{L}(\Phi - \pi_h\Phi))_K| \le Ch^s |u|_{H^{s+1}(\Omega)} \times h^{s+1} |\Phi|_{H^{s+1}(\Omega)} \\
\mathrm{III} &= |\sum_{K\in\mathcal{K}} \langle (\lambda\cdot n)^- [u]^+_-, \Phi - \pi_h\Phi\rangle_{\partial K\setminus\Gamma} \le Ch^{s+\frac{1}{2}} |u|_{H^{p+1}(\Omega)} \times h^{s+\frac{1}{2}} |\Phi|_{H^{s+1}(\Omega)} \\
\mathrm{IV} &= |\sum_{K\in\mathcal{K}} \langle (\lambda\cdot n)^- (g-u), \Phi - \pi_h\Phi\rangle_{\partial K\cap\Gamma} \le Ch^{s+\frac{1}{2}} |u|_{H^{s+1}(\Omega)} \times h^{s+1} |\Phi|_{H^{s+1}(\Gamma^-)}
\end{aligned} \tag{21}$$

for $0 \le s \le p$. In these estimates, each right-hand side term has been written as the product of two estimates coming from the primal and dual data respectively. If the infinite-dimensional primal and dual solution data are sufficiently smooth so that $|u|_{H^{p+1}(\Omega)}$ and $|\Phi|_{H^{p+1}(\Omega)}$ are bounded by a constant then

$$|M(u_h) - M(u)|_{\mathrm{DG}} \le Ch^{2p+1} \quad .$$

Examination of the right-hand side terms in (21) shows the important role of Galerkin orthogonality in attaining the "order doubling" superconvergence property of functionals in the discontinuous Galerkin method. Unfortunately, similar *a priori* results are not available for the Godunov finite volume method. If one only assumes orthogonality with respect to constants in the discontinuous Galerkin method then one would conclude from the above analysis that no superconvergence of functionals is attained. Fortunately, the computations given next indicate that this is not the case in the Godunov method and some limited superconvergence is observed but the prospect of order doubling is lost.

To numerically verify the convergence rate of functionals for smooth primal and dual data, numerical solutions of the following 2-D advection problem were obtained using the discontinuous Galerkin finite element method and the Godunov finite volume method with least-squares reconstruction operator as described in Barth and Larson [BL02]:

$$\begin{cases} \lambda\cdot\nabla u = 0 \ \text{ for } (x,y) \in [0,1]^2 \ , \\ u(1,y) = g_1(y) \ , \\ u(x,0) = g_2(x) \ , \end{cases} \tag{22}$$

with circular advection field $\lambda = (-y,x)^T$, $g_1(y) = 0$, and

Table 1. Convergence characteristics of the DG finite element and Godunov finite volume methods for the circular advection problem (22). Tabulated data for the global $L^2(\Omega)$ error and error in the weighted outflow flux functional.

Method	dofs	h	p	Global Error $\|\|u-u_h\|\|_{L^2}$ (rate)	Functional Error $\|M(u)-M(\tilde{u}_h)\|$ (rate)
DG FEM	1200	.0616	1	.639e-2	.232e-3
DG FEM	4800	.0258	1	.171e-2 (1.51)	.168e-4 (3.02)
DG FEM	19200	.0134	1	.390e-3 (2.26)	.217e-5 (3.12)
DG FEM	76800	.0071	1	.992e-4 (2.16)	.275e-6 (3.25)
Godunov FV	1600	.0258	1	.365e-2	.136e-3
Godunov FV	6400	.0134	1	.836e-3 (2.25)	.169e-4 (3.20)
Godunov FV	25600	.0071	1	.167e-3 (2.36)	.214e-5 (3.25)
Godunov FV	102400	.0036	1	.416e-4 (2.21)	.262e-6 (3.09)
DG FEM	2400	.0616	2	.936e-3	.128e-5
DG FEM	9600	.0258	2	.977e-4 (2.60)	.247e-7 (4.54)
DG FEM	38400	.0138	2	.108e-4 (3.51)	.110e-8 (4.97)
DG FEM	153600	.0071	2	.132e-5 (3.16)	.374e-10 (5.09)
Godunov FV	1600	.0258	2	.278e-2	.882e-4
Godunov FV	6400	.0134	2	.522e-3 (2.55)	.874e-5 (3.52)
Godunov FV	25600	.0071	2	.855e-3 (2.84)	.884e-6 (3.60)
Godunov FV	102400	.0036	2	.135e-4 (2.72)	.980e-7 (3.24)

$$g_2(x) = \begin{cases} \widetilde{\psi}(9/20;|x-1/2|)\cdot(1-\widetilde{\psi}(9/20;|x-1/20|)) & \text{if } x \le 1/2 \\ \widetilde{\psi}(9/20;|x-1/2|)\cdot(1-\widetilde{\psi}(9/20;|x-19/20|)) & \text{if } x > 1/2 \end{cases}$$

where $\widetilde{\psi}(\cdot;\cdot)$ is a C^∞ function

$$\widetilde{\psi}(r_0;r) = \begin{cases} 0 & r \ge r_0 \\ e^{1+r_0^2/(r^2-r_0^2)} & r < r_0 \end{cases} . \tag{23}$$

In addition, the weighted outflow flux functional

$$M_\psi(u) = \int_0^1 \psi_{\text{outflow}}(y)\,(\lambda\cdot n)^+ u(0,y)\,dy \tag{24}$$

was computed using the weighting function

$$\psi_{\text{outflow}}(y) = \begin{cases} \widetilde{\psi}(7/20;|y-3/5|)\cdot(1-\widetilde{\psi}(7/20;|y-1/4|)) & \text{if } y \le 3/5 \\ \widetilde{\psi}(7/20;|y-3/5|)\cdot(1-\widetilde{\psi}(7/20;|y-19/20|)) & \text{if } y > 3/5 \end{cases} . \tag{25}$$

This particular weighting function was chosen so that the corresponding dual solution would be smooth although the dual solution was not needed for the calculation of the functional error. Table 1 tabulates values of the global solution error and the error in the weighted outflow flux functional using a sequence of four meshes. The results using $p = 1$ approximation are very comparable between the finite element and finite volume methods. Each method shows second order convergence in the global L^2 error norm and third order convergence in the functional error. The numerical results using $p = 2$ approximation for the discontinuous Galerkin method confirm or exceed

the theoretically estimated rates for both the global L^2 error (third order) and the error in the weighted outflow functional (fifth order). Less favorably, the results for the Godunov finite volume method with $p = 2$ approximation show an improvement of no more than one power of h in the convergence rate of functional error when compared to the convergence rate of the global L^2 error. Although the numerical results for the Godunov finite volume method are far from conclusive and may depend on the details of the reconstruction operator, the results do indicate a marked difference between the finite element and finite volume method when computing functionals. It is conjectured that this difference can be explained to some extent by the lack of full Galerkin orthogonality in the finite volume method. Further analysis beyond the scope of this article is clearly needed to more fully explain the performance of the Godunov finite volume method.

5 Computable Error Estimates and Adaptivity

Computationally, the error representation formulas (12), (15) and (18) are not suitable for obtaining computable *a posteriori* error estimates and use in mesh adaptation.

- The functions $\Phi - \pi_h\Phi$ and $\Phi - \pi_0\Phi$ are unknown where $\Phi \in \mathcal{V}^B$ is a solution of the infinite-dimensional dual problem.
- The mean-value linearization used in the linearized dual problems (14) and (17) requires knowledge of the exact solution u.

Various strategies which address the numerical approximation of Φ are discussed in Barth and Larson [BL02], e.g. postprocessing, higher order solves, etc. Due to Galerkin orthogonality, the dual problem in the discontinuous Galerkin finite element method must be approximated in a larger space of functions than that utilized in the primal numerical problem. For purposes of the present study, this is achieved in the discontinuous Galerkin method by solving the dual problem using a polynomial space that is one polynomial degree higher than the primal numerical problem, viz. if $u_h \in \mathcal{V}^{\mathrm{B}}_{h,p}$ then $\Phi \approx \Phi_h \in \mathcal{V}^{\mathrm{B}}_{h,p+1}$. For the Godunov finite volume method, π_0 is the projection to piecewise constants. Consequently, the dual problem can be approximated using the same reconstruction operator as used in the primal problem ($p \neq 0$). In practice, there may be some additional improvement in accuracy by using an even higher order method for numerically approximating the dual problem. This is experimentally considered in Sect. 6.

In the present study, the mean-value linearization depending on the states u and u_h is replaced by the simpler jacobian linearization evaluated at the numerical state u_h. This is not the only practical choice. In Barth and Larson [BL02], a more sophisticated technique involving the postprocessing of primal data and the approximation of the mean-value linearization by numerical quadrature is employed in computations.

5.1 Direct Estimates

For brevity, the error representation formulas for the discontinuous Galerkin and Godunov finite volume methods can be combined into a single formula

$$\mathcal{M}(u) - \mathcal{M}(\tilde{u}_h) = F(\Phi - \pi\Phi) - \mathcal{B}_{\mathrm{DG}}(\tilde{u}_h, \Phi - \pi\Phi) \tag{26}$$

where $\tilde{u}_h \equiv u_h$, $\pi \equiv \pi_h$ for the discontinuous Galerkin method and $\tilde{u}_h \equiv R_p^0 u_0$, $\pi \equiv \pi_0$ for the Godunov finite volume method. When written in this global abstract form, the error representation formula does not indicate which elements in the mesh should be refined to reduce the measured error in a functional. By applying a sequence of direct estimates, error bounds suitable for adaptive meshing are easily obtained. The goal in constructing these estimates is to estimate the *local* contribution of each element in the mesh to the functional error. This local cell contribution will then be used as an error indicator for choosing which elements to refine or coarsen in the adaptive mesh procedure.

$$\begin{aligned}
\left|\mathcal{M}(u) - \mathcal{M}(\tilde{u}_h)\right| &= \left|\mathcal{B}_{\mathrm{DG}}(\tilde{u}_h, \Phi - \pi\Phi) - F(\Phi - \pi\Phi)\right| \quad \text{(error representation)} \\
&= \Big|\sum_{K\in\mathcal{K}} \left(\mathcal{B}_{\mathrm{DG},K}(\tilde{u}_h, \Phi - \pi\Phi) - F_K(\Phi - \pi\Phi)\right)\Big| \quad \text{(element assembly)} \\
&\le \sum_{K\in\mathcal{K}} \left|\left(\mathcal{B}_{\mathrm{DG},K}(\tilde{u}_h, \Phi - \pi\Phi) - F_K(\Phi - \pi\Phi)\right)\right| \quad \text{(triangle inequality)}
\end{aligned} \tag{27}$$

where $\mathcal{B}_{\mathrm{DG},K}(\cdot,\cdot)$ and $F_K(\cdot)$ are restrictions of $\mathcal{B}_{\mathrm{DG}}(\cdot,\cdot)$ and $F(\cdot)$ to the partition element K. The basic definition of the discontinuous Galerkin semilinear form given in (5) shows one possible element assembly form but this is not a unique representation. For example strong and weak forms of the semilinear operator $\mathcal{B}_{\mathrm{DG}}(\cdot,\cdot)$ yield differing assembly representations. For the discontinuous Galerkin and Godunov finite volume methods, the error representation formula (26) together with (5) for a single element K yields

$$\begin{aligned}
\mathcal{B}_{\mathrm{DG},K}(\tilde{u}_h, \Phi - \pi\Phi) - F_K(\Phi - \pi\Phi) \;=\; & -\int_K \sum_{i=1}^{d} f^i(\tilde{u}_h)\cdot(\Phi - \pi\Phi)_{,x_i}\, dx \\
& + \int_{\partial K\setminus\Gamma} (\Phi - \pi\Phi)_-\cdot h(n;(\tilde{u}_h)_-,(\tilde{u}_h)_+)\, ds \\
& + \int_{\partial K\cap\Gamma} (\Phi - \pi\Phi)_-\cdot h(n;(\tilde{u}_h)_-, g)\, ds\Big) \quad .
\end{aligned} \tag{28}$$

The present numerical computations utilize the numerical flux formula

$$h(n;u_-,u_+) = \frac{1}{2}\left(f(n;u_-) + f(n;u_+)\right) - \frac{1}{2}\left|A(n;\overline{u}(u_-,u_+))\right|[u]_-^+ \tag{29}$$

with $f(n;u) = \sum_{i=1}^{d} n_i f^i(u_-)$ and $A(n;u) = \partial f(n;u)/\partial u$. The state $\overline{u}(u_-,u_+)$ is chosen so that

$$[f(n;u)]^{+}_{-} = A(n;\overline{u}(u_-,u_+))\,[u]^{+}_{-} \quad . \tag{30}$$

Using this particular numerical flux, the following weighted residual (strong) form can be obtained upon integration by parts

$$\begin{aligned}\mathcal{B}_{\mathrm{DG},K}(\tilde{u}_h,\Phi-\pi\Phi) \;-\; F_K(\Phi-\pi\Phi) = &\int_K (\Phi-\pi\Phi)\cdot\sum_{i=1}^{d} f^i_{,x_i}(\tilde{u}_h)\,dx \\ &+ \int_{\partial K\setminus\Gamma} (\Phi-\pi\Phi)_-\cdot A^-(n;(\tilde{u}_h)_-,(\tilde{u}_h)_+)\,[\tilde{u}_h]^{+}_{-}\,ds \\ &+ \int_{\partial K\cap\Gamma} (\Phi-\pi\Phi)_-\cdot A^-(n;(\tilde{u}_h)_-,g)\,(g-(\tilde{u}_h)_-)\,ds\Big) \quad .\end{aligned} \tag{31}$$

This latter weighted residual form and the implied element assembly form $\sum_K \mathcal{B}_K(\cdot,\cdot) - F_K(\cdot)$ is preferred in the error estimates (27) since the individual terms represent residual components that vanish individually when the exact solution is inserted into the variational form and a slightly sharper approximation is obtained after application of the triangle inequality in (27).

5.2 Adaptive Meshing

Motivated by the direct estimates (27), we define for each partition element K the *adaptation element indicator* $|\eta_K|$

$$\eta_K \equiv \mathcal{B}_{\mathrm{DG,K}}(\tilde{u}_h,\Phi-\pi\Phi) - F_K(\Phi-\pi\Phi) \quad , \tag{32}$$

such that

$$\left|\mathcal{M}(u)-\mathcal{M}(\tilde{u}_h)\right| \le \sum_{K\in\mathcal{K}} |\eta_K| \tag{33}$$

and an accurate *adaptation stopping criteria*

$$\left|\mathcal{M}(u)-\mathcal{M}(\tilde{u}_h)\right| = |\sum_{K\in\mathcal{K}} \eta_K| \quad . \tag{34}$$

These quantities suggest a simple mesh adaptation strategy in common use with other indicator functions:

Mesh Adaptation Algorithm
(1) Construct an initial mesh $\mathcal{K}$.
(2) Compute a numerical approximation of the primal problem on the current mesh $\mathcal{K}$.
(3) Compute a suitable numerical approximation of the infinite-dimensional dual problem on the current mesh $\mathcal{K}$.
(4) Compute error indicators, η_K, for all elements $K \in \mathcal{K}$.
(5) If($|\sum_{K\in\mathcal{K}} \eta_K| < TOL$) STOP
(6) Otherwise, refine and coarsen a specified fraction of the total number of elements according to the size of $|\eta|_K$, generate a new mesh $\mathcal{K}$, and GOTO 2

6 Numerical Results

In this section, selected numerical examples are given for scalar advection and systems of nonlinear conservation laws. Further numerical examples can be found in [LB99] and [BL02].

6.1 Linear Advection

To assess the sharpness of the computable error estimates, the circular advection problem given in (22) is again considered. Table 2 tabulates values of the functional error and the estimated error as given in (27) using numerically approximated dual problems (see Sect. 5). In addition, the effectivity index is included to characterize sharpness of the estimates

$$\theta_{\text{eff}} \equiv \frac{|\text{estimated error}|}{|M(u) - M(\tilde{u}_h)|} \quad .$$

Recall that when the exact dual solution Φ is used

$$|\mathcal{M}(u) - \mathcal{M}(\tilde{u}_h)| = |\sum_{K \in \mathcal{K}} \eta_K| \quad .$$

Consequently, the seventh column measures the effect of numerically approximating the dual problem. For this particular test problem, the technique of approximating the dual problem using a higher order method for the discontinuous Galerkin method yields extremely accurate estimates of the functional error with θ_{eff} very close to unity. This comes at a fairly high price given the dramatic increase in arithmetic complexity of the discontinuous Galerkin method with increasing p. The results for the Godunov finite volume method show that reasonable estimates can be obtained by computing the dual problem with the same order method as the primal problem. For $p = 1$, some improvement in the Godunov finite volume method is achieved using a higher order method for the dual problem. For $p = 2$, the need for solving the dual problem using a higher order method seems entirely unnecessary since effectivity indices near unity are achieved.

After application of the triangle inequality, the estimate

$$|\mathcal{M}(u) - \mathcal{M}(\tilde{u}_h)| \leq \sum_{K \in \mathcal{K}} |\eta_K| \tag{35}$$

is obtained for use in mesh adaptivity. Column eight in Table 2 shows some loss in sharpness in this error estimate since the possibility of interelement error cancellation is precluded. Even so, the working assumption is that this estimate is sufficiently accurate to drive efficient mesh adaptivity. The results for the Godunov method with $p = 1$ for the primal numerical problem show no significant differences in the (35) estimate using either $p = 1$ or $p = 2$ solves for the dual problem. These numerical results again illustrate significant differences between the Godunov finite volume method and the discontinuous Galerkin methods that are worth further investigation.

Table 2. Efficiency of the DG finite element and Godunov finite volume methods error estimates for the circular advection problem (22). Tabulated data for the weighted outflow flux functional error and the estimates given in (27) .

Method	dofs	h	primal p	dual p	$\lvert M(u)-M(\tilde{u}_h)\rvert$	$\lvert\sum_K \eta_K\rvert$ (θ_{eff})	$\sum_K \lvert\eta_K\rvert$ (θ_{eff})
DG FEM	1200	.0616	1	2	.232e-3	.233e-3 (1.00)	.377e-3 (1.62)
DG FEM	4800	.0258	1	2	.168e-4	.168e-4 (1.00)	.380e-4 (2.26)
DG FEM	19200	.0134	1	2	.217e-5	.217e-5 (1.00)	.498e-5 (2.29)
DG FEM	76800	.0071	1	2	.275e-6	.276e-6 (1.00)	.582e-6 (2.11)
Godunov FV	1600	.0258	1	1	.136e-3	.100e-3 (.735)	.727e-3 (5.35)
Godunov FV	6400	.0134	1	1	.169e-4	.152e-4 (.900)	.167e-3 (9.88)
Godunov FV	25600	.0071	1	1	.214e-5	.188e-5 (.879)	.378e-4 (17.7
Godunov FV	102400	.0036	1	1	.262e-6	.245e-6 (.935)	.900e-5 (34.4)
Godunov FV	1600	.0258	1	2	.136e-3	.141e-3 (1.04)	.756e-3 (5.56)
Godunov FV	6400	.0134	1	2	.169e-4	.174e-4 (1.03)	.167e-3 (9.88)
Godunov FV	25600	.0071	1	2	.214e-5	.216e-5 (1.01)	.378e-4 (17.7)
Godunov FV	102400	.0036	1	2	.262e-6	.263e-6 (1.00)	.892e-5 (34.0)
DG FEM	2400	.0616	2	3	.128e-5	.959e-6 (.750)	.990e-5 (7.73)
DG FEM	9600	.0258	2	3	.247e-7	.237e-7 (.960)	.158e-6 (6.40)
DG FEM	38400	.0138	2	3	.110e-8	.109e-8 (.991)	.465e-8 (4.22)
DG FEM	153600	.0071	2	3	.374e-10	.373e-10 (.997)	.143e-9 (3.83)
Godunov FV	1600	.0258	2	2	.882e-4	.893e-4 (1.01)	.294e-3 (3.33)
Godunov FV	6400	.0134	2	2	.874e-5	.875e-5 (1.00)	.355e-4 (4.06)
Godunov FV	25600	.0071	2	2	.884e-6	.885e-6 (1.00)	.400e-5 (4.52)
Godunov FV	102400	.0036	2	2	.980e-7	.980e-7 (1.00)	.494e-6 (5.04)

6.2 Compressible Euler Flow

In this example, Ringleb flow (an exact solution of the 2-D Euler equations obtained via hodograph transformation, see [Chi85]) is computed in the channel geometry using the discontinuous Galerkin method with linear elements. To illustrate the use of the element indicators (32) in adaptive meshing, the mollified pointwise functional

$$M_\delta(u) = \int_\Omega \text{Energy}(u)\ \tilde{\psi}(r_0;|x-x_0|)\,dx\ ,\quad x_0 = (-.63,1.70)^T\ ,\quad r_0 = 1/10$$

has been implemented for the energy component of the solution. Using this functional, the corresponding dual problem has been computed and the mesh adapted using the adaptation algorithm of Sect. 5.2. Figure 2 shows the resulting dual solution and the adapted mesh with three levels of refinement. The adapted mesh shows the upstream dependence of numerical residual errors on the accuracy of this local functional. Further details of the mesh adaptation process are given in Table 3. This table also gives the approximate number of cells needed in a uniformly refined mesh to achieve the same level of accuracy in the target functional. With just three levels of refinement, the number of mesh cells in the adapted mesh is reduced by over a factor

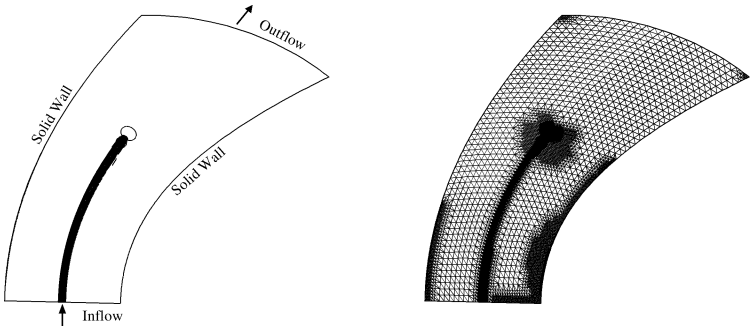


Fig. 2. Ringleb channel geometry. Dual energy isocontours solution corresponding to mollified delta functional (left) and final adapted mesh (3 levels) (right).

Table 3. Performance of adaptive meshing algorithm for the Ringleb flow problem and the mollified pointwise energy functional.

Level	$\lvert M(u)-M(\tilde{u}_h)\rvert$	(Adaptive) # Mesh Cells	(Uniform) # Mesh Cells
0	2.40e-7	1482	1482
1	6.18e-8	2020	3422
2	8.82e-9	4010	14042
3	1.25e-9	10214	56882

of five from uniform refinement. This indicates the significant savings achieved by the adaptive algorithm.

7 Concluding Remarks

A simple *a posteriori* error estimation theory for user specified functionals has been constructed that is tailored to Godunov finite volume and discontinuous Galerkin methods. Many issues remain unresolved:

- Mean-value linearization for schemes with non-differentiable limiters and/or reconstruction algorithms.
- Existence and solvability of dual problems.
- Numerical approximation of dual problems.

Even though error representations formula have been developed for both linear and nonlinear functionals, very little is theoretically known about the existence or solvability of dual solutions for general functionals. The analysis is made particularly difficult since the linearized infinite-dimensional dual problem can have discontinuous coefficients. In practice, one often finds that dual solutions are more complicated in structure than the sought after primal solution. As an example, consider transonic Euler flow ($M\infty = .85, \alpha = 2.0^\circ$) over the NACA 0012 airfoil geometry. Suppose the aerodynamic lift functional is chosen for evaluation:

$$M_{\text{Lift}}(u) = \int_{\Gamma_{wall}} (n \cdot V^{\perp}/|V|)\,\text{Pressure}(u)\; dx \;\; .$$

Figure 3 shows isodensity contours of the primal solution and isodensity contours of the dual solution corresponding to the lift functional. The dual solution has a com-

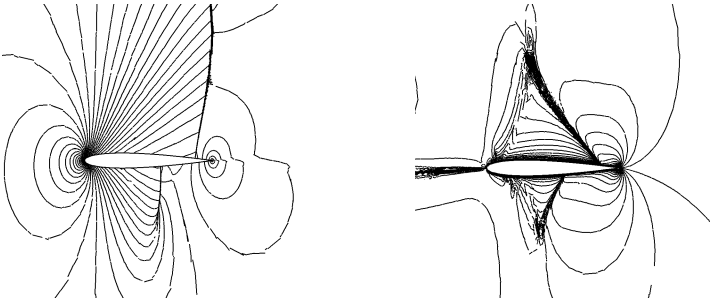


Fig. 3. NACA airfoil geometry, $M_\infty = .85, \alpha = 2.0°$. Isodensity contours of primal solution (left) and corresponding contours of the dual density solution (right) for the lift functional.

plicated structure with a large dipole singularity at the trailing edge of the airfoil and numerous layers emanating from the airfoil surface near the leading edge stagnation point, upper and lower sonic points, and the base of the upper and lower shockwaves. These structures signify the sensitivity of the lift force to these features. These structures place additional demands on the discretization and suggests that the extension to 3-D is a truly challenging problem. This will be pursued in future work.

References

Abg94. R. Abgrall. An essentially non-oscillatory reconstruction procedure on finite-element type meshes. *Comp. Meth. Appl. Mech. Engrg.*, 116:95–101, 1994.

Bar98. T.J. Barth. Numerical methods for gasdynamic systems on unstructured meshes. In Kröner, Ohlberger, and Rohde, editors, *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, volume 5 of *Lecture Notes in Computational Science and Engineering*, pages 195–285. Springer-Verlag, Heidelberg, 1998.

BD02. T.J. Barth and H. Deconinck(eds). Error estimation and adaptive discretization methods in CFD. In Barth and Deconinck, editors, *Error Estimation and Adaptive Discretization Methods in CFD*, volume 25 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Heidelberg, 2002.

BF90. T. J. Barth and P.O. Frederickson. Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction. Technical Report 90-0013, AIAA, Reno, NV, 1990.

BJ89. T. J. Barth and D. C. Jespersen. The design and application of upwind schemes on unstructured meshes. Technical Report 89-0366, AIAA, Reno, NV, 1989.

BL02. T.J. Barth and M.G. Larson. A-posteriori error estimation for higher order Godunov finite volume methods on unstructured meshes. In Herbin and Kröner, editors, *Finite Volumes for Complex Applications III*, pages 41–63. Hermes Science Pub., London, 2002.

BR98. R. Becker and R. Rannacher. Weighted a posteriori error control in FE methods. In *Proc. ENUMATH-97, Heidelberg*. World Scientific Pub., Singapore, 1998.

Chi85. G. Chiocchia. Exact solutions to transonic and supersonic flows. Technical Report AR-211, AGARD, 1985.

CLS89. B. Cockburn, S.Y. Lin, and C.W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems. *J. Comp. Phys.*, 84:90–113, 1989.

CS97. B. Cockburn and C.W. Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws V: Multidimensional systems. Technical Report 201737, Institite for Computer Applications in Science and Engineering (ICASE), NASA Langley R.C., 1997.

DOE90. L. Durlofsky, S. Osher, and B. Engquist. Triangle based TVD schemes for hyperbolic conservation laws. Technical Report 90-10, Institite for Computer Applications in Science and Engineering (ICASE), NASA Langley R.C., 1990.

EEHJ95. K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to numerical methods for differential equations. *Acta Numerica*, pages 105–158, 1995.

GLLS97. M. Giles, M. Larson, M. Levenstam, and E. Süli. Adaptive error control for finite element approximations of the lift and drag coefficients in viscous flow. preprint NA-97/06, Comlab, Oxford University, 1997.

God59. S. K. Godunov. A finite difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sb.*, 47:271–290, 1959.

GP99. M. Giles and N.A. Pierce. Improved lift and drag estimates using adjoint Euler equations. Technical Report 99-3293, AIAA, Reno, NV, 1999.

Har83. A. Harten. High resolution schemes for hyperbolic conservation laws. *J. Comp. Phys.*, 49:357–393, 1983.

Har89. A. Harten. ENO schemes with subcell resolution. *J. Comp. Phys.*, 83:148–184, 1989.

HH02. R. Hartmann and P. Houston. Adaptive discontinuous galerkin methods for the compressible euler equations. *J. Comp. Phys.*, 182(2):508–532, 2002.

HOEC87. A. Harten, S. Osher, B. Engquist, and S. Chakravarthy. Uniformly high-order accurate essentially nonoscillatory schemes III. *J. Comp. Phys.*, 71(2):231–303, 1987.

Joh87. C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, Cambridge, 1987.

JP86. C. Johnson and J. Pitkäranta. An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation. *Math. Comp.*, 46:1–26, 1986.

JRB95. C. Johnson, R. Rannacher, and M. Boman. Numerics and hydrodynamics stability theory: towards error control in CFD. *SIAM J. Numer. Anal.*, 32:1058–1079, 1995.

LB99. M.G. Larson and T.J. Barth. A posteriori error estimation for adaptive discontinuous Galerkin approximations of hyperbolic systems. In Cockburn, Karniadakis, and Shu, editors, *Discontinuous Galerkin Methods*, volume 11 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Heidelberg, 1999.

OP99. J. T. Oden and S. Prudhomme. Goal-oriented error estimation and adaptivity for the finite element method. Technical Report 99-015, TICAM, U. Texas, Austin,TX, 1999.

PO99. S. Prudhomme and J.T. Oden. On goal-oriented error estimation for elliptic problems: application to the control of pointwise errors. *Comp. Meth. Appl. Mech. and Eng.*, pages 313–331, 1999.

RH73. W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos National Laboratory, Los Alamos, New Mexico, 1973.

S̈98. E. Süli. A posteriori error analysis and adaptivity for finite element approximations of hyperbolic problems. In Kröner, Ohlberger, and Rohde, editors, *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, volume 5 of *Lecture Notes in Computational Science and Engineering*, pages 122–194. Springer-Verlag, Heidelberg, 1998.

Van93. P. Vankeirsblick. *Algorithmic Developments for the Solution of Hyperbolic Conservation Laws on Adaptive Unstructured Grids*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 1993.

vL79. B. van Leer. Towards the ultimate conservative difference schemes V. A second order sequel to Godunov's method. *J. Comp. Phys.*, 32:101–136, 1979.

AMR for low Mach number reacting flow

John Bell

Lawrence Berkeley National Laboratory
Berkeley, CA 94720
jbbell@lbl.gov

Summary. We present a summary of recent progress on the development and application of adaptive mesh refinement algorithms for low Mach number reacting flows. Our approach uses a form of the low Mach number equations based on a general equation of state that discretely conserves both mass and energy. The discretization methodology is based on a robust projection formulation that accommodates large density contrasts. The algorithm supports modeling of multicomponent systems and incorporates an operator-split treatment of stiff reaction terms. The basic computational approach is embedded in an adaptive projection framework that uses structured hierarchical grids with subcycling in time that preserves the discrete conservation properties of the underlying single-grid algorithm. We present numerical examples illustrating the application of the methodology to turbulent premixed combustion and nuclear flames in type Ia supernovae.

1 Introduction

Detailed modeling of time-dependent reacting flows with realistic reaction mechanisms places severe demands on computational resources. These computational costs can be dramatically reduced by combining a low Mach number formulation that allows a large increase in time step size with local adaptive mesh refinement to reduce the total number of computational zones that must be advanced for a specific problem. Low Mach number models analytically eliminate acoustic waves from the system while preserving compressibility effects arising from the reaction process and associated thermal behavior of the fluid. These types of models for reacting flows were first introduced by Rehm and Baum [29]. A systematic approach for deriving these types of models using aympotitics in Mach number was given by Majda and Sethian [23]. Low Mach number models replace the compressible Navier Stokes equations with a system evolving subject to a constraint on the velocity field. Since acoustic waves have been analytically removed from the system, the time step is determined by the advective time scale of the flow.

Local refinement for steady combustion has been discussed by a number of authors. See for example, Smooke *et al.* [34], Coelho and Pereira [13], de Lange and de

Goey [16], Mallens *et al.* [24] Somers and de Goey [31], Bennett and Smooke [10], Bennett *et al.* [9], Becker *et al.* [3] and the references cited in these works.

For time-dependent flows, Najm *et al.* [26] couple a local refinement algorithm for species and temperature with a vortex method for momentum. Pember *et al.* [28] present an adaptive projection algorithm for time-dependent low Mach number combustion using simplified kinetics and an assumption of unity Lewis number. The methodology in Pember *et al.* [28] uses a hierarchical structured refinement approach based on the local adaptive projection methodology developed by Almgren *et al.* [2]. The method discussed here represents a generalization of the Pember *et al.* methodology. In particular, it incorporates complex chemistry and the effects of differential diffusion as discussed in Day and Bell [15] and the extension of the methodology to nonideal equations of state as discussed in Bell *et al.* [5]. The reader is referred to those papers for additional detail about the methodology.

We note that our basic discretization approach differs from the standard approach to solving the low Mach number system originally proposed by McMurtry *et al.* [25]. In the McMurtry *et al.* approach an auxiliary equation for the density in convective form is derived by differentiating the equation of state in time and replacing temporal derivatives of temperature and species by spatial derivatives of these quantities. This equation is then used to advance the density in time with temperature being determined from the equation of state. In the projection step, the McMurtry *et al.* algorithm solves a constant coefficient Poisson equation to modify the velocity field so that the conservation of mass equation is satisfied.

In contrast to this approach, we directly solve the conservation form of the equations for both enthalpy and density. Our projection step solves a variable coefficient elliptic equation to enforce the velocity constraint given in equations orginally introduced by Bell and Marcus [8]. This approach was first extended to combustion by Lai [21] and Lai *et al.* [22]. Related implementations or extensions include Hilditch and Colella [19] and Pember *et al.* [27]. Unlike the standard approach, the approach discussed here conserves both mass and energy; however, the evolution does not remain on the constraint imposed by the equation of state. Instead, the evolution is allowed to drift within a small neighborhood of that constraint.

In section 2, we introduce the low Mach number equations for a general equation of state. In section 3, we describe a second-order projection algorithm for integrating the low Mach number equations and give an overview of our adaptive methodology. In the final two sections we present prototype applications of this methodology to premixed turbulent combustion and nuclear flames. The results shown here are taken from studies presented in [7] and [6], respectively.

2 Low Mach number model

The low Mach number model is derived from the compressible flow equations using asymptotic analysis. These equations describe conservation of mass, momentum and energy augmented with equations for the species representing the composition of the fluid.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho U = 0$$
$$\frac{\partial \rho U}{\partial t} + \nabla \cdot (\rho UU + p) = \nabla \cdot \tau + \rho \vec{g}$$
$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho UE + pU) = -\nabla \cdot q + \nabla \cdot \tau U + \rho U \cdot \vec{g}$$
$$\frac{\partial \rho Y_k}{\partial t} + \nabla \cdot \rho U Y_k = \nabla \cdot \rho D_k \nabla Y_k + \rho \dot{\omega}_k$$

Here, ρ, U, T and p are the density, velocity, temperature, and pressure, respectively, and $E = e + U \cdot U/2$ is the total energy with e representing the internal energy. Note that e includes the energy of formation and mixing so that there is no explicit term in the energy equation due to reaction; those effects are implicitly included in the definition of e. (See, for example, Bird, Steward and Lightfoot [11].) In addition, Y_k is the mass fraction of species k, with associated production rate $\dot{\omega}_k$. Here, both the $\dot{\omega}_k$ and the $\rho D_k \nabla Y_k$ must sum to zero, expressing the notion that reactions conserve mass and species diffusion cannot transport net mass. For simplicity we assume a mixture model for species diffusion and ignore thermal diffusion (Soret) and Dufour effects as well as radiative heat transfer. (Generalizing the formulation to include these effects is straightforward.) For this case, the heat flux, q, is given by [1]

$$q = -\kappa \nabla T - \rho \xi_k D_k \nabla Y_k$$

with $\xi_k = \frac{\partial h}{\partial Y_k}\Big|_{p,T,Y_{j,j\neq k}}$, where the enthalpy, $h = e + p/\rho$. Finally, τ is the stress tensor, $\vec{g}$ is the gravitational force and κ is the thermal conductivity.

As a prelude to developing the low Mach number equations, we first rewrite the energy equation in terms of the enthalpy,

$$\rho \frac{Dh}{Dt} - \frac{Dp}{Dt} = \nabla \cdot \kappa \nabla T + \nabla \cdot \rho \xi_k D_k \nabla Y_k$$

For the low Mach number asymptotic analysis (following Majda and Sethian [23]), we introduce scaled coordinates in which the time scale is proportional to the spatial scale divided by the advective velocity scale. In this scaling, we expand pressure and velocity in Mach number, $M = U/c_s$, (c_s is the sound speed). Substituting these expansions in M into the equations of motion given above, retaining highest order terms in M we find that the $O(1)$ pressure term is independent of the spatial coordinate and the $O(M)$ pressure term is zero. Thus, in the low Mach number expansion, presure is of the form

$$p(x,t) = p_0(t) + M^2 \pi(x,t)$$

Thus, the pressure is decomposed into a thermodynamic component, p_0, that depends only on time and a perturbation component, π, that is $O(M^2)$. For the low Mach

[1] Unless otherwise noted, we use the summation convention throughout the paper.

number model, we ignore the $O(M^2)$ effects on the thermodynamics. For simplicity, in this paper we will assume that the flow occurs in an open environment under constant pressure so that the thermodynamic pressure is, in fact, a constant which we denote as p_0. This leads to a modified momentum equation

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot \rho UU = -\nabla \pi + \nabla \tau + \rho \vec{g}. \tag{1}$$

and reduces the enthalpy equation to

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho U h) = \nabla \cdot \kappa \nabla T + \nabla \cdot \rho \xi_k D_k \nabla Y_k \tag{2}$$

The enthalpy and momentum equations combined with the species equations (and conservation of mass) describe the evolution of the low Mach number system. However, this evolution is also constrained by the equation of state. This constraint is equivalent to a constraint on the divergence of the velocity field that is obtained by differentiating the equation of state along particle paths

$$0 \equiv \frac{Dp}{Dt} = \frac{\partial p}{\partial \rho}\frac{D\rho}{Dt} + \frac{\partial p}{\partial T}\frac{DT}{Dt} + \frac{\partial p}{\partial Y_k}\frac{DY_k}{Dt} \quad .$$

Combining this equation with the mass conservation equation, we obtain

$$\nabla \cdot U = \frac{1}{\rho \frac{\partial p}{\partial \rho}} \left(\frac{\partial p}{\partial T}\frac{DT}{Dt} + \sum_k \frac{\partial p}{\partial Y_k}\frac{DY_k}{Dt} \right)$$

To complete the specification of the low Mach number model, we need to derive the temperature evolution equation. For this derivation, we express the enthalpy as a function of p, T, and Y_k. and differentiate the enthalpy equation to obtain

$$\frac{Dh}{Dt} = \left.\frac{\partial h}{\partial T}\right|_{p,Y_k} \frac{DT}{Dt} + \left.\frac{\partial h}{\partial p}\right|_{T,Y_k} \frac{Dp}{Dt} + \left.\frac{\partial h}{\partial Y_k}\right|_{p,T,Y_{j,j\neq k}} \frac{DY_k}{Dt}$$

After substituting from the above equations and using the low Mach number condition on p we have

$$\rho c_p \frac{DT}{Dt} = \nabla \cdot \kappa \nabla T + \rho D_k \nabla \xi_k \cdot \nabla Y_k - \rho \xi_k \dot{\omega}_k \tag{3}$$

where $c_p = \left.\frac{\partial h}{\partial T}\right|_{p,Y_k}$ is the specific heat at constant pressure.

Substituting into the above equation for $\nabla \cdot U$ yields an expression for a constraint on the advective flow velocities:

$$\begin{aligned} \nabla \cdot U = & \frac{1}{\rho \frac{\partial p}{\partial \rho}} \left(\frac{1}{\rho c_p}\frac{\partial p}{\partial T} \left(\nabla \cdot \kappa \nabla T + \rho D_k \nabla \xi_k \cdot \nabla Y_k - \rho \xi_k \dot{\omega}_k \right) \right. \\ & \left. + \frac{1}{\rho}\frac{\partial p}{\partial Y_k} \left(\nabla \rho D_k \nabla Y_k + \rho \dot{\omega}_k \right) \right) \\ & \equiv S. \end{aligned} \tag{4}$$

3 Numerical methodology

In this section we discuss the numerical methodology used to integrate the low Mach number equations. The spatial discretization uses finite volume differencing with ρ, h, U, $\nabla\pi$ and the Y_k's defined on cell centers. The perturbational pressure is staggered in both space and time . Advection is discretized using a second-order Godunov-type procedure while diffusion is approximated with standard finite different methods. Our temporal discretization strategy is a fractional step approach based on a projection approximation. In this approach we integrate the equations for momentum, species and enthalpy using a lagged approximation to the constraint. We then apply a discrete projection to the intermediate velocity computed in the first step to enforce the constraint. This basic fractional step algorithm is embedded in a hierarchical adaptive mesh refinement (AMR) algorithm. The methodology presented here is documented in more detail in Day and Bell [15] for gaseous combustion and the extension to general equation of state is discussed in Bell *et al.* [5]. In the next subsection we describe the single-grid algorithm. We then discuss incorporation of that algorithm into an adaptive projection framework.

3.1 Single grid algorithm

The single grid algorithm is essentially a three-step process. First, we use an unsplit second-order Godunov procedure to predict a time-centered ($t^{n+1/2}$) advection velocity, $U^{\mathrm{ADV},*}$, using the cell-centered data at t^n and the lagged pressure gradient at $t^{n-1/2}$. The provisional field, $U^{\mathrm{ADV},*}$, represents a normal velocity on cell edges analogous to a MAC-type staggered grid discretization of the Navier-Stokes equations (see [18], for example). However, $U^{\mathrm{ADV},*}$ fails to satisfy the time-centered divergence constraint. We apply a discrete projection by solving the elliptic equation

$$D^{\mathrm{MAC}}\frac{1}{\rho^n}G^{\mathrm{MAC}}\phi^{\mathrm{MAC}} = D^{\mathrm{MAC}}U^{\mathrm{ADV},*} - \left(S^n + \frac{\Delta t^n}{2}\,\frac{S^n - S^{n-1}}{\Delta t^{n-1}}\right) \tag{5}$$

for ϕ^{MAC}, where D^{MAC} represents a centered approximation to a cell-based divergence from edge-based velocities, and G^{MAC} represents a centered approximation to edge-based gradients from cell-centered data. The solution, ϕ^{MAC}, is then used to define

$$U^{\mathrm{ADV}} = U^{\mathrm{ADV},*} - \frac{1}{\rho^n}G^{\mathrm{MAC}}\phi^{\mathrm{MAC}}.$$

U^{ADV} is a second-order accurate, staggered-grid vector field at $t^{n+1/2}$ that discretely satisfies the constraint (4), and is used for computing the advective derivatives for U, ρh and ρY_k.

In the next step of the algorithm we advance the advection-reaction-diffusion system for ρh and ρY_k. For the types of problems considered here, the reactions typically occur on a scale faster than the fluid dynamics. For that reason, we treat the reactions using a symmetric Strang-splitting approach so that the reactions can be treated with stiff ODE methodology. We first advance the reactions terms $\Delta t/2$

in time. We then advance the advection-diffusion part of the equation Δt in time followed by a second advancement of the reaction terms $\Delta t/2$ in time.

The reaction part of the enthalpy and species equations are of the form

$$\frac{\partial Y_k}{\partial t} = \dot{\omega}_k$$

and

$$c_p \frac{\partial T}{\partial t} = -\sum_k \xi_k \dot{\omega}_k$$

Here, we include the evolution of temperature as part of the system of ODE's integrated in the reaction step. We note, however, that for the reaction phase of the computation the enthalpy remains constant. Thus, the evolution of temperature can also be computed from the evolution of species by solving for the temperature given the updated composition and the (constant) enthalpy. We have not used this later approach because of the added computational expense; however, to preserve the conservation of energy we do not use the updated temperature from the reaction step to update the enthalpy. Instead, after the ODE integration, we recompute temperature from the enthalpy and final species concentrations.

In our implementation, we integrate the chemistry component using time-implicit backward difference methods, as implemented in `VODE` [12], a general-purpose stiff ODE integration software package. `VODE` utilizes adaptivity in order of accuracy and subcycled time-step selection so that an absolute error tolerance of 10^{-16} in mass fractions is maintained throughout. Typically, the resulting scheme is between third and fifth order convergent in time.

After completing the first reaction step, we update the advection-diffusion component of the system. One numerical issue that must be addressed at this point is the nonlinearity of the enthalpy diffusion. To facilitate the solution of the enthalpy equation we rewrite the heat flux in terms of enthalpy diffusion

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot U\rho h = \nabla \cdot \frac{\kappa}{c_p} \nabla h - \nabla \cdot \left(\sum_k \xi_k \frac{\kappa}{c_p} \nabla Y_k \right) + \nabla \cdot \rho D_k \xi_k \nabla Y_k \qquad (6)$$

We advance the species equations and this modfied form of the enthalpy equations using a Crank-Nicolson treatment of diffusion and an explicit treatment of advection. For the diffusive terms, the coefficients D_k, ξ_k, κ and c_p are nonlinear functions of ρ, h and the Y_k's. We treat this nonlinearity using lagged coefficients in a simple iterative scheme. Two steps of this iteration are sufficient to guarantee second-order convergence of the scheme. Another issue in the treatment of these equations is that mixture models for species diffusion do not necessarily determine diffusion fluxes that sum to zero. A standard method for correcting this discrepancy is to define a "conservation diffusion velocity" as recommended by Coffee and Heimerl [14], and rigorously justified as a first term in a convergent series expression for full matrix diffusion by Giovangigli [17]. In the algorithm, we compute species diffusion implicitly then explictly apply this correction. The corrected diffusion fluxes will be denoted with an overbar.

We begin with data obtained by advancing the first half of the chemistry integration which we denote with a superscript n. We compute edge-centered states for ρY_k and T at $t^{n+1/2}$ using a second-order Godunov procedure. Time-centered edge values of ρY_k and T are used to compute $t^{n+1/2}$ edge values for $\rho = \sum \rho Y_k$ and ρh. We now update ρ using the discrete form of the continuity equation

$$\rho^{n+1} = \rho^n - \Delta t \, \nabla \cdot \left(\sum_k U^{ADV} \rho Y_k^{n+1/2} \right).$$

Given the updated density, we define $Y_k^{n+1,0} = Y_k^n$, $h^{n+1,0} = h^n$ and $T^{n+1,0} = T^n$ to initialize the nonlinear iteration of the diffusion coefficients.

With the current iterate we now compute the approximations to the terms required to form the diffusion coefficients, D_k, ξ_k, κ and c_p which we denote with the superscript $n+1,m$. We then solve for a provisional update to the mass fractions

$$\frac{\rho^{n+1}\tilde{Y}_k^{n+1,m} - \rho Y_k}{\Delta t} + \left(\nabla \cdot U^{ADV} \rho Y_k\right)^{n+1/2} = \frac{1}{2} \nabla \cdot \left(\rho^{n+1} D_k^{n+1,m} \nabla \tilde{Y}_k^{n+1,m} + \overline{\rho^n D_k^n \nabla Y_k^n} \right). \tag{7}$$

After the solution of (7) the initial new-time fluxes, $-\rho^{n+1} D_k^{n+1,m} \nabla \tilde{Y}_k^{n+1,m}$ do not sum to zero. We again modify the fluxes using the correction velocity approach and use the result to correct the provisional update to the mass fractions

$$\frac{\rho^{n+1} Y_k^{n+1,m} - \rho^n Y_k^n}{\Delta t} + \left(\nabla \cdot U^{ADV} \rho Y_m\right)^{n+1/2} = \frac{1}{2} \left(\overline{\rho^{n+1} D_k^{n+1,m} \nabla \tilde{Y}_k^{n+1,m}} + \overline{\rho^n D_k^n \nabla Y_k^n} \right).$$

We now update the enthalpy equation using

$$\frac{\rho^{n+1} h^{n+1,m} - \rho^n h^n}{\Delta t} + \left(\nabla \cdot U^{ADV} \rho h\right)^{n+1/2} = \frac{1}{2} \left[\nabla \cdot \frac{\kappa^{n+1,m}}{c_p^{n+1,m}} \nabla h^{n+1,m} + \nabla \cdot \frac{\kappa^n}{c_p^n} \nabla h^n \right] + \frac{1}{2} \nabla \cdot \left[\xi_k^{n+1,m} \left(\overline{\rho^{n+1} D_k^{n+1,m} \nabla \tilde{Y}_k^{n+1,m}} - \frac{\kappa^{n+1,m}}{c_p^{n+1,m}} \nabla Y_k^{n+1,m} \right) + \xi_k^n \left(\overline{\rho^n D_k^n \nabla Y_k^n} - \frac{\kappa^n}{c_p^n} \nabla Y_k^n \right) \right]. \tag{8}$$

Finally, we solve for an updated temperature, $T^{n+1,m}$, consistent with the new approximation to species and enthalpy. We note that two passes through this iteration

is sufficient to guarantee second-order accuracy of the advection-diffusion component of the algorithm.

The integration of the enthalpy and species equations is completed by advancing the reaction part of the system an additonal $\Delta t/2$ in time. This provides a complete update of the ρ, h, T, and Y_k's at the new time and allows us to evaluate the constraint on the velocity field, S^{n+1}, at the new time.

The final step of basic integration step is to advance the velocity to the new time level. For this step we first obtain a provisional cell-centered velocity at t^{n+1} using a time-lagged pressure gradient,

$$\rho^{n+1/2}\frac{U^{n+1,*}-U^n}{\Delta t}+\rho^{n+1/2}\left[(U^{\mathrm{ADV}}\cdot\nabla)U\right]^{n+1/2}=1/2(\nabla\cdot\tau^*+\nabla\cdot\tau^n)$$
$$-\nabla\pi^{n-1/2}+\rho^{n+1/2}\vec{g}.$$

At this point $U^{n+1,*}$ does not satisfy the constraint. We apply an approximate projection to simultaneously update the pressure and to project $U^{n+1,*}$ onto the constraint surface. In particular, we solve

$$L^\rho\phi=D(U^{n+1,*}+\frac{\Delta t}{\rho^{n+1/2}}G\pi^{n-1/2})-S^{n+1} \tag{9}$$

for nodal values of ϕ, where L^ρ is the standard bilinear finite element approximation to $\nabla\cdot\frac{1}{\rho}\nabla$ with ρ evaluated at $t^{n+1/2}$. In this step, D is a discrete second-order operator that approximates the divergence at nodes from cell-centered data, and $G=-D^T$ approximates a cell-centered gradient from nodal data. In the formulation, ϕ satisfies Neumann boundary conditions at solid walls and inflow boundaries. At outflow boundaries, Dirichlet conditions are generated to suppress any tangential accelerations on the fluid leaving the domain. Nodal values for S^{n+1} for the solution of (9) are computed using a volume-weighted average of cell-centered values. Finally, we determine the new-time cell-centered velocity field from

$$U^{n+1}=U^{n+1,*}-\frac{\Delta t}{\rho^{n+1/2}}\left(G\phi-G\pi^{n-1/2}\right)$$

and the new time-centered pressure from

$$\pi^{n+1/2}=\phi.$$

This completes the description of the time-advancement algorithm.

Before discussing the incorporation of this methodology in an adaptive mesh refinement algorithm, we note some of the properties of the algorithm. As noted earlier, although the scheme rigorously satisfies conservation of mass and enthalpy, the evolution does not strictly maintain the equation of state at ambient pressure. (It is not possible to preserve all three conditions in a projection-type fractional step scheme.) Since the low Mach number asymptotics used to derive the governing equation show that the thermodynamic pressure only satisfies this constraint to $O(M^2)$, relaxing the

imposition of the constraint is a reasonable way of dealing with splitting errors. To control the deviation from the equation of state we add a correction to the right hand side of equation (5) that approximates

$$\frac{f}{\gamma\rho\frac{\partial p}{\partial\rho}}\left(\frac{\partial p}{\partial t}+U\cdot\nabla p\right).$$

In this expression $\gamma = c_p/c_v$ is the ratio of the two thermodynamic specific heats, and f is a constant relaxation factor. We approximate $\partial p/\partial t$ by $(p_{\text{amb}} - p_0)/\Delta t$, where p_0 is the thermodynamic pressure defined by ρ, h and the Y_k's and p_{amb} is the specified ambient pressure, and $U\cdot\nabla p$ is approximated with upwind differences using p_0. Thus, we are effectively adding a first-order approximation to the material derivative of $p_0 - p_{\text{amb}}$ along streamlines. This forcing term adjusts the advection velocity to drive the evolution toward the constraint, preventing the solution from deviating an appreciable amount from the equation of state while maintaining the second-order accuracy of the overall scheme.

3.2 Adaptive mesh refinement

In this section we present an overview of the adaptive projection algorithm. This framework, used in both Day and Bell [15] and Bell *et al.* [5] was initially developed by Almgren *et al.* [1], and extended to low Mach number combustion by Pember *et al.* [28]. The discussion provides only an overview of the methodology. We refer the reader to the above papers for more details of the basic algorithm.

Our implementation of adaptive mesh refinement (AMR) is based on a sequence of nested grids with successively finer spacing in both time and space. In this approach, fine grids are formed by evenly dividing coarse cells by a refinement ratio, r, in each direction. Increasingly finer grids are recursively embedded in coarse grids until features of the solution are adequately resolved. An error estimation procedure based on user-specified criteria evaluates where additional refinement is needed and grid generation procedures dynamically create or remove rectangular fine grid patches as resolution requirements change.

The adaptive integration algorithm advances grids at different levels using time steps appropriate to that level, based on CFL considerations. The multi-level procedure can most easily be thought of as a recursive algorithm in which, to advance level ℓ, $0 \le \ell \le \ell_{\max}$, the following steps are taken:

- Advance level ℓ in time one time step, Δt^ℓ, as if it is the only level. If $\ell > 0$, obtain boundary data using time-interpolated data from the grids at $\ell - 1$, as well as physical boundary conditions, where appropriate.
- If $\ell < \ell_{\max}$
  - Advance level $(\ell + 1)$ for r time steps, $\Delta t^{\ell+1} = \frac{1}{r}\Delta t^\ell$, using level-$\ell$ data and the physical boundary conditions.
  - Synchronize the data between levels ℓ and $\ell + 1$, and interpolate corrections to finer levels $[\ell + 2, \ldots, \ell_{\max}]$.

The adaptive algorithm, as outlined above, performs operations to advance the grids at each level independent of other levels in the hierarchy (except for boundary conditions) and then computes a correction to synchronize the levels. Loosely speaking, the objective in this synchronization step is to compute the modifications to the coarse grid that reflect the change in the coarse grid solution due to the presence of the fine grid. More specifically, when solving on a fine grid, we supply Dirichlet boundary conditions from the coarse grid. This leads to a mismatch in the associated fluxes at the coarse-fine interface that is corrected by the synchronization.

For the adaptive projection methodology presented here there are three basic steps in the synchronization. First, the values obtained for U, ρY_k and ρh are averaged from the fine grid onto the underlying coarse grid. We view the resulting data as defining a preliminary composite grid solution that is consistent between levels. We denote this preliminary solution with a p superscript in the remainder of the section. To complete the synchronization we need to correct inconsistencies arising from the use of Dirichlet boundary conditions at coarse-fine boundaries. In particular, we compute increments to ρY_k and ρh that correct the flux mismatches at coarse-fine interfaces. Finally, we correct the velocity field to satisfy a divergence constraint on the composite grid system.

There are two components that contribute to flux mismatch. First, U^{ADV}, the edge-based advection velocity satisfies the constraint on the coarse level and the fine level separately. However, since we only satisfy the Dirichlet matching condition for ϕ^{MAC} in (5), the value of U^{ADV} computed on the coarse level does not match the average value on the fine grid. We define the mismatch in advection velocities in d dimensions by

$$\delta U^{\mathrm{ADV},\ell} = -U^{\mathrm{ADV},\ell,n+1/2} + \frac{1}{r^d}\sum_{k=0}^{r-1}\sum_{\mathrm{edges}} U^{\mathrm{ADV},\ell+1,n+k+1/2}$$

along the coarse-fine boundary. We then solve the elliptic equation

$$D^{\mathrm{MAC}}\frac{1}{\rho}G^{\mathrm{MAC}}\delta e^{\ell} = D^{\mathrm{MAC}}\delta U^{\mathrm{ADV},\ell}$$

and compute

$$U^{\mathrm{ADV},\ell,\mathrm{corr}} = -\frac{1}{\rho}G^{\mathrm{MAC}}\delta e^{\ell}$$

which is the correction needed for U^{ADV} to satisfy the constraint and matching conditions on the composite $(\ell,\ \ell+1)$ grid hierarchy. This correction field is used to compute a modification to the advective fluxes for species and enthalpy that reflects an advection velocity field that satisfies the constraint on the composite grid.

The second part of the mismatch arises because the advective and diffusive fluxes on the coarse grid were computed without explicitly accounting for the fine grid, while on the fine grid the fluxes were computed using coarse-grid Dirichlet boundary data. We define the flux discrepancies

$$\delta F_{\rho h} = \Delta t^{\ell} \left(-F_{\rho h}^{\ell,n+1/2} + \frac{1}{r^d} \sum_{k=0}^{r-1} \sum_{edges} F_{\rho h}^{\ell+1,n+k+1/2} \right)$$

and

$$\delta F_{\rho Y_k} = \Delta t^{\ell} \left(-F_{\rho Y_k}^{\ell,n+1/2} + \frac{1}{r^d} \sum_{k=0}^{r-1} \sum_{edges} F_{\rho Y_k}^{\ell+1,n+k+1/2} \right)$$

where F is the total (advective+diffusive) flux through a given interface prior to these synchronization operations. Corrections to density, $\delta\rho^{sync}$, on the coarse grid associated with mismatched advection fluxes may be computed explicitly

$$\delta\rho^{sync} = -D^{MAC} \left(\sum_k U^{ADV,corr} \rho Y_k \right)^{n+1/2} + \sum_k \delta F_{\rho Y_k}.$$

The post-sync new-time value of density is given by $\rho^{n+1} = \rho^{n+1,p} + \delta\rho^{sync}$. Given the corrected density, ρ^{n+1}, we can decompose the corrections for Y_k and h into

$$\delta(\rho Y_k)^{sync} = Y_k^{n+1,p} \delta\rho^{sync} + \rho^{n+1} \delta Y_k^{sync}$$

and

$$\delta(\rho h)^{sync} = h^{n+1,p} \delta\rho^{sync} + \rho^{n+1} \delta h^{sync}.$$

Computing δY_k^{sync} and δh^{sync} requires solution of a linear system, since the flux mismatch contains implicit diffusion fluxes from the Crank-Nicolson discretization. The provisional correction $\delta\tilde{Y}_k^{sync}$ on the coarse level ℓ grids is obtained by solving

$$\left(\rho^{n+1} - \frac{\Delta t}{2} \nabla \rho^{n+1} D_k^{n+1} \nabla \right) \delta\tilde{Y}_k^{sync} = -D^{MAC} \left(U^{ADV,corr} \rho Y_k \right)^{n+1/2} + \delta F_{\rho Y_k}.$$

However, as in the single-level algorithm, the species correction fluxes must sum to zero to preserve mass conservation. We compute the adjusted species diffusion correction fluxes which sum to zero and then define Y_k^{sync} from

$$\rho^{n+1} \delta Y_k^{sync} = \frac{\Delta t}{2} \nabla \overline{\rho^{n+1} D_k^{n+1} \nabla \delta\tilde{Y}_k^{sync}} - D^{MAC} \left(U^{ADV,corr} \rho Y_k \right)^{n+1/2} + \delta F_{\rho Y_k}.$$

We then compute the enthalpy correction from

$$\left(\rho^{n+1} - \frac{\Delta t}{2} \nabla \rho^{n+1} \frac{\kappa^{n+1}}{c_p^{n+1}} \nabla \right) \delta h^{sync} = -D^{MAC} \left(U^{ADV,corr} \rho h \right)^{n+1/2} + \delta F_{\rho h} + \nabla \cdot \xi_k \left(\frac{\kappa^{n+1}}{c_p^{n+1}} \nabla \delta Y_k^{sync} - \overline{\rho^{n+1} D_k^{n+1} \nabla \delta\tilde{Y}_k^{sync}} \right).$$

The corrections, δY_k^{sync}, and δh^{sync} are added to the coarse field at level ℓ, and interpolated to all finer levels. Finally, a new temperature field is computed using the corrected h and Y_k's.

A similar process is also used to generate a correction to the velocity field. However, the velocity flux correction must be projected to obtain the component satisfying the constraint that updates U and the component that updates π. At this point there are two additional corrections needed for the composite velocity field:

- A correction arising because the projection at level $\ell + 1$ used Dirichlet data from level ℓ, leading to a mismatch in normal derivative at coarse-fine boundaries
- The temperature and species adjustment in the first part of the synchronization leads to an increment in the computed S field.

Since the projection is linear, both of these corrections as well as the projection of the velocity flux correction can be combined into a single, multi-level node-based synchronization solve performed at the end of a coarse-grid time step.

We note that with the synchronization procedure outlined above the adaptive algorithm preserves the second-order accuracy and the conservation properties of the single-grid algorithm. The methodology has been implemented for distributed memory parallel processors using the BoxLib class libraries described by Rendleman et al. [30]. Futher discussion of the parallelization of the low Mach number model can be found in Bell *et al.* [4].

4 Combustion

In this section, we present a prototype application of the above methodology to turbulent combustion. The results presented here are taken from a study by Bell *et al.* [7]. The goal of this study was to assess the resolution requirements needed to capture the dominant features of an experimental methane flame using detailed chemistry and transport without explict models for turbulence or turbulence chemistry interaction. A photograph of the experiment is present in Figure 1. In the experiment, a plate with 3.2 mm holes arranged in a 4.8 mm hexagonal lattice is placed in the inflow stream 9 cm below the exit to the nozzle. A 2 mm rod is placed across the nozzle exit where it serves to anchor a turbulent V-flame.

The computational strategy for this simulation is to independently characterize the turbulence generation in the nozzle using experimental data and auxiliary computations. Turbulent fluctuations with appropriate statistical properties are then superimposed on the mean inflow velocity. For the case considered here the inflow velocity is 3 m/sec with a turbulent intensity of approximately 7% in the axial direction and 5% in the transverse directions. The inflow conditions correspond to methane at an equivalence ratio of 0.75 at 300K.

With this characterization of the inflow we solve the low Mach number equations in a cubical domain indicated in Figure 1. For this computation, methane chemistry was modeled using the methane mechanism, DRM-19, developed by Frenklach which contains 19 species and 84 reactions. A mixture model was used for species diffusion. The computation was performed with a base grid of 96^3 with 2 refinements of a factor of 2 each. This lead to an effective resolution of approximately 312.5 μm which corresponds to approximately 2-3 zones across the thermal thickness of the flame.

A picture of the computed flame surface is presented in Figure 2. Figure 3 presents a quantitative comparison of the computed solution to the experimental data. Experimentally, the instantaneous flame location is determined from PIV measurements. Due to the large difference in Mie scattering intensities from the reactants

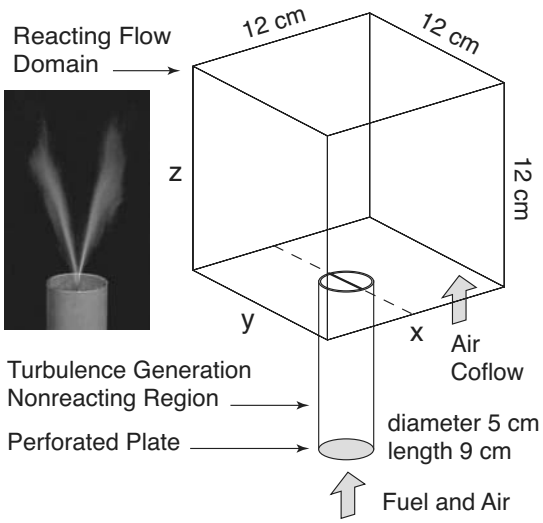


Fig. 1. *Photograph of the experiment and schematic for the V-flame computation*

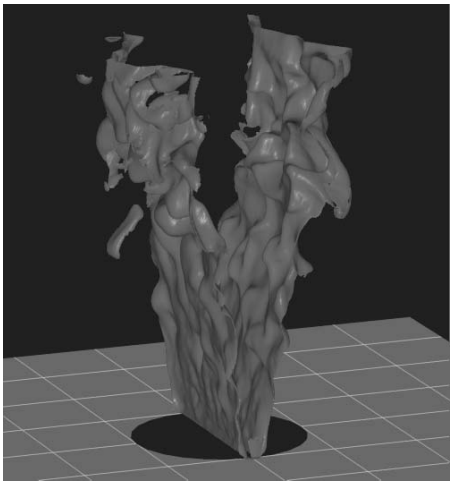


Fig. 2. *Simulated instantaneous flame surface*

and products, the instantaneous wrinkled flame front is clearly outlined on the PIV image (Figure 3b). Compared to a centerline slice of the methane concentration obtained from the simulation (Figure 3a) the wrinkling of the flame in the experiment and the computation is similar in size and structure.

To characterize the flame brush which gives a statistical picture of the flame, the position of the flame fronts were obtained from 100 PIV images by an edge detection

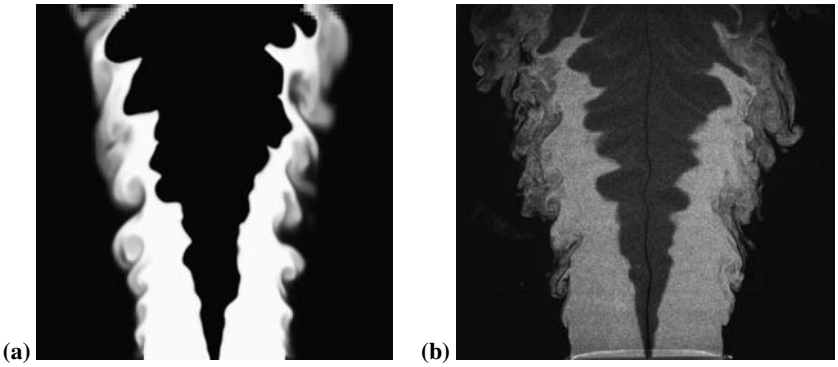


Fig. 3. *(a) Computed CH_4 mole fraction, (b) Typical PIV image.*

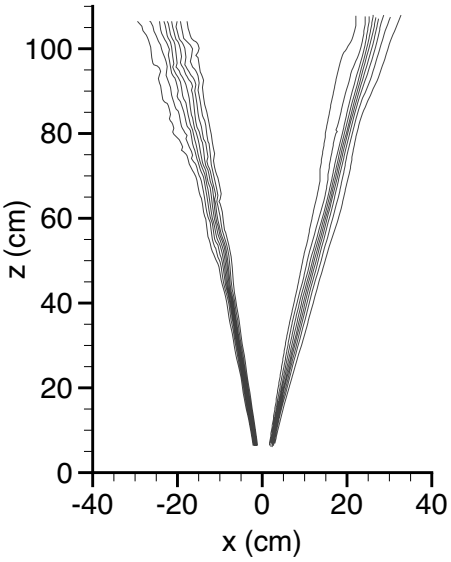


Fig. 4. *Comparison of $\bar{c}$ contours. Left (red) is experiment; right (blue) is simulation*

algorithm. Their average produces a map of the mean reaction progress, $\bar{c}$, where $\bar{c} = 0$ in reactants, and $\bar{c} = 1$ in the products. For the simulation data, we define an instantaneous progress variable $c = (\rho_u - \rho)/(\rho_u - \rho_b)$ where $\rho_{u,b}$ are the densities of the unburned and burned gas, respectively. Averaging c over a sample of slices through the computed flame defines an analogous $\bar{c}$ for the computation.

A comparison of experimental and computational $\bar{c}$ contours which show the growth of the flame brush is given in Figure 4. The simulation shows excellent agreement up to approximately $z < 10$ cm at which point the computational results begin to show the effects of the outflow imposed at $z = 12$ cm. The simulation and the experimental results have slightly different included angles; the $\bar{c} = 0.5$ contour forms an angle of approximately 11° with the vertical in the experiment compared to 13° for the computation. Additional comparisons with the experimental data are presented in Bell *et al.* [7].

5 Nuclear flames

In the second example, we consider $C + O$ nuclear flames in a carbon-oxygen white dwarf typical of a Type Ia supernova. The results presented here are taken from a study of the effect of Rayleigh-Taylor instabilites on flame propagation at conditions corresponding to the late stages of a type Ia supernova by Bell *et al.* [6]. For this example a generalized equation of state is needed to describe the fluid. In particular, for the stellar conditions being considered here the pressure contains contributions from ions, radiation, and electrons. (See Kippenhahn and Weigert [20] for a discussion of equations of state for stellar matter.) Thus,

$$p = p_{\text{ion}} + p_{\text{rad}} + p_{\text{ele}} \tag{10}$$

with

$$p_{\text{ion}} = \frac{\rho k T}{\bar{A} m_p} \quad , \quad p_{\text{rad}} = a T^4/3$$

and p_{ele} is the contribution to the thermodynamic pressure due to fermions. In these expressions, m_p is the mass of the proton, a is related to the Stefan-Boltzmann constant $\sigma = ac/4$, c is the speed of light, $1/\bar{A} = X_k/A_k$, and k is Boltzmann's constant. The ionic component has the form associated with an ideal gas but the radiation and electron pressure components do not. The numerical simulations were performed using the equation of state described by Timmes and Swesty [33] which computes the internal energy, pressure and thermodynamic derivatives (including the specific heats at constant volume and pressure) of these quantities as functions of temperature, density and the nuclear-species mass fractions.

For the stellar conditions typical of C + O flames we are considering here, the Lewis number, which is the ratio of energy transport to species diffusion, is $O(10^7)$ and the Prandtl number, which is the ratio of fluid viscosity to energy transport, is $O(10^{-5})$. For these conditions, we can ignore both fluid viscosity and species diffusion. The values of the thermal conductivity, κ, are calculated using the procedure described by Timmes [32].

We present computations corresponding to fuel densities of 1.5×10^7 g/cc and 1.0×10^7 g/cc. For each simulation, $g = 10^9 \text{cm/sec}^2$ which is appropriate for the outer region of the white dwarf after some pre-expansion has taken place. We flow fuel in from the top of the domain at the laminar flame velocity and impose an outflow boundary condition at the bottom of the domain. The lateral sides of the domain

Fig. 5. *Time sequence of Rayleigh Taylor unstable nuclear flame at density fuel density* 1.5×10^7 *g/cc. Times are 0.0, 0.7, 1.1 and 1.4 msec. The image shows carbon mass fraction with red to blue corresponding to high to low values.*

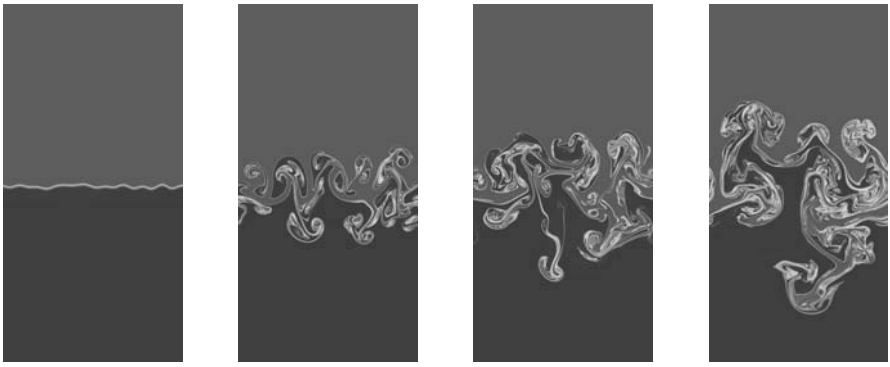


Fig. 6. *Time sequence of Rayleigh Taylor unstable nuclear flame at density fuel density* 1.0×10^7 *g/cc. Times are 0.0, 1.2, 1.8, and 2.3 msec. The image shows carbon mass fraction with red to blue corresponding to high to low values.*

are periodic. Each computation is performed with one level of refinement with an effective resolution that is approximately 10% of the thermal flame thickness. The domains are 163.84 cm $\times$327.7 cm and 53.5 cm $\times$107 cm for the low and high density cases, respectively, which correspond to approximately 90 thermal flame thicknesses in each case.

Figures 5 and 6 correspond to the high and low density cases, respectively. This range of densities represents a transition from flamelet combustion at the higher density in which a clearly defined flame front is apparent to a distributed flame in which the reactions occur in dispersed pockets at the lower density. Analyses of these types of simulations are being used to help determine how much the flame is accelerated by interaction with the Rayleigh-Taylor instability which plays a key role in determining the mechanisms that lead to type Ia supernova explosions.

6 Conclusion

We have presented an adaptive algorithm for reacting flows based on a low Mach number formulation. This formulation, which leads to constrained evolution equations, analytically removes sound waves from the system and allows time steps to be chosen based on the advective time scales. The low Mach number system is integrated using a fractional step projection algorithm that evolves the system without enforcing the constraints and then projects the solution back onto the constraint manifold. The basic projection discretization has been incorporated into a parallel adaptive mesh refinement algorithm. We have shown two examples of problems that would have been infeasible without this type of methodology.

Acknowledgement

This work was supported the Applied Mathematics Program of the DOE Office of Mathematics, Information, and Computational Sciences under contract number DE-AC03-76SF00098.

References

1. A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comput. Phys.*, 142:1–46, 1998.
2. A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. 142:1–46, May 1998.
3. R. Becker, M. Braack, and R. Rannacher. Numerical simulation of laminar flames at low mach number by adaptive finite elements. *Combust. Theory Modelling*, 3:503–534, 1999.
4. J. B. Bell, M. S. Day, A. S. Almgren, M. J. Lijewski, and C. A. Rendleman. A parallel adaptive projection method for low Mach number flows. *Int. J. Numer. Meth. Fluids*, 40:209–216, 2002.
5. J. B. Bell, M. S. Day, C. A. Rendleman, S. E. Woosley, and M. A. Zingale. Adaptive low mach number simulations of nuclear flames. Technical Report LBNL-52395, Lawrence Berkeley National Laboratory, March 2003. to appear in J. Comp. Phys.
6. J. B. Bell, M. S. Day, C. A. Rendleman, S. E. Woosley, and M. A. Zingale. Direct numerical simulations of type ia supernovae flames ii: The rayleigh-taylor instability. Technical Report LBNL-54300, Lawrence Berkeley National Laboratory, January 2004. submitted Astrophysical Journal.
7. J. B. Bell, M. S. Day, I. G. Shepherd, M. Johnson, R. K. Cheng, V. E. Beckner, M. J. Lijewski, and J. F. Grcar. Numerical simulation of a laboratory-scale turbulent v-flame. Technical Report LBNL-54198, Lawrence Berkeley National Labortory, 2003. submitted Proceedings of the Combustion Institute.
8. J. B. Bell and D. L. Marcus. A second-order projection method for variable density flows. 101(2):334–348, August 1992.
9. B. A. V. Bennett, C. S. McEnally, L. D. Pfefferle, and M. D. Smooke. Computational and experimental study of axisymmetric coflow partially premixed methane/air flames. *Combust. Flame*, 123:522–546, 2000.

10. B.A.V. Bennett and M. D. Smooke. Local rectangular refinment with application to axisymmetric laminar flames. *Combust. Theory Modelling*, 2:221–258, 1998.
11. R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport Phenomena*. Wiley, New York, 2 edition, 2002.
12. P. N. Brown, G. D. Byrne, and A. C. Hindmarsh. VODE: A variable coefficient ode solver. *SIAM J. Sci. Stat. Comput.*, 10:1038–1051, 1989.
13. P.J. Coelho and J.C.F. Pereira. Calculation of a confined axisymmetric laminar diffusion flame using a local grid refinement technique. *Combust. Sci. Tech.*, 92:243–264, 1993.
14. T. P. Coffee and J. M. Heimerl. Transport algorithms for premixed laminar steady-state flames. *Comb. Flame*, 43:273, 1981.
15. M. S. Day and J. B. Bell. Numerical simulation of laminar reacting flows with complex chemistry. *Combust. Theory Modelling*, 4:535–556, 2000.
16. H.C. de Lange and L.P.H. de Goey. Numerical modeling in a locally refined grid. *Int. Jour. Num. Mech. Eng.*, 37:497–515, 1994.
17. V. Giovangigli. Convergent iterative methods for multicomponent diffusion. *IMPACT Comput. Sci. Engrg.*, 3:244–276, 1991.
18. F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluids with free surfaces. *Physics of Fluids*, 8:2182–2189, 1965.
19. J. Hilditch and P. Colella. A front tracking method for compressible flames in one dimension. *SIAM J. Sci. Comput.*, 16:755–772, 1995.
20. Rudolf Kippenhahn and Alfred Weigert. *Stellar Structure and Evolution*. Springer Verlag, 1992.
21. M. F. Lai. *A Projection Method for Reacting Flow in the Zero Mach Number Limit*. PhD thesis, University of California at Berkeley, 1993.
22. M. F. Lai, J. B. Bell, and P. Colella. A projection method for combustion in the zero Mach number limit. In *Proceedings of the Eleventh AIAA Computational Fluid Dynamics Conference*, pages 776–783, 1993.
23. A. Majda and J. A. Sethian. The derivation and numerical solution of the equations for zero Mach number combustion. *Combust. Sci. Tech.*, 42:185–205, 1985.
24. R.M.M. Mallens, H.C. de Lange, C.H.J. van de Ven, and L. P. H. de Goey. Modeling of confined and unconfined laminar premixed flames on slit and tube burners. *Combust. Sci. Tech.*, 107:387–401, 1995.
25. P.A. McMurtry, W.-H. Jou, J.J. Riley, and R.W. Metcalfe. Direct numerical simulations or a reacting mixing layer with chemical heat release. *AIAA J.*, 24:962, 1986.
26. H.N. Najm, R.W. Schefer, R.B. Milne, C.J. Mueller, K.D. Devine, and S.N. Kempka. Numerical and experimental investigation of vortical flow–flame interaction. Technical Report SAND98-8232, Sandia National Laboratory, February 1998.
27. R. B. Pember, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Lai. A higher-order projection method for the simulation of unsteady turbulent nonpremixed combustion in an industrial burner. *Transport Phenomena in Combustion*, pages 1200–1211, 1996. Taylor and Francis.
28. R. B. Pember, L. H. Howell, J. B. Bell, P. Colella, W. Y. Crutchfield, W. A. Fiveland, and J. P. Jessee. An adaptive projection method for unsteady, low-Mach number combustion. *Comb. Sci. Tech.*, 140:123–168, 1998.
29. R.G. Rehm and H.R. Baum. The equations of motion for thermally driven buoyant flows. *N. B. S. J. Res.*, 83:297–308, 1978.
30. Charles A. Rendleman, Vincent E. Beckner, Mike Lijewski, William Y. Crutchfield, and John B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3(3):147–157, 2000.

31. L.T. Somers and L.P.H. de Goey. A numerical study of a premixed flame on a slit burner. *Combust. Sci. Tech.*, 108:121–132, 1995.
32. Frank X. Timmes. The physical properties of laminar helium deflagrations. *Astrophysical Journal*, 528:913, 2000.
33. Frank X. Timmes and F. Douglas Swesty. The accuracy, consistency, and speed of an electron-positron equation of state based on table interpolation of the helmholtz free energy. *Astrophysical Journal Supplement*, 126:501–516, 2000.
34. M. D. Smooke A. A. Turnbull, , R. E. Mitchell, and D. E. Keyes. Solution of two-dimensional axisymmetric laminar diffusion flames by adaptive boundary value methods. In *Mathematical Modeling in Combustion and Related Topics*, pages 261–300, 1988.

Simulations of Relativistic Astrophysical Flows

J. Bergmans[1], R. Keppens[2], D.E.A. van Odyck[3], A. Achterberg[1]

[1] Sterrenkundig Instituut, Utrecht
[2] Instituut voor Plasmafysica "Rijnhuizen", Nieuwegein
[3] Centrum voor Wiskunde en Informatica, Amsterdam

The multidimensional grid-adaptive magnetofluid dynamics code AMRVAC [1, 2] has been extended with special relativistic hydrodynamics and magnetohydrodynamics modules to simulate the complex and ultra-relativistic flow dynamics associated with astrophysical objects like Gamma Ray Bursts and Active Galactic Nuclei. The shock capturing numerical scheme and mesh refinement algorithm are crucial to resolve the strong shocks and wide range of length scales present in these flows.

The first results look promising, we can simulate strongly relativistic blast waves, fireballs, and jets in 1D, 2D, and 3D.

1 Introduction

Relativistic flows are directly observed, or indirectly inferred, in a variety of astrophysical objects. Accretion flows and collimated jets around massive black holes in Active Galactic Nuclei have Lorentz factors of order 10, which means velocities of 99.5% of the speed of light (the Lorentz factor is defined as $\Gamma = (1 - v^2/c^2)^{-1/2}$). Even stronger relativistic flows are presumed to be the origin of Gamma Ray Bursts, the most energetic events in the observable universe since the Big Bang. Energy and time scales suggest that the source must be accompanied by an extremely relativistic explosion or jet flow with Lorentz factors in the order of 10^2 to 10^3, almost certainly also in the form of a collimated jet. Strong shocks and magnetic fields are responsible for the production of high-energy particles and radiation from these systems.

Numerical simulations of the dynamical interplay between relativistic flow, strong shocks and magnetic fields are valuable to describe and understand the observed properties of relativistic astrophysical objects. The calculation will have to be capable of resolving the large-scale flows and at the same time accurately capture the small-scale dynamics and shocks. This wide range of dynamical length scales makes it necessary to employ adaptive mesh refinement techniques.

Because the flows under consideration are in general on a much larger scale than the size of the central compact object, it is in first approximation reasonable to describe the system in flat space-time, i.e. using the special relativistic incarnation of

ideal (magneto-)hydrodynamics. The governing equations are given in Sec. 2. In Sec. 3 we describe the numerical code AMRVAC which has been extended with relativistic HD and MHD physics modules. In Sec. 4 some test problems and results for both relativistic HD and MHD calculations are shown.

2 Equations for Relativistic Fluids

The fluid equations follow from particle conservation, the conservation of the energy-momentum tensor, and an induction equation for the magnetic field. The continuity equation (in units where $c = 1$, Greek indices denote four-vectors and Latin indices denote three-vectors) is given by

$$\partial_\alpha(\rho u^\alpha) = 0\,, \tag{1}$$

where ρ is the rest mass density, $u^\alpha = \Gamma(1, \vec{v})$ is the four-velocity for which $u_\alpha u^\alpha = -1$, and the Lorentz factor is $\Gamma = (1 - v^2)^{-1/2}$.

The second ingredient is the conservation of the stress-energy tensor

$$\partial_\alpha(T_{\rm fl}^{\alpha\beta} + T_{\rm em}^{\alpha\beta}) = 0\,. \tag{2}$$

For pure hydrodynamics the electromagnetic part is absent, so for an ideal fluid the stress-energy tensor is given by (See [3])

$$T^{\alpha\beta} = T_{\rm fl}^{\alpha\beta} = w u^\alpha u^\beta + p\eta^{\alpha\beta}\,. \tag{3}$$

The relativistic enthalpy is $w = e + p$, where the energy per unit volume e includes the rest mass energy, and p is the kinetic pressure. The Minkowski metric tensor is $\eta^{\alpha\beta} = \mathrm{diag}\{-1, 1, 1, 1\}$. All thermodynamic quantities (ρ, p, e, w) are measured in the rest frame of the fluid. To close the system of equations we need an equation of state relating the kinetic pressure with density and internal energy, in this work we take a polytropic gas law with adiabatic index γ, for which the enthalpy is given by

$$w = \rho + \frac{\gamma}{\gamma - 1} p\,. \tag{4}$$

For real-world applications we separate the temporal from the spatial components and the equations of special relativistic hydrodynamics (from here onwards abbreviated as RHD) read

$$\partial_t(\Gamma\rho) + \partial_i(\Gamma\rho v^i) = 0\,, \tag{5}$$

$$\partial_t(\Gamma^2 w v^j) + \partial_i(\Gamma^2 w v^i v^j + p\delta^{ij}) = 0\,, \tag{6}$$

$$\partial_t(\Gamma^2 w - p) + \partial_i(\Gamma^2 w v^i) = 0\,. \tag{7}$$

To arrive at a set of equations describing a relativistic *magneto*fluid (special relativistic magnetohydrodynamics or RMHD) we need to add the electromagnetic part

$T^{\alpha\beta}_{\rm em}$ to the momentum-energy tensor and add an equation for the electromagnetic field. Using the field tensor $F^{\alpha\beta}$ ($F^{\alpha\beta} = -F^{\beta\alpha}$, $E_i = F^{0i}$ and $B_i = \frac{1}{2}\varepsilon^{ijk}F^{jk}$) and its dual $\mathcal{F}^{\alpha\beta} = \frac{1}{2}\varepsilon^{\alpha\beta\gamma\delta}F_{\gamma\delta}$ the Maxwell equations in covariant form are

$$\partial_\alpha F^{\alpha\beta} = -j^\beta\,, \qquad \partial_\alpha \mathcal{F}^{\alpha\beta} = 0\,. \tag{8}$$

The inhomogeneous equation is used to calculate the four-current j^α. The current density in RMHD is given by $\vec{J} = \nabla\times\vec{B} - \partial_t\vec{E}$. The homogeneous equation comprises both the induction equation and $\nabla\cdot\vec{B} = 0$. The electromagnetic contribution to the momentum-energy tensor is given by

$$T^{\alpha\beta}_{\rm em} = F^\alpha{}_\gamma F^{\gamma\beta} + \tfrac{1}{4}\eta^{\alpha\beta}F_{\gamma\delta}F^{\gamma\delta}\,. \tag{9}$$

Ohm's law for a perfectly conducting fluid ($\vec{E} + \vec{v}\times\vec{B} = 0$) is written covariantly as

$$F^{\alpha\beta}u_\beta = 0\,. \tag{10}$$

This expression can be used to eliminate $\vec{E}$ from the equations. This is done by introducing the four-vector $b^\alpha \equiv \mathcal{F}^{\alpha\beta}u_\beta$. The field tensor can then be written in the form $F^{\gamma\delta} = \varepsilon^{\alpha\beta\gamma\delta}b_\alpha u_\beta$ The components of this new vector are given by

$$b^\alpha = \left[\Gamma(\vec{v}\cdot\vec{B}), \vec{B}/\Gamma + \Gamma(\vec{v}\cdot\vec{B})\vec{v}\right], \tag{11}$$

and we can write the momentum-energy tensor (9) in the following form (note that $b^\alpha u_\alpha = 0$ and that $|b|^2 = b^\alpha b_\alpha = (\vec{v}\cdot\vec{B})^2 + \vec{B}^2/\Gamma^2$)

$$T^{\alpha\beta}_{\rm em} = |b|^2\,(u^\alpha u^\beta + \tfrac{1}{2}\eta^{\alpha\beta}) - b^\alpha b^\beta\,. \tag{12}$$

The energy-momentum conservation equation (2) takes the form

$$\partial_\alpha[w_{\rm tot}u^\alpha u^\beta - b^\alpha b^\beta + p_{\rm tot}\eta^{\alpha\beta}] = 0\,, \tag{13}$$

where $w_{\rm tot} = w + |b|^2$ and $p_{\rm tot} = p + \frac{1}{2}|b|^2$. The induction equation from (8) transforms to

$$\partial_\alpha(u^\alpha b^\beta - b^\alpha u^\beta) = 0\,. \tag{14}$$

The set of RMHD equations is composed of the continuity equation (1), the temporal and spatial parts of the momentum-energy equation (13), and the field equation (14). Again we split these equations in their temporal and spatial parts to obtain

$$\partial_t(\Gamma\rho) + \partial_i(\Gamma\rho v^i) = 0\,, \tag{15}$$

$$\partial_t\left(\Gamma^2 w_{\rm tot}v^j - b^0 b^j\right) + \partial_i\left(\Gamma^2 w_{\rm tot}v^i v^j - b^i b^j + p_{\rm tot}\delta^{ij}\right) = 0\,, \tag{16}$$

$$\partial_t\left(\Gamma^2 w_{\rm tot} - (b^0)^2 - p_{\rm tot}\right) + \partial_i\left(\Gamma^2 w_{\rm tot}v^i - b^0 b^i\right) = 0\,, \tag{17}$$

$$\partial_t B^j + \partial_i\left(v^i B^j - B^i v^j\right) = 0\,, \qquad \partial_i B^i = 0\,. \tag{18}$$

The RHD equations (5)–(7) may be obtained from these equations simply by taking $\vec{B} = 0$; the non-relativistic MHD equations emerge in the limit $v^2 \ll 1$. When taking this limit one needs to subtract the contribution of the rest-mass energy density from the energy equation. The fact that the magnetic field equations are unchanged from classical MHD is of course due to the fact that Maxwell's equations are already Lorentz invariant.

2.1 Recovering the Primitive Variables

Although (15)–(18) are already in conservation form, we will rewrite them in the form actually used in the code:

$$\partial_t \vec{U} + \sum_i \partial_i \vec{F}^i = 0\,, \tag{19}$$

where $\vec{U} = [D, \vec{S}, E, \vec{B}]^T$ with $D \equiv \Gamma\rho$, $\vec{S} \equiv (\Gamma^2 w + B^2)\vec{v} - (\vec{v}\cdot\vec{B})\vec{B}$ and $E \equiv \Gamma^2 w + B^2 - p_{\rm tot}$ are conserved variables. The fluxes are given by

$$\vec{F}^i = \vec{U} v^i + [0, -\left(\vec{B}/\Gamma^2 + (\vec{v}\cdot\vec{B})\vec{v}\right) B^i + p_{\rm tot}\vec{I}^i, p_{\rm tot} v^i - (\vec{v}\cdot\vec{B})B^i, -\vec{v}B^i]^T.$$

$\vec{I}^i$ denotes the ith column of the identity matrix.

To numerically advance the conserved variables $(D, \vec{S}, E, \vec{B})$ in time one needs to calculate the values of the primitive variables $(\rho, \vec{v}, p, \vec{B})$ in order to determine the fluxes. In classical (M)HD this is an almost trivial operation, but due to the coupling between variables by various powers of the Lorentz factor we have to compute the primitive variables by numerically solving a nonlinear equation.

Following the method outlined in [4] we reduce the problem to an equation for $\xi \equiv \Gamma^2 w$, which is then solved by the Newton–Raphson method. First note that from the definition of $\vec{S}$ follows that $\vec{v}\cdot\vec{B} = \xi^{-1}\vec{S}\cdot\vec{B}$ and the velocity can be expressed in conserved variables and ξ only:

$$\vec{v} = \left(\vec{S} + \xi^{-1}(\vec{S}\cdot\vec{B})\vec{B}\right)/(\xi + B^2)\,. \tag{20}$$

Using the equation of state (4), the density and kinetic pressure are

$$\rho = D/\Gamma, \qquad p = (\gamma - 1)\left(\xi/\Gamma^2 - D/\Gamma\right)/\gamma\,. \tag{21}$$

From the definition of the total energy E we find a single equation for ξ:

$$f(\xi) = \xi - \frac{\gamma - 1}{\gamma}\left(\frac{\xi}{\Gamma^2} - \frac{D}{\Gamma}\right) + B^2 - \tfrac{1}{2}\left[\frac{B^2}{\Gamma^2} + \frac{(\vec{S}\cdot\vec{B})^2}{\xi^2}\right] - E = 0\,, \tag{22}$$

where the Lorentz factor is given by

$$1/\Gamma^2 = 1 - \left(\vec{S} + \xi^{-1}(\vec{S}\cdot\vec{B})\vec{B}\right)^2/(\xi + B^2)^2\,. \tag{23}$$

Given a set of values for the conserved variables $(D, \vec{S}, E, \vec{B})$ (22) is solved by means of the Newton–Raphson method. By (20) and (21) the complete set of primitive variables is then determined.

From (23) we see that $\Gamma^2 \geq 1$ and thus $v^2 < 1$, as required. The positivity of the density directly leads to the positivity of D. The condition $p > 0$ translates into the condition

$$\xi^2/\Gamma^2 > D^2\,. \tag{24}$$

This condition is satisfied for $\xi > \xi_0$, where ξ_0 is the maximum root of the fourth order equation $\xi_0^2/\Gamma^2 = D^2$. Because $f(\xi)$ is a monotonically increasing function of ξ, we find that if $f(\xi_0) < 0$ then (22) has a single solution representing a physically allowable state.

2.2 Calculating Propagation Speeds

The numerical fluxes for the TVDLF method used in this work are based on the maximum propagation speed for a given state (see section 3). As the equations themselves are formulated in the observers frame, the propagation speeds are measured in that same frame. Although at the moment we need only the largest eigenvalue for our numerical scheme, in the future one might want to implement more sophisticated solvers based on more or all characteristic speeds, so for completeness we summarize all waves. For details we refer to [5, 4, 6] and references therein.

The system of RMHD equations admits seven waves: two Alfvén, two fast and two slow magnetosonic waves and one entropy wave. In contrast to classical MHD the symmetry of the wave speeds with respect to the fluid velocity is lost, but the ordering is still maintained: $\lambda_F^-, \lambda_A^-, \lambda_S^-, \lambda_0, \lambda_S^+, \lambda_A^+, \lambda_F^+$. For waves propagating along the x-axis in the observers frame, the entropy wave propagates with the fluid velocity: $\lambda_0 = v_x$. The eigenvalues of the two Alfvén waves are given by

$$\lambda_A^{\pm} = (u^x \sqrt{w_{\rm tot}} \pm b^x)/(u^0 \sqrt{w_{\rm tot}} \pm b^0) \,. \tag{25}$$

In our numerical scheme, however, we are interested in the eigenvalue with the largest absolute value for a given state (i.e. one of the fast magnetosonic waves) and unfortunately in RMHD there is no simple analytical expression for the magnetosonic eigenvalues. The eigenvalues $\lambda_{F,S}^{\pm}$ are the roots of the following quartic equation:

$$\begin{aligned} & w(e'_p - 1)(u^0\lambda - u^x)^4 \\ & \qquad - (1-\lambda^2)\left[(w + e'_p\,|b|^2)(u^0\lambda - u^x)^2 - (b^0\lambda - b^x)^2\right] = 0 \,, \end{aligned} \tag{26}$$

where $e'_p = (\partial e/\partial p)_s = c_s^{-2}$, with $c_s = (\gamma p/w)^{1/2}$ the gas dynamic sound speed. Rewriting the left hand side yields a simple fourth order polynomial in λ, but the exact analytical solution for its roots is cumbersome and inefficient to calculate numerically in the code. There is an extra complication: when the fluid velocity approaches the speed of light, the eigenvalues all tend to unity and lie very close together. This is a notoriously difficult case for root finding routines, so we transform to a new variable $\mu \equiv 1/(1-\lambda)$ and multiply the resulting equation by μ^4 resulting in a fourth order polynomial equation for μ. The roots of this equation are well separated (for $\lambda \to 1$ we have $\mu \to \infty$) and can be found using a root finding technique. In our code we employ Laguerre's method [7] for polynomials.

3 The AMRVAC Code

The Versatile Advection Code (VAC, see [8]) is a finite volume based numerical code to solve a system of conservation laws on a fixed computational grid. Recently, the code AMRVAC (Adaptive Mesh Refinement VAC) has been developed to extend the capabilities of VAC to calculations with dynamical and automatic grid refinement [1, 2]. The refinement method is summarized in Sec. 3.1. Due to the use of LASY syntax (Loop Annotated Syntax, see [9]) the code can easily be configured to run in 1D, 2D or 3D, and with one, two or three components for vector quantities. In 1D and 2D grids with cylindrical and spherical symmetry are selectable. Available physics modules are ideal hydrodynamics, ideal and resistive magnetohydrodynamics, and, as described here, also relativistic hydro- and magnetohydrodynamics.

The scheme we use for this work is a Lax–Friedrich type Total Variation Diminishing (TVD) scheme called TVDLF. The finite volume discretization of the conservation law (19) is

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}\left(f_{i+1/2}^n - f_{i-1/2}^n\right), \tag{27}$$

where U_i^n is the cell average value of the conserved variable U at the discrete time t^n for the i-th mesh cell with interfaces indicated by $i \pm 1/2$. The numerical fluxes are

$$f_{i+1/2}^n = \tfrac{1}{2}\left(F(U_{i+1/2}^L) + F(U_{i+1/2}^R) - c_{\max}\left[U_{i+1/2}^R - U_{i+1/2}^L\right]\right), \tag{28}$$

where the maximum propagation speed is $c_{\max} = \max\left(\left|\lambda_{\max}(U^L)\right|, \left|\lambda_{\max}(U^L)\right|\right)$ and the states left and right of the interface are

$$U_{i+1/2}^L = U_i^n + \tfrac{1}{2}\Delta\bar{U}_i^n, \qquad U_{i+1/2}^R = U_{i+1}^n - \tfrac{1}{2}\Delta\bar{U}_{i+1}^n. \tag{29}$$

In the work presented here we use *minmod* limited slopes $\Delta\bar{U}_i^n$. Time integration is done using an explicit second order predictor-corrector scheme.

In 2D and 3D MHD calculations various schemes are selectable to ensure that $\nabla \cdot \vec{B} = 0$ is preserved during the calculation (See [2] for details). Although most of these schemes are straightforwardly adaptable to the relativistic case, we did not yet perform runs testing their results for RMHD. The simulations shown in this paper are therefore without any scheme to control $\nabla \cdot \vec{B}$ errors in the calculations.

3.1 AMR Strategy

Our AMR implementation closely follows the original algorithm in [10] for creating a properly nested hierarchy of 'optimally fitted' grids. An important difference is that during the automated creation of 'rectangular' grids, we algorithmically exclude the formation of overlapping grids at the same nesting level and enforce a minimal efficiency of each individual new grid. This efficiency is determined as the ratio of cells identified for refinement to the total number of cells contained within the rectangle. Specifically, once an arbitrary number of cells is flagged for refinement, one subsequently surrounds each cell with a user-set zone of buffer cells (connected to

the temporal frequency with which a full regridding cycle is activated to ensure consistency with the CFL constraint); which are then processed into non-overlapping rectangles by a sequence of mergers and bisections. Note that this approach differs from the Berger and Rigoutsous [11] algorithm exploiting edge detection from pattern recognition techniques. Similarly though, our AMR implementation creates optimally fitted, but very different sized higher level grids. Parallelization is done using `OpenMP` which was shown to work effectively on up to several tens of processors on `ccNUMA` architectures [1, 12, 2]. The parallelism is exploited over the grids per refinement level, and works well when the number of grids on each grid level exceeds the number of processors in use.

The refinement criterion is a variant of the Richardson-type error estimation procedure from Berger and Collela [13], which is generally applicable to any system of equations and integration method used. Needed are two solution vectors $\vec{U}_l^{n-1}$ and $\vec{U}_l^n$ on level l, separated in time by the corresponding Δt_l^{n-1}. The error is estimated as follows:

- coarsen the $\vec{U}_l^{n-1}$ spatially by a factor of 2 to obtain $\vec{U}_{2\Delta x_l}^{n-1}$;
- advance $\vec{U}_{2\Delta x_l}^{n-1}$ using $2\Delta t_l^{n-1}$ to $\vec{U}_{\rm c}^{n+1}$;
- advance $\vec{U}_l^n$ with Δt_l^{n-1} to $\vec{U}_{\rm f}^{n+1}$;
- spatially coarsen (average) $\vec{U}_{\rm f}^{n+1}$ to $\overline{U_{\rm f}}$
- using the solutions $\vec{U}_{\rm c}^{n+1}$, $\vec{U}_{\rm f}^{n+1}$, and $\overline{U_{\rm f}}$ all corresponding to time $t = t_l^{n-1} + 2\Delta t_l^{n-1}$, flag for refinement on the basis of a selection of weighted tolerance criteria.

The latter typically compares selected components of the conserved variables, chosen problem-dependently with mutual weighting factors. One then flags for refinement when a critical tolerance level $\varepsilon_{\rm tol}$ is exceeded. It is possible to extend the refinement criteria with user-forced refinement or derefinement, or exploit primitive or auxiliary variables (like the Lorentz factor for the relativistic simulations discussed here) in the flagging criterion. The need for additionally exploiting uncoarsened data $\vec{U}_{\rm f}^{n+1}$ is advocated by Bell et al. [14]. As a result, where the criterion is fulfilled, a multitude of 2^D level l cells are flagged, for a D-dimensional problem.

We point out that in AMRVAC, we can already allow for a level-dependent choice of the discretization scheme employed. This is a first step toward full Adaptive Mesh and Algorithm Refinement (AMAR), introduced by Garcia et al. [15], who coupled a continuum description with a particle method on the finest level of the AMR hierarchy. The possibility to couple different physics modules across the various grid levels is due to the fact that the only communication between different level solutions is through (i) the update process that is needed to keep the different level solutions consistent and to restore global conservation and (ii) the filling of 'ghost' cells as used for imposing boundary conditions. It is conceivable that once a suitable mapping of the fluxes used in the update process for non-relativistic versus fully relativistic simulations is properly implemented, one could solve for (M)HD flow problems where both non-relativistic versus ultra-relativistic flow regions are essential ingredients. On the various grid levels, one would then solve the corresponding HD, MHD,

RHD, or RMHD equations depending on the locally occurring flow velocities and magnetic field strengths.

4 Results

With the methods outlined in the previous sections various test runs were performed.For all runs we took $\gamma = 4/3$ for the adiabatic constant.

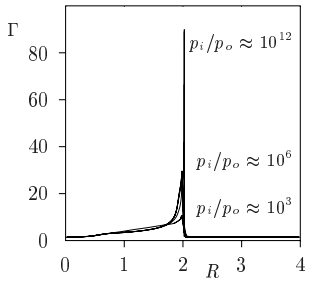


Fig. 1. The Lorentz factor Γ versus radius for spherical explosions with different initial pressure ratios. A base grid of 1000 cells and four refinement levels with a refinement factor of 4 on each level was used, resulting in an effective resolution of 128 000 cells.

First a simple 1D spherical RHD explosion was set up by putting a high pressure in a small region around the origin and simulating the resulting expansion. At $t = 0$ we take $(\rho, v_r, p) = (3.0, 0, p_i/p_o)$ for $0 < r \leq 0.5$ and $(\rho, v_r, p) = (0.001, 0, p_o)$ for $0.5 < r \leq 4.0$, where $p_o = 0.001$ in all cases. The outgoing blast wave sweeps up matter in a thin and dense shell. This shell lies between the forward shock and a contact discontinuity, and due to Lorentz contraction its thickness, and thus the separation between the two discontinuities, is inversely proportional to the Lorentz factor of the blast wave. By increasing the initial pressure ratio p_i/p_o we increase the Lorentz factor of the resulting blast wave, as can be seen in Fig. 1. The extreme pressure contrast and thinness of the shock wave require locally very high grid resolutions, which shows that AMR is indispensable in relativistic simulations.

Next we performed two 1D RMHD shock-tube tests, the first is the relativistic analog of the Brio–Wu shock tube test, see Fig. 2. Initial conditions are $(\rho, v_x, v_y, p, B_x, B_y) = (1.0, 0, 0, 1.0, 0.5, 1.0)$ for $x \leq 0.5$ and $(0.125, 0, 0, 0.1, 0.5, -1.0)$ for $x > 0.5$. The second (Fig. 3) is a flow collision problem with two $\Gamma = 100$ flows colliding at $x = 0.5$. Initial conditions are $(\rho, v_x, v_y, p, B_x, B_y) = (1.0, \pm 0.99995, 0, 0.1, 10.0, \pm 7.0)$ for $x \lessgtr 0.5$. In both cases the AMR algorithm accurately tracks all shocks and waves (4 levels with a refinement ratio of 4 on each level, the base level consists of 50 cells). In the flow collision problem a noticeable 'wall heating' error around $x = 0.5$ occurs, but otherwise the code handles the extreme shocks satisfactory.

A more astrophysically relevant simulation is the axisymmetric 2D RHD relativistic jet shown in Fig. 4. A jet with radius $r_0 = 1.0$, density $\rho = 0.1$ and pressure

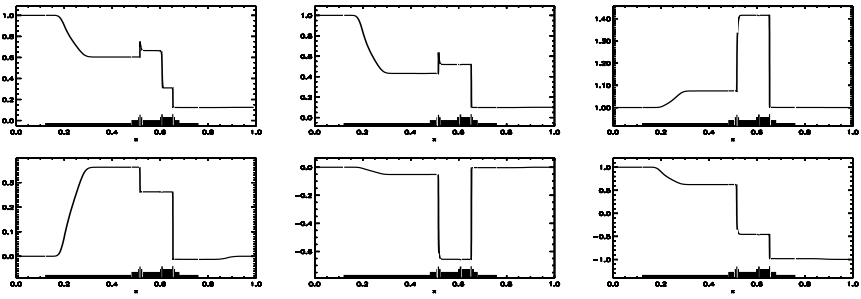


Fig. 2. Relativistic analog of the Brio–Wu shock tube test. The panels, from left to right and from top to bottom, show: density, pressure, Lorentz factor, x-velocity, y-velocity and perpendicular magnetic field B_y.

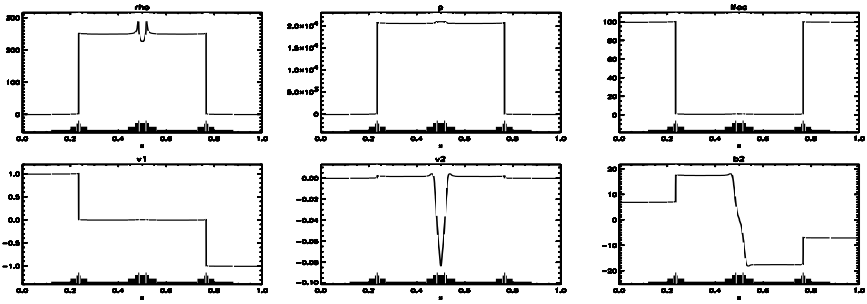


Fig. 3. Flow collision test; ultra-relativistic flows ($\Gamma = 100$) collide at $x = 0.5$. Panels are the same as Fig. 2.

$p = 0.01$ is injected with $v_z = 0.99$ ($\Gamma \approx 7$) into a homogeneous background with $\rho = 10.0$ and $p = 0.01$. The domain of the calculation is $0 < r < 8, 0 < z < 20$ with a base grid of 80×120. Three additional grid levels were used with a refinement factor of two, giving an effective resolution of 640×1600.

Finally the relativistic extension of the 2D MHD rotor test from [6] is shown in Fig. 5. An initially rotating disk of dense plasma is embedded in a homogeneous magnetic field $\vec{B} = 1.0\vec{e}_x$ ($(\rho, v_x, v_y, p) = (10.0, -\alpha y, \alpha x, 1.0)$ for $(x^2 + y^2)^{1/2} < 0.1$, where $\alpha = 9.95$; the background is $(\rho, v_x, v_y, p) = (1.0, 0, 0, 1.0)$). The maximum Lorentz factor is about 10. The disk is decelerated by the winding of the magnetic field lines, and the plasma is flung outward in and oblong shell. Strong shocks and torsional Alfvén waves are generated. The results of this complex 2D RMHD problem agree quite well with the results published in [6].

5 Conclusions

The multi-dimensional grid adaptive code AMRVAC is extended with relativistic hydrodynamics and relativistic magnetohydrodynamic modules. The code performed

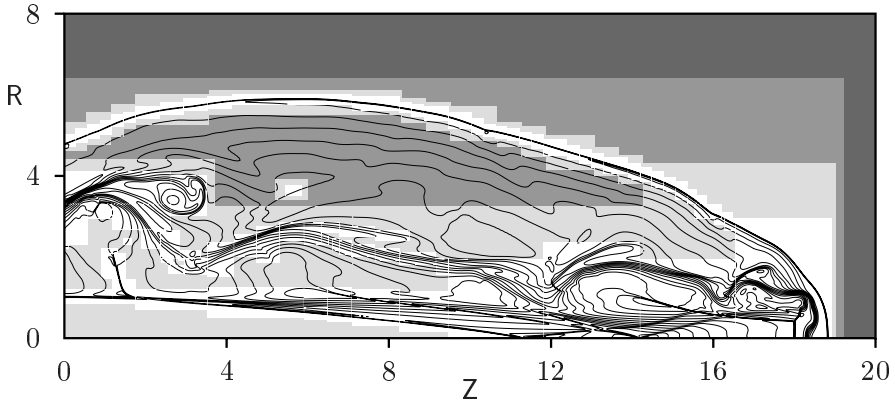


Fig. 4. Axisymmetric RHD jet with $\Gamma \approx 7$. Equidistant contours of $\log \rho$ are shown and the 4-level grid structure is indicated by grayscales.

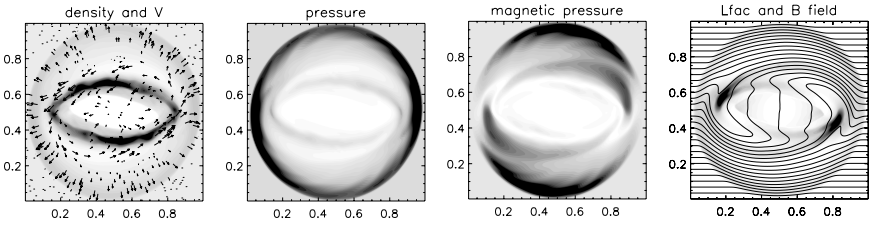


Fig. 5. RMHD rotor simulation at $t = 0.4$. Three grid levels with a refinement factor 2, effective resolution is 400×400 cells.

quite well on all test problems, the AMR algorithm speeds up the calculations presented here so that they could easily be performed on a single desktop processor. Extension toward larger resolutions and higher Lorentz factors is expected to be straightforward as AMRVAC has been shown to perform well on larger parallel computers [12]. To asses the reliability and accuracy of the multidimensional ultra-relativistic calculations it is however desirable to compare results obtained by different R(M)HD solvers in a more quantitative study. Such a validation is essential to get a reliable understanding of the astrophysical objects that we try to model with these computations.

Acknowledgments

This work is part of the Computational Science project “Rapid Changes in Complex Flows” of the Dutch Organization for Scientific Research NWO.

References

1. M. Nool and R. Keppens, Comput. Methods Appl. Math., **2** 92 (2002).
2. R. Keppens, M. Nool, G. Tóth, and J.P. Goedbloed, Comput. Phys. Commun., **153** 317 (2003).
3. L. D. Landau and E. M. Lifshitz: Fluid mechanics (Pergamon Press, 1959).
4. A. V. Koldoba, O. A. Kuznetsov, and G. V. Ustyugova, Mon. Not. Astron. Soc. **333** 932 (2002).
5. S. S. Komissarov, Mon. Not. Astron. Soc. **303** 343 (1999).
6. L. Del Zanna, N. Bucciantini, and P. Londrillo, A&A **400** 397 (2003).
7. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery: Numerical recipes, 2nd ed. (Cambridge University Press, 1992).
8. G. Tóth, Astrophys. Lett. Comm. **34** 245 (1996).
9. G. Tóth, J. Comput. Phys. **138** 981 (1997).
10. M.J. Berger, SIAM J. Sci. Stat. Comput. **7** 904 (1986).
11. M.J. Berger and I. Rigoutsos, IEEE Transactions on Systems, Man and Cybernetics **21**, 1278 (1991).
12. R. Keppens and G. Tóth, Lect. Notes Comp. Science **2329** 940 (2002).
13. M.J. Berger and P. Colella, J. Comput. Phys. **82** 64 (1989).
14. J. Bell, M. Berger, J. Saltzman and M. Welcome, SIAM J. Sci. Comp. **15** 127 (1994).
15. A.L. Garcia, J.B. Bell, W.Y. Crutchfield and B.J. Alder, J. Comput. Phys. **154** 134 (1999).

AMR applied to non-linear Elastodynamics

S. A. E. G. Falle

Department of Applied Mathematics, University of Leeds, Leeds LS2 9JT, UK.
sam@amsta.leeds.ac.uk

Summary. We describe an AMR scheme for non-linear elastodynamics in Lagrangean coordinates. The scheme uses a linear Riemann solver and computes the deformation gradient from the displacements in order to ensure that it is consistent. Solid bodies with stress free boundaries are modeled by embedding them in a very weak material with a smooth transition in material properties at the boundary. A full approximation multigrid is used to compute states in dynamical equilibrium.

1 Introduction

Since the equations governing the dynamics of hyperelastic materials are a set of non-linear hyperbolic conservation laws (see e.g. [MC01]), the most appropriate way to solve these equations numerically is to use a characteristic based conservative scheme. However, it is generally believed that the complexity of associated Riemann problem makes the computational cost of such schemes prohibitive. Here we show that this difficulty is more apparent than real and that it is possible to devise a method that is almost as fast as the conventional finite element schemes. This has several advantages: it is much more robust; it has excellent shock capturing properties; a characteristic based algorithm is essential for an adaptive scheme in order to avoid the generation of errors at refinement boundaries. As one would expect, the scheme is excellent at propagating the waves that occur in dynamic problems and, when combined with a multigrid, it is also very effective at finding equilibrium solutions.

Using such a scheme, it is possible to study a number of interesting properties of such systems, such as the existence, or otherwise, of non-evolutionary shocks and the appearance of stress singularities at stress-free boundaries. The latter are particularly important because they cause considerable difficulties for conventional finite element methods.

It is well known that, in linear elasticity, a concave corner in an elastic body with a stress free boundary leads to a singularity, but that such a corner cannot appear if it is not present in the initial configuration. In the non-linear case a body with an initially smooth boundary can develop corners if the deformation is sufficiently

large, but, since the location of this singularity is not necessarily known a priori, a dynamically adaptive grid is required to ensure high resolution in the neighbourhood of the singularity. However, the mere use of an adaptive grid is not enough unless something is also done about the singularity.

An effective way of dealing with this is to model a stress free boundary by immersing the elastic body in a very weak material with a smooth transition in properties between the two materials. This removes the singularity, but its nature can be studied by varying the thickness of the transition region. This technique has the additional advantage that it provides a simple way of representing complex body shapes on a Cartesian grid. For this an adaptive grid is essential, since an accurate representation of the body would be prohibitively expensive with a uniform grid.

These ideas will be illustrated by an example in which the material law models polyurethane foams.

2 Elastodynamics

As for fluid dynamics, it is possible to write the equations of elastodynamics in either Eulerian or Lagrangean form. The unlimited distortion that occur in fluid flows make Lagrangian methods unsuitable for multi-dimensional problems, but this is not true for elastodynamics unless the material undergoes a substantial degree of plastic flow. [MC01] used an Eulerian formulation precisely because they were interested in problems with plastic flow, but since we are not, a Lagrangean method is the most appropriate.

Let $\mathbf{x} = \mathbf{x}(\mathbf{X}, t)$, where $\mathbf{x}$ is the current (Eulerian) position of a particle with initial (Lagrangean) position $\mathbf{X}$. Define the deformation gradient, A, by

$$A_{ij} = \frac{\partial x_i}{\partial X_j}.$$

2.1 Kinematic Equations

Since we have

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{v}, \tag{1}$$

the deformation gradient evolves according to

$$\frac{\partial A_{ij}}{\partial t} = \frac{\partial v_i}{\partial X_j}. \tag{2}$$

The stretches, λ_i ($i = 1 \cdots 3$), are the square roots of the eigenvalues of $\mathbf{A}^t\mathbf{A}$, and the Jacobian, J, is then given by

$$J = |\mathbf{A}| = \lambda_1\lambda_2\lambda_3.$$

2.2 Equation of Motion

In Cartesian coordinates the equation of motion is simply

$$\frac{\partial \rho v_i}{\partial t} = \frac{\partial S_{ji}}{\partial X_j} + f_i \tag{3}$$

where $\mathbf{S}$ is the nominal stress tensor and f_i is ith component of the body force.

A hyperelastic material possesses a strain energy, U, which is a function of the deformation gradient, $\mathbf{A}$. The simplest way to determine the relation between $\mathbf{S}$ and the strain energy is to use Hamilton's variational principle with the body force set to zero

$$\delta \int_{t_1}^{t2} \int_V (\frac{1}{2}\rho v^2 - U) dV dt = 0. \tag{4}$$

The corresponding Euler equations are

$$\frac{\partial \rho v_i}{\partial t} = \frac{\partial}{\partial X_j} \frac{\partial U}{\partial A_{ij}}.$$

Comparing this with (3) gives

$$S_{ij} = \frac{\partial U}{\partial A_{ji}} \tag{5}$$

2.3 Strain energy

The behaviour of the material is determined by the form of the elastic strain energy, U, which is a function of $\mathbf{A}^T\mathbf{A}$ since it must be invariant under rigid body rotations. It can therefore be written as a function of the stretches, $\lambda_1, \lambda_2, \lambda_3$ defined in section 2.1, or as a function of the invariants I_1, I_2, I_3 where

$$\lambda^3 - I_1\lambda^2 + I_2\lambda - I_3 = 0$$

is the characteristic polynomial of $\mathbf{A}^T\mathbf{A}$. Note that $I_3 = J^2$.

For rubberlike materials, such as polyurethane, it is common practice to use an expression of the form

$$U = \sum_{i=1}^{N} \frac{2\mu_i}{\alpha_i^2} \left(\lambda_1^{\alpha_i} + \lambda_2^{\alpha_i} + \lambda_3^{\alpha_i} + \frac{1}{\beta_i J^{\alpha_i \beta_i}} \right) \tag{6}$$

in which μ_i, α_i and β_i are determined empirically (e.g. [OG84]). Although it is possible to determine similar expressions in terms of I_1, I_2 and I_3, this is less common. This unfortunate since it is much simpler and computationally cheaper to work with the invariants.

2.4 Cauchy Stress Tensor

The nominal stress tensor describes the stress in the Lagrangean frame, but it is more useful to display the stresses in an Eulerian frame as given by the Cauchy stress tensor, **T**. This is related to the nominal stress tensor by

$$\mathbf{T} = \frac{1}{J}\mathbf{A}\mathbf{S}. \tag{7}$$

2.5 Conservation Form

The kinematic equations (2) together with the equations of motion (3) constitute a set of non-linear, hyperbolic conservation laws, which we can write in the form

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial X} + \frac{\partial \mathbf{G}}{\partial Y} + \frac{\partial \mathbf{H}}{\partial Z} = \mathbf{s}, \tag{8}$$

in Cartesian coordinates. Here **Q** is a vector of conserved quantities, Q_i, given by

$$\begin{aligned} Q_i &= \rho v_i \;\; i = 1 \cdots 3, \\ Q_i &= A_{j1} \;\; i = 4 \cdots 6, \;\; j = i - 3, \\ Q_i &= A_{j2} \;\; i = 7 \cdots 9, \;\; j = i - 6, \\ Q_i &= A_{j3} \;\; i = 10 \cdots 12, \;\; j = i - 9. \end{aligned} \tag{9}$$

F, **G**, **H** are the corresponding fluxes and **s** represents the body forces. The fluxes are

$$\begin{aligned} F_i &= -S_{1i} \;\; i = 1 \cdots 3, \\ F_i &= -v_j \;\; i = 4 \cdots 6, \;\; j = i - 3, \\ F_i &= 0 \qquad i = 7 \cdots 12, \end{aligned}$$

$$\begin{aligned} G_i &= -S_{2i} \;\; i = 1 \cdots 3, \\ G_i &= 0 \qquad i = 4 \cdots 6, \\ G_i &= -v_j \;\; i = 7 \cdots 9, \;\; j = i - 6 \\ G_i &= 0 \qquad i = 10 \cdots 12, \end{aligned}$$

$$\begin{aligned} H_i &= -S_{3i} \;\; i = 1 \cdots 3, \\ H_i &= 0 \qquad i = 4 \cdots 6, \\ H_i &= 0 \qquad i = 7 \cdots 9, \\ H_i &= -v_j \;\; i = 10 \cdots 12, \;\; j = i - 9. \end{aligned}$$

The source term is

$$\begin{aligned} s_i &= f_i \;\; i = 1 \cdots 3, \\ s_i &= 0 \;\; i = 4 \cdots 12. \end{aligned}$$

3 Numerical Scheme

It has long been known (see e.g. [RO86]) that the most effective numerical schemes for equations of this type are conservative finite volume schemes that utilise the solution to one dimensional Riemann problems to determine the fluxes. Not only are such schemes essential for problems involving the propagation of large amplitude waves, their upwind nature ensures that no errors are generated at the boundaries between coarse and fine grids that arise with an adaptive grid methods such as that described in section 3.6. Furthermore, as we shall see in section 3.5, they can be combined with a multigrid to give a very efficient method of obtaining a steady states. They do suffer from the disadvantage that the elastic Riemann problem is quite complicated, but, as we shall see, its computational cost is not excessive.

3.1 Riemann Problem

In order to determine the fluxes for a multidimensional scheme, it is sufficient to solve the one dimensional Riemann problem for each direction. For the X direction, we set the derivatives with respect to Y and Z to zero in equations (8) to obtain

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial X} = \mathbf{s}, \tag{10}$$

where now

$$\begin{aligned} Q_i &= \rho v_i \ i = 1 \cdots 3, \\ Q_i &= A_{i1} \ i = 4 \cdots 6. \end{aligned}$$

Note that although all components of $\mathbf{A}$ play a role, only the first column can vary for a one dimensional problem in the X_1 direction.

A Riemann problem for such a system is one for which the source terms are zero and the initial data consists of a single discontinuity at $X = 0$, i.e.

$$\begin{aligned} \mathbf{Q} &= \text{constant} = \mathbf{Q}_L \ X \leq 0, \\ \mathbf{Q} &= \text{constant} = \mathbf{Q}_R \ X > 0. \end{aligned} \tag{11}$$

As is well known (see e.g. [RO86]), the solution to the Riemann problem consists of the 6 elementary waves of the system, separated by constant regions. Although it is possible to determine the exact solution, this is not necessary for a numerical scheme. For this particular system we find that a linear solution suffices even for very strong waves.

With the source term set to zero, the linear version of (10) is

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{B}\frac{\partial \mathbf{Q}}{\partial X} = 0, \tag{12}$$

where

$$B_{ij} = \frac{\partial F_i}{\partial Q_j} \quad \text{evaluated at} \quad \mathbf{Q}_m = \frac{1}{2}(\mathbf{Q}_L + \mathbf{Q}_R).$$

B is of the form

$$\mathbf{B} = \begin{pmatrix} 0 & -\mathbf{C} \\ -\mathbf{I}/\rho & 0 \end{pmatrix},$$

where **I** is the 3×3 identity matrix and

$$C_{ij} = \frac{\partial U}{\partial A_{i1} \partial A_{j1}}. \tag{13}$$

Let s_i, $\mathbf{l}_i$, $\mathbf{r}_i$ be the eigenvalues and left, right eigenvectors of **B**. We then have

$$\mathbf{r}_i = (\mathbf{r}_{pi}, \mathbf{r}_{Ai})^t, \qquad \mathbf{l}_i = (\mathbf{l}_{pi}, \mathbf{l}_{Ai}),$$

where

$$\begin{aligned} \mathbf{r}_{pi} &= -\rho s_i \mathbf{r}_{Ai}, & \mathbf{C}\mathbf{r}_{Ai} &= -s_i \mathbf{r}_{pi}, \\ \mathbf{l}_{pi}\mathbf{C} &= -s_i \mathbf{l}_{Ai}, & \mathbf{l}_{Ai} &= -\rho s_i \mathbf{l}_{pi}. \end{aligned}$$

This gives

$$\mathbf{C}\mathbf{r}_{Ai} = \rho s_i^2 \mathbf{r}_{Ai}, \quad \mathbf{l}_{pi}\mathbf{C} = \rho s_i^2 \mathbf{l}_{pi}.$$

The wavespeeds, s_i, of this system are given by the eigenvalues of the matrix, $\mathbf{C}/\rho$, which is called the acoustic tensor in the X_1 direction. From (13) it is clear that it is symmetric, so that the eigenvectors $\mathbf{r}_{Ai}$ are orthogonal and $\mathbf{l}_{Ai} \propto \mathbf{r}_{Ai}^t$. Furthermore, it is also positive definite provided that U is a convex function. There are therefore 6 real wavespeeds of the form $s_i = \pm c_j$, where c_j^2 $(j = 1 \cdots 3)$ are the eiqenvalues of the acoustic tensor. These correspond to fast, slow and rotational acoustic waves.

The normalised eigenvectors of **B** are

$$\mathbf{r}_i = \left[-\rho s_i \mathbf{r}_{Ai}, \mathbf{r}_{Ai}\right]^t, \quad \mathbf{l}_i = \frac{1}{2}\left[-\frac{1}{\rho s_i}\mathbf{r}_{Ai}, \mathbf{r}_{Ai}\right],$$

where $\mathbf{r}_{Ai}$ are the normalised eigenvectors of **C**. The solution to equations (12) with the initial data (11) has a constant state, $\mathbf{Q}_*$, at $X = 0$ given by

$$\mathbf{Q}_* = \mathbf{Q}_L + \sum_{s_i<0} \mathbf{l}_i \cdot (\mathbf{Q}_R - \mathbf{Q}_L) = \mathbf{Q}_* = \mathbf{Q}_R + \sum_{s_i>0} \mathbf{l}_i \cdot (\mathbf{Q}_L - \mathbf{Q}_R). \tag{14}$$

Although these eigenvectors and eigenvalues have to be calculated numerically, this requires only the computation of the eigenvalues and eigenvectors of the symmetric matrix, **C**, which is not particularly expensive.

3.2 Finite Volume Numerical Scheme

Had we determined the exact state at the interface, $\mathbf{Q}_*$, then the corresponding fluxes, $\mathbf{F}_*$ could have been obtained from

$$\mathbf{F}_* = \mathbf{F}(\mathbf{Q}_*).$$

However, since the $\mathbf{Q}_*$ given by (14) is a solution to the linearised equations (12), it is much better to calculate fluxes that are consistent with these equations. In the linearised equations, the fluxes at the interface are given by

$$\mathbf{F}_* = \frac{1}{2}[\mathbf{F}_L + \mathbf{F}_R - \sum_i |s_i| \mathbf{l}_i \cdot (\mathbf{Q}_R - \mathbf{Q}_L)\mathbf{r}_i]$$

The fluxes at the Y, Z interfaces are calculated in a similar manner and these are then used to construct a second order conservative scheme of the form

$$\begin{aligned} \frac{1}{\Delta t}(\mathbf{Q}^{n+1}_{ijk} - \mathbf{Q}^n_{ijk}) = & \frac{1}{\Delta X}(\mathbf{F}^{n+1/2}_{i-1/2jk} - \mathbf{F}^{n+1/2}_{i+1/2jk}) + \\ & \frac{1}{\Delta Y}(\mathbf{G}^{n+1/2}_{ij-1/2k} - \mathbf{G}^{n+1/2}_{ij+1/2k}) + \\ & \frac{1}{\Delta Z}(\mathbf{H}^{n+1/2}_{ijk-1/2} - \mathbf{H}^{n+1/2}_{ijk+1/2}) + \mathbf{s}^{n+1/2}_{ijk}. \end{aligned} \tag{15}$$

Here $\mathbf{Q}^n_{ijk}$ is the numerical solution in the ijk cell at timestep n and $\mathbf{F}^{n+1/2}_{i+1/2jk}$ etc are the interface fluxes calculated from the Riemann problems at the half-time, $t = t + \Delta t/2$. These are obtained by first calculating the solution at the half-time using a first order scheme and then using averaged gradients to determine the states on the left and right of the interface [FA91]. Note that, since this is an explicit scheme, the timestep is limited by the Courant condition

$$\Delta t \leq \frac{\Delta x}{s_m}$$

where s_m is the maximum wavespeed in the computational domain.

3.3 Consistency of deformation gradient

If (15) is used to calculate the components of the deformation gradient, then there is no guarantee that these are consistent with the displacement field. Even worse, the accumulation of numerical errors means that the body will not, in general, return to the undeformed state once the external stresses are removed.

It is therefore better to use (15) for the velocities only and to then use these to calculate the displacements from a numerical approximation to (1)

$$\frac{1}{\Delta t}(\mathbf{x}^{n+1}_{ijk} - \mathbf{x}^n_{ijk}) = \mathbf{v}^{n+1/2}_{ijk}$$

The deformation gradients at the new time are then obtained from the $\mathbf{x}^n_{ijk}$ via a central difference. This does not destroy conservation since it amounts to using the mean velocities at the interface to compute the fluxes of the deformation gradient. It does, however, mean that the scheme is not strictly upwind, but this seems to have no adverse effect on either wave propagation or the performance of the adaptive grid.

3.4 Boundaries

In problems involving elastic bodies there are generally three types of boundary, rigid walls, free surfaces and regions of self-contact. Each of these can be dealt with automatically by appropriate modifications to the equations.

Free Boundaries and Self-contact

We introduce a scalar, α, that is unity in the material and zero in the surrounding medium (usually air). The elastic energy then takes the form

$$U = \alpha U_m + (1 - \alpha) U_s,$$

where U_m is the elastic energy function for the material and U_s is chosen so as to model the surroundings. It would be nice if U_s could correspond to a fluid, but this is impossible since there would then be no restriction on the strain. However, the surrounding material can be made sufficiently weak for this to give a very good approximation to a free surface. U_s can also be chosen so as to approximate an appropriate external pressure. This also automatically deals with self contact, since as the two surfaces approach each other, the stress will tend towards that in the elastic material.

Although the real body may have a sharp boundary, there are several reasons why the scalar cannot be a discontinuous function of position. Since we are working with a Cartesian grid, this would mean that the surface of the body consists of sharp steps, which would not only be unrealistic, but would also lead to numerical difficulties. There is, however, a much more important reason for imposing a smooth transition between the material and the surroundings. The equilibrium equations are a set of non-linear elliptic equations and such equations, like their hyperbolic counterparts, can have singularities even if the undeformed body has a smooth boundary. Indeed, it seems that a sufficiently large compression generates a singularity for any body for which the undeformed boundary is not convex. This is presumably why finite element codes that attempt to impose a sharp boundary fail for large deformations whenever the numerical resolution is sufficiently high.

Fortunately, there is a simple solution to this difficulty, which is to smooth out the discontinuous boundary by applying a diffusion operator to the scalar. Experience shows that there are no numerical difficulties provided the boundary is diffused over about 5 to 10 grid points, which requires about 20 to 40 iterations of a diffusion operator. As we shall see, it is possible to study the form of such singularities by carrying out a sequence of calculations with different numerical resolution.

It might seem that one cannot impose accurate boundary conditions with this technique, but since the width of the boundary scales with the mesh spacing, we can make the boundary as sharp as we please simply by increasing the numerical resolution. The adaptive grid described in 3.6 makes it possible to achieve this at minimal cost.

Rigid Bodies

We now define a scalar, α, which is zero in the material and unity inside the rigid body. In order to impose a given displacement, $\mathbf{x}_b$, on the rigid body, we add a force

$$\mathbf{f} = -\alpha \left[\mathbf{c} - K^2 \rho (\mathbf{x}_b - \mathbf{x}) + 2K \frac{d\mathbf{x}}{dt} \right] \tag{16}$$

where K is a positive coefficient,

$$c_i = \frac{\partial \mathbf{S}_{ji}}{\partial X_j}$$

is the force due to the stresses and the second term on the RHS is a drag that ensures critical damping. This has the effect of forcing $\mathbf{x} = \mathbf{x}_b$ inside the body.

3.5 Steady States

A time dependent code can obviously be used to obtain steady states by simply imposing the appropriate boundary and initial conditions and then marching forward in time until the solution becomes steady. However, a hyperelastic material is a Hamiltonian system and will therefore not reach a steady state unless there is some dissipative mechanism. In the real system this is provided by viscoelastic and other irreversible effects, but for numerical purposes it is much better to introduce an artificial dissipative process.

The easiest way to do this is to add a drag force of the form

$$\mathbf{f}_d = -D\rho\mathbf{v} \tag{17}$$

where D is a drag coefficient. Ideally, D should to be such as to make the system critically damped, but it is not, in general, possible to do this exactly. However, a suitable value of D is given by

$$D = 2\omega, \tag{18}$$

where $\omega/2\pi$ is the smallest frequency of vibration of the undamped system. ω can be estimated from

$$\omega \simeq 2\pi s/L, \tag{19}$$

where s is a mean sound speed and L is the largest dimension of the system.

If there are considerable differences between the sound speeds in different parts of the system, then it is better use a drag coefficient determined from the local sound speed

$$D(\mathbf{X}) = \frac{4\pi s(\mathbf{X})}{L}.$$

Matrix Timestep

If the sound speed is approximately the same everywhere, then time to reach the steady state should be of order L/s. However, suppose there are two regions, R_1, R_2, with sizes L_1, L_2 and sound speeds, s_1, s_2. If $L_1/s_1 \gg L_2/s_2$ because $s_1 \ll s_2$, region R_2 will relax much more rapidly than R_1. The time to reach the steady state will then be of order L_1/s_1. For an explicit scheme, the timestep scales like $1/s_2$, which means that cost of reaching the steady state is increased by a factor s_2/s_1.

This inefficiency can be removed using a matrix timestep in which the solution in each cell is advanced with a timestep determined by the sound speed in that cell, i.e the solution at $n+1$ is obtained from

$$\mathbf{Q}^{n+1}_{ijk} = \mathbf{Q}^{n}_{ijk} + \Delta t^{n}_{ijk} \frac{d\mathbf{Q}^{n}_{ijk}}{dt}$$

where $d\mathbf{Q}^{n}_{ijk}/dt$ is calculated from (15) and Δt^{n}_{ijk} is the maximum stable timestep for cell ijk.

Multigrid

Suppose that we have a uniform grid with mesh spacing ΔX covering a domain of size L in which the sound speed is of order s. The timestep is then proportional to $\Delta X/s$ and the time to reach the steady state is proportional to L/s. The number of timesteps therefore increases like $1/\Delta X$. Since the computational cost per timestep is proportional to $1/\Delta X^d$, where d is the number of dimensions, the cost of the calculation grows like $1/\Delta X^{d+1}$.

It is clear from this that such calculations will be very expensive if there are regions in which the solution varies so rapidly that an accurate solution requires a small ΔX. If these regions only occupy a small fraction of the domain, then the adaptive grid described in the next section can reduce the cost considerably. However, if we are merely interested in reaching the steady state, then a multigrid can be used to improve the efficiency, irrespective of any gains to be had from an adaptive grid (see e.g. [BR82]).

A multigrid exploits the fact that, for a hyperbolic system, the explicit timestep on a grid with mesh spacing ΔX is of the same order as the frequency of waves with wavelength ΔX. The components of the residual with wavelength of order ΔX will therefore be destroyed after a few timesteps, provided that they are significantly damped. The residual then consists of waves whose wavelength is significantly larger than ΔX. Since the decay time of these waves is much longer than the timestep on this grid, it is more efficient to destroy them by evolving the solution on a coarser grid. This procedure is repeated recursively until the coarsest possible grid is reached, whereupon the change in the solution is mapped down to the finest grid. This constitutes one multigrid cycle. Because the interpolation from a coarse to fine grid introduces components in the residual whose wavelength is comparable to the mesh spacing on the fine grid, the effect of a multigrid cycle is to destroy the longest wavelengths that are present in the residual. Successive multigrid cycles remove shorter and shorter wavelengths until the solution converges to the steady state.

In order to implement this algorithm, we set up N grids, $G_0 \cdots G_N$, with mesh spacings $\Delta X_n = \Delta X_0/2^n$. Now write the discrete equations on G_n as

$$\frac{\mathbf{Q}_n^{m+1} - \mathbf{Q}_n^m}{t_{m+1} - t_m} + L_n(\mathbf{Q}_n^m) = \mathbf{S}_n$$

where

$\mathbf{Q}_n^m$ solution on G_n after m relaxations
L_n relaxation operator on grid G_n
I_n^{n-1} fine to coarse operator from G_n to G_{n-1}
I_{n-1}^n coarse to fine operator from G_{n-1} to G_n
$\mathbf{S}_n$ multigrid source term on grid G_n.

The multigrid source term is given by

$$\mathbf{S}_n = L_n(I_{n+1}^n \mathbf{Q}_{n+1}) + I_{n+1}^n[\mathbf{S}_{n+1} - L_{n+1}(\mathbf{Q}_{n+1})]$$

with $\mathbf{S}_N = 0$. Since the second term is simply the rate of change of the solution on G_{n+1} projected onto G_n, it vanishes if the solution is converged on G_{n+1}. But then the first term ensures that the solution is also converged on G_n.

After an appropriate number of relaxations on a coarse grid, G_{n-1}, we cannot just project the solution on G_{n-1} onto G_n since this would destroy the small scale information on G_n. Instead the G_n solution is updated by interpolating the difference between the G_{n-1} and G_n solutions according to

$$\mathbf{Q}_n \to \mathbf{Q}_n + I_{n-1}^n(\mathbf{Q}_{n-1} - I_n^{n-1}\mathbf{Q}_n)$$

There remains the question of how relaxations should be distributed among the grids in order to achieve the fastest convergence. The simplest approach is to use a fixed strategy in which the number of relaxations per grid is always the same. However, it is possible to do better by using the information generated by the multigrid process to choose the most appropriate grid for the relaxations. [AF91] showed that a simple way of doing this is to use the operators I_{n-1}^n, I_n^{n-1} to determine the wavelength of the dominant component of the residual and then to select the grid which damps this most effectively.

Clearly if the residual on a grid G_n,

$$\mathbf{R}_n = \mathbf{S}_n - L_n(\mathbf{Q}_n),$$

is dominated by components whose wavelength is much larger than ΔX_n, then it can be accurately represented on G_{n-1}. We should therefore expect the quantity

$$K_n = \frac{||I_{n-1}^n I_n^{n-1}\mathbf{R}_n - \mathbf{R}_n||}{||\mathbf{R}_n||}$$

to be small. Here $\|\cdot\|$ stands for some appropriate norm. [AF91] show that the grid that most effectively damps the dominant mode in the residual is the coarsest grid for which $K_n \leq 1/\sqrt{2}$. A simple, but effective strategy is to relax a few times on this grid, correct to the next finer grid, relax and so on until the finest grid is reached. Since

the longest wavelengths are destroyed first, one would expect the grid selected by this criterion to become finer and finer as the steady state is approached. Eventually it only chooses the finest grid, at which point convergence has been achieved.

3.6 Hierarchical Adaptive Grid

In order to achieve high resolution without incurring excessive computational costs, we use a hierarchical adaptive grid. As for the multigrid described in the previous section, this uses a hierarchy of grids $G_0 \ldots G_N$ such that, if the mesh spacing is ΔX on G_0, then it is $\Delta X/2^n$ on G_n. Grids G_0 and G_1 cover the whole domain, but the finer grids need only exist in regions which require high resolution. The grid hierarchy is used to generate an estimate of the truncation error by comparing solutions on grids with different mesh spacings and the grid refines if this error exceeds a given tolerance. Refinement also occurs in time so that if the time step on G_0 is Δt, then it is $\Delta t/2^n$ on G_n.

Unlike most AMR codes (e.g. [BO84], [BC89], [BBSW94], [QU96]), refinement is on a cell by cell basis instead of being organized into patches. This gives a more efficient grid at the cost of some increase in the cost of integration. It is particularly efficient when the regions requiring high resolution are thin sheets, such as boundary layers and shocks, or, as in this application, the diffused boundary of an elastic body. As we have shown in section 2.5, the equations of elastodynamics are hyperbolic conservation laws and can therefore readily be incorporated into such a code.

4 Examples

4.1 One dimensional Riemann problems

For this problem we use a strain energy of the form (6) with $N = 1$, $\mu_1 = 1$, $\alpha_1 = 12$, $\beta_1 = 0.125$, which models a polyurethane foam. Note that the large value of α_1 means that the stress becomes very large even for moderate deformations.

We start by showing that the numerical algorithm can cope with reasonably strong one dimensional Riemann problems. As we have shown in section 3.1, there are two non-linear waves, fast and slow and a linearly degenerate rotational wave. Figure 1 shows the solution for initial data that generates only fast and slow waves and we can see that the scheme is able to capture very strong shocks even though it uses a linear Riemann solver.

Figure 2 shows the solution for a Riemann problem with a discontinuity in the direction of the transverse shear. This generates two strong rotational waves and weaker waves of the other two families.

We have not compared these numerical solutions with the exact solutions since the latter would be tedious to calculate. However, the numerical scheme evidently converges and the discontinuities are evolutionary and are captured without oscillations or other numerical artifacts. Since this is a conservative scheme, the Lax theorem [LA73] assures us that the scheme does indeed converge to the appropriate weak solution.

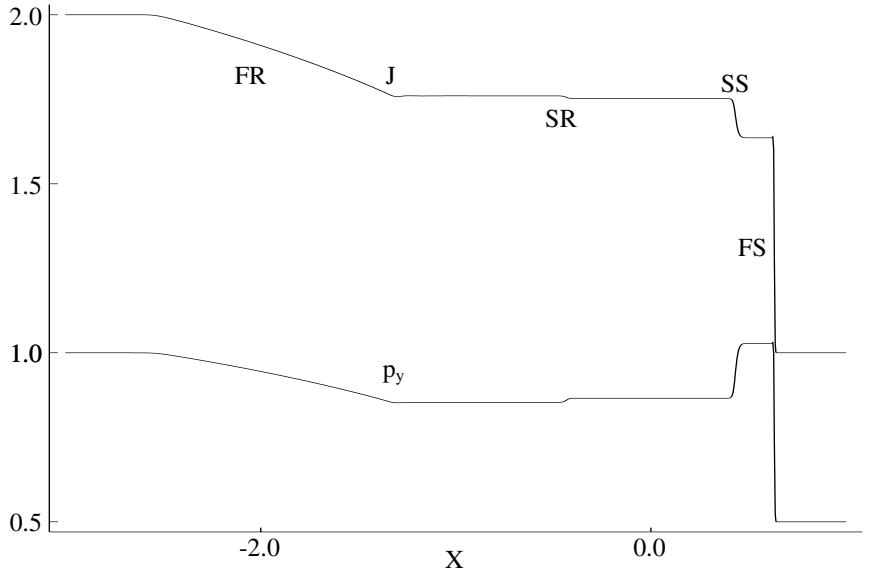


Fig. 1. Jacobian, J and $p_y = A_{21}$ for a Riemann problem with $J = 2.0$, $p_y = 1.0$, $p_z = 0.0$, $v_x = 0.0$, $v_y = 0.1$ in $x \leq 0$ and $J = 1.0$, $p_y = 0.5$, $p_z = 0.0$, $v_x = 0.0$, $v_y = -0.1$ in $x > 0$. FR – fast rarefaction, SR – slow rarefaction, FS – fast shock, SS – slow shock.

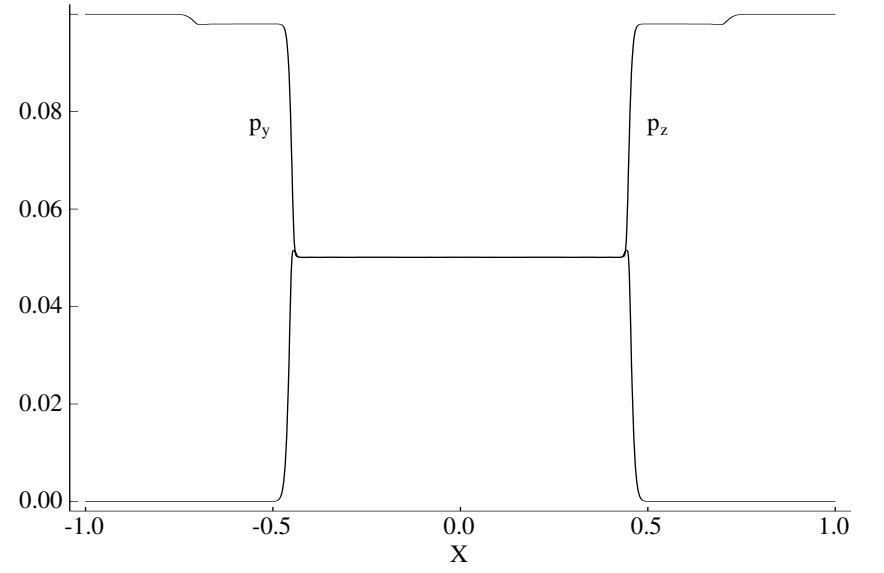


Fig. 2. $p_y = A_{21}$ and $p_z = A_{31}$ for a Riemann problem with $J = 1.0$, $p_y = 0.1$, $p_z = 0.0$, $\mathbf{v} = 0.0$ in $x \leq 0$ and $J = 1.0$, $p_y = 0.0$, $p_z = 0.1$, $\mathbf{v} = 0.0$ in $x > 0$.

4.2 Two dimensional steady problem

In order to illustrate the methods for steady problems described in 3.5 and the utility of the adaptive grid, we consider the deformation of a two dimensional body through a sequence of steady states. The initial shape of the body (figure 3) consists of a square with a circular cut-out. It is defined by a scalar, α, which is 1 in the body and 0 in the surroundings.

The energy function is

$$U = \alpha\mu_m U + (1-\alpha)\mu_s U$$

where $\mu_m = 1$ and $\mu_s = 0.001$ and

$$U = \frac{2}{\alpha_1^2}\left(\lambda_1^{\alpha_1} + \lambda_2^{\alpha_1} + \frac{1}{\beta J^{\alpha_1\beta_1}}\right)$$

with $\alpha_1 = 11.5$, $\beta_1 = 0.125$. In order to make the wave speeds the same in both materials in the initial state, we set the density in the material to 1 and that in the surroundings to 0.001. The small value of μ_s ensures that the stress at the boundary of the body is very small, which gives a good approximation to a stress free boundary. As can be seen in figure 2, the scalar has been diffused by a number of iterations of a diffusion operator to give a smooth transition between the two materials.

Symmetry is imposed at $X = 0, X = 2$ and $Y = 0$ and the boundary at $Y = 1$ is a piston that imposes zero tangential stress and moves vertically downwards in a series of steps. After each change in the position of the piston, the solution is relaxed to a steady state using the multigrid.

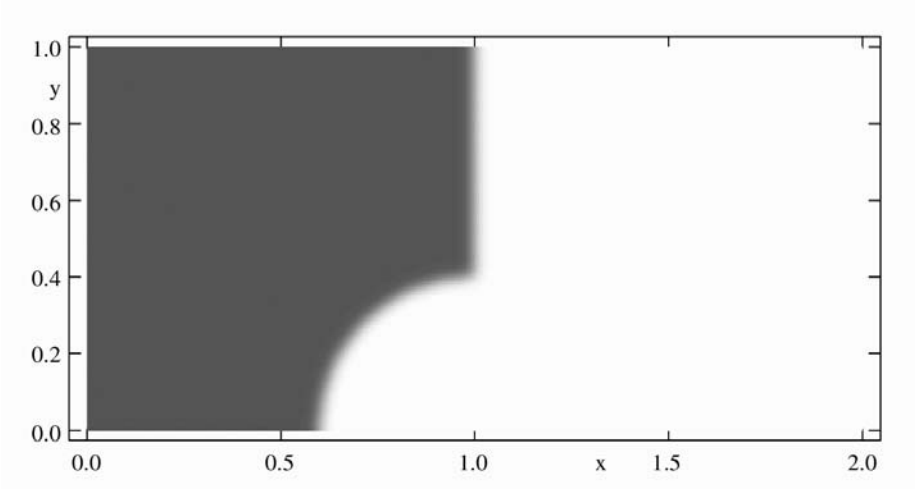


Fig. 3. Scalar describing the initial shape of the body with a diffused bounday.

Figure 4 shows the deformation of the body at various positions of the top boundary. It can be seen that at large deflections the initial circular cut-out closes and the radius of curvature of the boundary becomes very large near the $y = 0$ plane. If the boundary were infinitely sharp, this would be a cusp but the diffused boundary avoids this. There is, nevertheless, a large stress at the point where a cusp would form, as can be seen in figure 5.

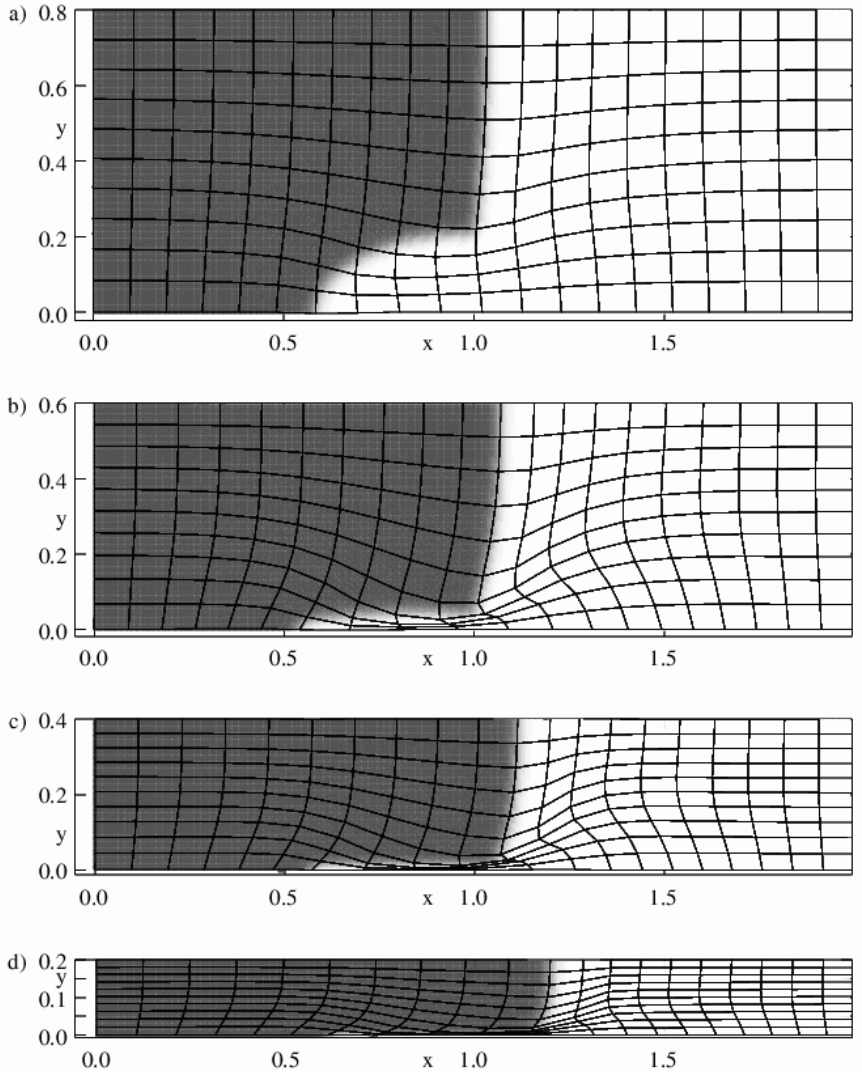


Fig. 4. Scalar and level 0 grid in the Eulerian frame. The deflection of the top boundary is: a) -0.2, b) -0.4, c) -0.6, d) -0.8.

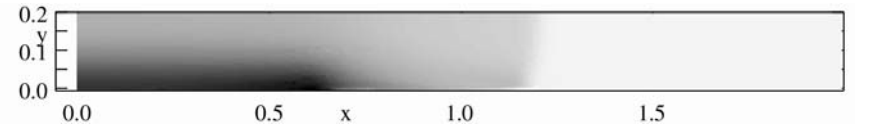


Fig. 5. Hydrostatic pressure at a deflection of the top boundary of -0.8. Linear grey scale with white = -0.284, black = 0.645.

This is an example of the appearance of a sharp corner in an initially smooth body and this would lead to a stress singularity. Of course, this cannot happen in a real system, instead the material must either deform inelastically, or there must be some physical mechanism that limits the stress. In the case of a polyurethane foam, it seems likely that the inhomogeneity due to the presence of bubbles introduces a length scale that prevents the stress from becoming singular. Unfortunately, the proper treatment of this effect requires a non-local material law i.e. one in which the stress is not merely a function of the local strain.

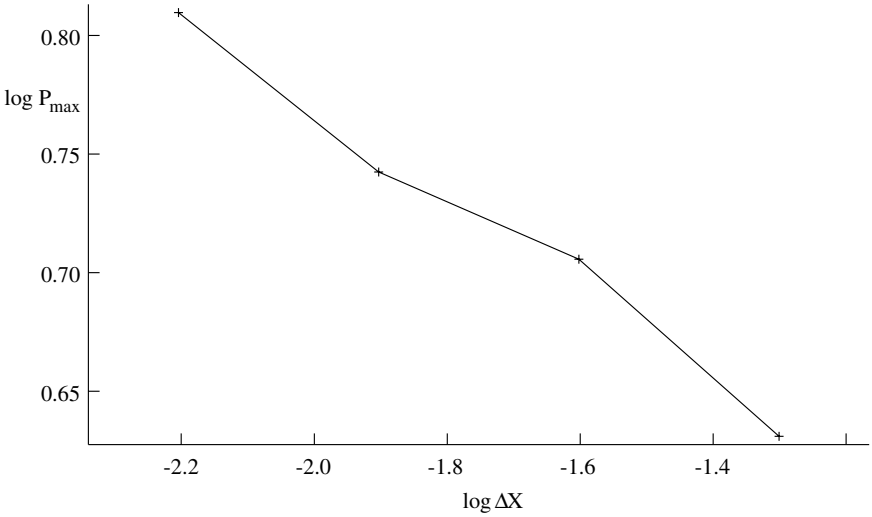


Fig. 6. Maximum hydrostatic pressure as a function of mesh spacing on the finest grid.

We can, however, investigate the nature of the singularity by looking at how the maximum stress behaves as a function of the thickness of the transition region between the body and the surroundings. The simplest way to do this is to compute the solution at a number of different resolutions and to use a fixed number of iterations of the diffusion operator to smooth the boundary. We find that the discretization errors are neglible as long as there are about 10 cells in the transition region, so the differences between these solutions are entirely due to the thickness of the transition region. Figure 6 is a log–log plot of the maximum hydrostatic pressure against the

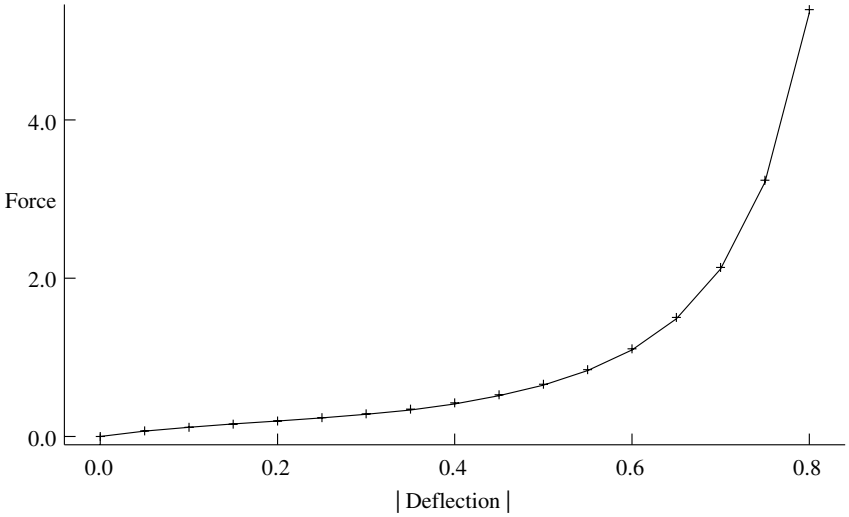


Fig. 7. Force on the $Y = 1$ boundary as a function of the deflection of the boundary. The line is for $\Delta X = 6.25\ 10^{-3}$ on the finest grid and markers are for $\Delta X = 1.25\ 10^{-2}$.

mesh spacing, which shows that the pressure appears to be increasing as an inverse power of the mesh spacing and hence as an inverse power of the thickness of the transition region.

Table 1. Grid filling factors in the undeformed state.

Grid	Size	Filling Factor
G^0	20×10	1.0
G^1	40×20	1.0
G^2	80×40	0.260
G^3	160×80	0.166
G^4	320×160	0.07

Table 2. Grid filling factors at a deflection of -0.8.

Grid	Size	Filling Factor
G^0	20×10	1.0
G^1	40×20	1.0
G^2	80×40	0.498
G^3	160×80	0.290
G^4	320×160	0.135

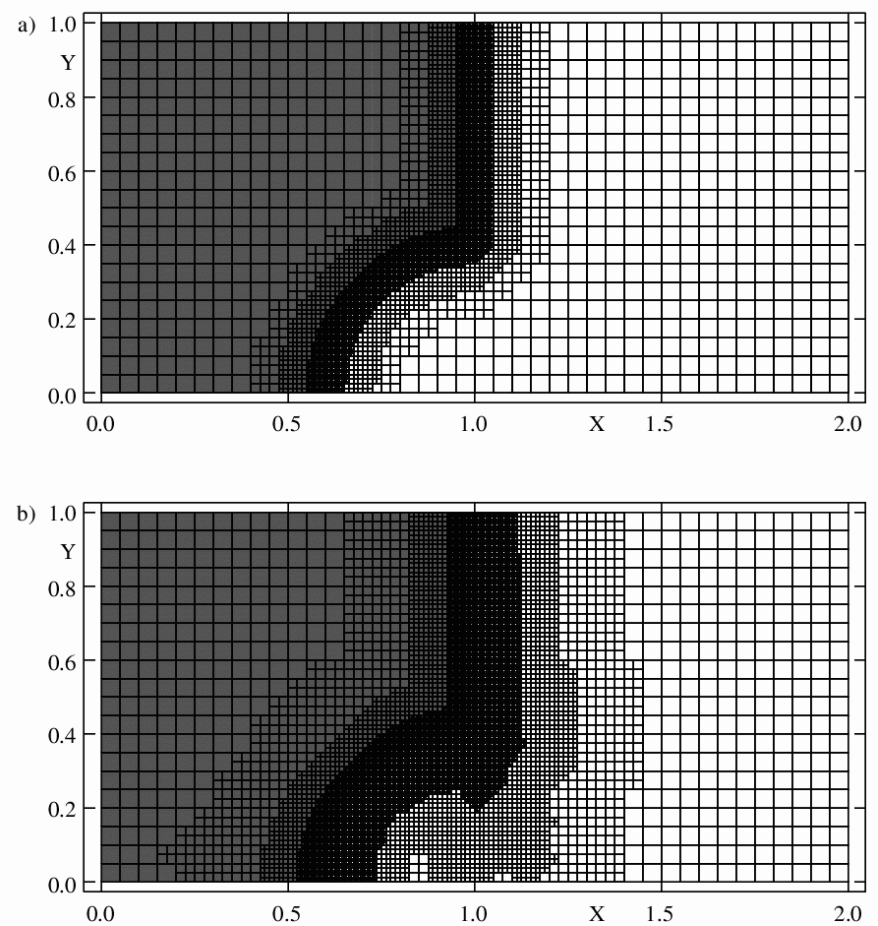


Fig. 8. Grids $G^1 \cdots G^4$. a) Initial state, b) at a deflection of -0.8.

Although the maximum stress increases without limit as the resolution is increased, the force on the $Y = 1$ boundary converges as can be seen in figure 7. This also illustrates the strongly non-linear nature of the material.

Finally, it is worth looking at filling factors on the various grid levels. From table 1, it is clear that, in the initial state, the filling factor reduces by a roughly a factor of 2, which is simply because, as can be seen in 8a, the grid is only refining where the scalar varies. The filling factors behave in the same way in the final state (table 2), even though they are somewhat larger because the grid now refines in regions, such as the slot, in which the distortion is large (8b).

5 Conclusions

We have shown that it is possible to extend upwind methods and adaptive grids to non-linear Lagrangean elastodynamics and that they are just as effective in this case as they are for fluid dynamics. In order to do this, we have found simple, but effective, solutions to the two main difficulties: the complexity of the Riemann problem and the consistency of the deformation gradient. Furthermore, the combination of an adaptive grid and a variable material strength makes it possible to compute equilibrium states of complex elastic bodies without encountering the problems associated with singularities that afflict conventional finite element methods.

References

AF91. Arthur, S.J., Falle, S.A.E.G.: Multigrid methods applied to an explosion at a plane density interface. MNRAS, **251**, 93–104 (1991)

BBSW94. Bell, J., Berger, M., Saltzmann, J., Welcome, M.: Three dimensional dadative mesh refinement for hyperbolic conservation laws. Siam J. Sci. Comput., **15**, 127–138 (1994)

BO84. Berger, M.J., Oliger J.:Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys., **53**, 484–512 (1984)

BC89. Berger, M.J., Colella, P.: Local adaptive mesh refinement for shock hydrodynamics. J. Comput. Phys., **82**, 64–84 (1989)

BR82. Brandt, A.: Guide to multigrid development. Lect. Notes Math., **960**, 220–312 (1982)

FA91. Falle, S.A.E.G.: Self-similar jets. MNRAS, **250**, 581–596 (1991)

LA73. Lax, P.D.: Hyperbolic systems and the mathematical theory of shock waves. Regional conference Series in Applied Mathematics: **11**, Philadelphia, Society for Industrial and Applied Mathematics (1973)

MC01. Miller, G.H., Colella, P.: A high order Eulerian Godunov method for elastic-plastic flow in solids. J. Comput. Phys., **167**, 131–176 (2001)

OG84. Ogden , R.W.: Non-linear elastic deformations. Ellis Horwood (1984)

QU96. Quirk, J.J.: A parallel adative grid algorithm for computational shock hydrodynamics. Appl. Numer. Math., **20**, 427–453 (1996)

RO86. Roe, P.L.: Characteristic–based schemes for the Euler equations. Ann. Rev. Fluid Mech., **18**, 337–365 (1986)

A Parallel AMR Implementation of The Discrete Ordinates Method for Radiation Transport

Louis H. Howell

Lawrence Livermore National Laboratory, Livermore, CA 94551
nazgul@llnl.gov

Summary. Requirements for efficient parallel numerical computations fall into two broad categories: scalable algorithms and scalable implementations. For the discrete ordinates method the basic algorithmic building block is the transport sweep, and some form of convergence acceleration is required in order to combine sweeps into an efficient solver. This paper will focus on the implementation side of the issue. I will present a method for performing parallel transport sweeps in the context of a code for time-dependent radiation hydrodynamics using block-structured adaptive mesh refinement. Sweep patterns involving a single refinement level as well as sweeps coupling multiple levels will both be discussed. 2D and 3D numerical results will compare the parallel performance of these schemes with the multigrid solvers called by a radiation diffusion package implemented in the same code.

1 Introduction

This paper will present some details of parallel implementation and performance of a radiation transport code that is still under development. The best background for this work is [HG03], in which I describe the algorithm used in the same code for single-group radiation diffusion, using time-dependent, block-structured adaptive mesh refinement (AMR) and coupling to a Godunov scheme for multi-fluid Eulerian gasdynamics. The present work develops a discrete ordinates transport algorithm as an alternative to radiation diffusion, but many aspects of the diffusion model—in particular, the nonlinear coupling between the radiation field and the fluid energy, and the role of the radiation solver in the AMR timestepping scheme—apply in very similar ways to the discrete ordinates model. The block structured AMR algorithm itself has its roots in the schemes developed for hyperbolic systems by Berger and Oliger [BO84] and extended by Berger and Colella [BC89] and by Bell et al. [BBSW94]. Work on a previous, serial AMR implementation of discrete ordinates for combustion applications appeared in [HPCJF99]. More extensive information on the genealogy of these algorithms can be found in several of these references.

Solution of the equations for the discrete ordinate discretization of the radiative transport equation is an iterative process. The steps of this iteration are built up out of operations called transport sweeps, often combined with some form of convergence

acceleration. In an earlier paper [How02] I discussed the form of the equations to be solved for radiation transport, the parallel iterative scheme for AMR calculations, and conjugate gradient acceleration using two different preconditioners. The first two of these topics will be summarized as background in the following sections, but the acceleration scheme will not be discussed. (The main results shown in [How02] were that an adaptive mesh calculation could reproduce the accuracy of a similar uniform grid calculation at much lower cost, and that conjugate gradient acceleration could greatly reduce the number of transport sweep iterations required for convergence. See [RAN97] for additional discussion of acceleration techniques.)

Instead, this paper will focus on the parallel scaling properties of the sweep implementation, both in single- and multiple-level problems and in 2 and 3 spatial dimensions. Other developments since [How02] include support for 2D axisymmetric (RZ) coordinates and 3D adaptive mesh refinement.

2 Derivation of Equations

This section and the next will present bare-bones descriptions of the equations for radiation transport and the issues arising from the AMR grid structure, respectively. More complete treatments are found in the references.

The single-level timestepping scheme operates in a split fashion, starting with advection and conduction and followed by the radiation update. Let $(\rho E)^-$ represent the new time $(n+1)$ fluid energy just prior to the radiation step:

$$(\rho E)^- = (\rho E)^n + \Delta t \left\{ -[\nabla \cdot (u\rho E + up)]^{n+1/2} + \frac{1}{2}\nabla \cdot \left(\kappa_0 T^{5/2} \nabla T\right)^{n+1/2} \right\}. \quad (1)$$

Radiation transport then uses a gray, fully-implicit discretization coupled to the fluid energy equation:

$$\frac{I^{n+1} - I^n}{c\Delta t} + (\Omega \cdot \nabla) I^{n+1} + \left(\kappa_a^{n+1} + \kappa_s^{n+1}\right) I^{n+1} = \kappa_a^{n+1} B^{n+1} + \kappa_s^{n+1} \phi^{n+1}, \quad (2)$$

$$\phi^{n+1} = \frac{1}{4\pi} \int_{4\pi} I^{n+1} d\Omega, \quad (3)$$

$$(\rho E)^{n+1} = (\rho E)^- - \Delta t \cdot 4\pi \kappa_a^{n+1} \left(B^{n+1} - \phi^{n+1}\right). \quad (4)$$

The two coefficients κ_a and κ_s are for absorption and for isotropic scattering, respectively. Emission from the fluid to the radiation field is $\kappa_a B$, and I is the gray radiative intensity and is a function of both position and direction.

The discrete ordinate method discretizes the directional dependence of I by choosing a set of discrete directions Ω_m and weights w_m that satisfy appropriate quadrature properties on the unit sphere. Equation (3) becomes

$$\phi^{n+1} = \frac{1}{4\pi} \sum_m w_m I_m^{n+1}. \quad (5)$$

In Cartesian coordinates (2) remains essentially the same, with I replaced by I_m and Ω by Ω_m. Coupling between ordinate directions takes place only through ϕ (via scattering), and through reflections at the boundaries. (Axisymmetric and other curvilinear coordinate systems introduce an additional complication as the streaming term $(\Omega \cdot \nabla)I$ gives rise to angular differencing between ordinate directions. See [LM93] or [CL68] for this and other general information about discrete ordinates.)

The multifluid formulation stores separate volume, mass, and energy fractions for each material present in a cell. Quantities derived from the fluid state, such as emission and absorption coefficients, are likewise computed separately for each material, but for radiation quantities only a single cell-based value is used. There is thus a separate fluid energy equation for each fluid, and terms involving κ_a and κ_s in the radiation equation become sums over materials. These details do not significantly impact the solution algorithm and are omitted from the rest of this paper.

The implicit coupling between the radiation and fluid energy equations—most quantities in (2) and (4) are at time $n+1$—requires a nonlinear update iteration similar to those used in [HG98] and [HG03] for diffusion. First, the new-time emission term is expressed as an extrapolation from the current best approximation (designated by a star (*) superscript):

$$\kappa_a^{n+1} B^{n+1} = \kappa_a^* B^* + \frac{1}{c_v}(e^{n+1} - e^*)\frac{\partial(\kappa_a B)}{\partial T}. \tag{6}$$

By substituting into (2) and (4), we obtain

$$\frac{I^{n+1} - I^n}{c\Delta t} + (\Omega \cdot \nabla) I^{n+1} + (\kappa_a^* + \kappa_s^*) I^{n+1} =$$

$$(\kappa_s^* + \eta^* \kappa_a^*)\phi^{n+1} + (1 - \eta^*)\kappa_a^* B^* - \frac{\eta^*}{4\pi\Delta t}\left((\rho E)^* - (\rho E)^-\right), \tag{7}$$

$$(\rho E)^{n+1} = \eta^* (\rho E)^* + (1 - \eta^*)\left\{(\rho E)^- - \Delta t \cdot 4\pi\kappa_a^*\left(B^* - \phi^{n+1}\right)\right\}, \tag{8}$$

where

$$\eta^* = \frac{\dfrac{\partial(\kappa_a B)^*}{\partial T}}{\dfrac{\rho^{n+1} c_v}{4\pi\Delta t} + \dfrac{\partial(\kappa_a B)^*}{\partial T}}. \tag{9}$$

It remains only to group similar terms in (7) together to reduce it to a convenient form for development of the transport solver,

$$(\Omega_m \cdot \nabla) I_m + \sigma_t I_m = \frac{1}{4\pi}\sigma_s \sum_{m'} w_{m'} I_{m'} + S_m, \tag{10}$$

where

$$\sigma_t = \kappa_a^* + \kappa_s^* + \frac{1}{c\Delta t} \tag{11}$$

$$\sigma_s = \kappa_s^* + \eta^* \kappa_a^* \tag{12}$$

$$S_m = \frac{1}{c\Delta t} I_m^n + (1-\eta^*)\kappa_a^* B^* - \frac{\eta^*}{4\pi\Delta t}\left((\rho E)^* - (\rho E)^-\right). \tag{13}$$

Note that the effective scattering coefficient σ_s includes a contribution $\eta\kappa_a$ from the linearization of the emission term. This can be thought of as representing absorption and re-emission during the timestep, and can play a major role in the behavior of the solver even when true (physical) scattering is absent from the problem.

Finite volume spatial discretization of the streaming operator in (10) gives—in 2D Cartesian coordinates—

$$\frac{\mu_m}{\Delta x}(I_{m,i+\frac{1}{2},j} - I_{m,i-\frac{1}{2},j}) + \frac{\xi_m}{\Delta y}(I_{m,i,j+\frac{1}{2}} - I_{m,i,j-\frac{1}{2}}) + \sigma_t I_m = \frac{1}{4\pi}\sigma_s \sum_{m'} w_{m'} I_{m'} + S_m. \tag{14}$$

The simplest solution algorithm for this equation consists of repeated transport sweeps, holding the scattering (σ_s) term fixed. For each ordinate direction Ω_m, a sweep begins at the upstream corner of the domain and moves through the grid, using an upwind discretization to specify the relationship between the quantities at edges and centers of each cell. Many such closures are possible; the ones used for the calculations in this paper were the Step [CL68] and Simple Corner Balance (SCB) [Ada97] discretizations. The scattering term is updated after each sweep until convergence, or more sophisticated convergence acceleration techniques can be used.

3 AMR Timestep

The previous section described a timestep for a uniform mesh. The elaboration of this for AMR is very similar to the algorithm for diffusion I present in [HG03], which itself descends from previous AMR work on other systems combining explicit advection with some kind of implicit global coupling. Finer levels are advanced at smaller timesteps than coarser levels, in a recursive fashion. The basic outline is,

Multilevel timestep for levels $\ell \ldots \ell_{\max}$:

1. Advance level ℓ one timestep as a uniform mesh,
2. Do $\Delta t_\ell / \Delta t_{\ell+1}$ multilevel timesteps for levels $\ell+1, \ldots, \ell_{\max}$,
3. Synchronize intensities between levels ℓ and $\ell+1$.

The single level steps require calls to a single level transport solver, inside a nonlinear update loop coupling the radiation to the fluid energy. The synchronization operations require calls to a multilevel transport solver to compute a consistent, coupled solution across all active levels. This is a more complicated solver, but is typically

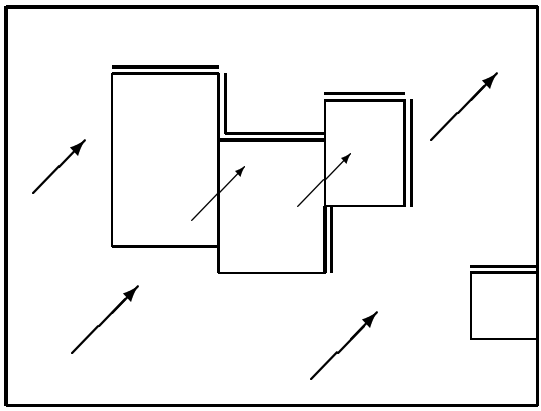


Fig. 1. Two level sweep: First sweep through the entire coarse level, then sweep the finer level grids using interpolated coarse data at their upstream edges. At the downstream edges (doubled lines), accumulate flux differences to be applied as internal edge sources during the next coarse sweep.

called only once per coarse timestep, not embedded within a nonlinear iteration. A variant of the multilevel timestep, involving an additional multilevel solve at the beginning of each coarse timestep, is also described in [HG03]. I have not implemented this variation for discrete ordinates but could easily add it in the future if needed.

On each level of grids, sweeps proceed grid-by-grid across the entire level, passing through upstream grids before downstream grids in each ordinate direction. (How to do this in parallel, with different grids on different processors, will be discussed in the next section.) At upstream edges of a level, intensities are interpolated from the next coarser level. The intent is that the computed solution depend on the region of cells making up the level, but not on the pattern of rectangular grids into which the region is broken up.

For multilevel solutions, the algorithm is similar to that described for a serial implementation in [JFHCP98]:

Multilevel transport sweep for levels $\ell, \ldots, \ell_{\max}$:

1. Sweep all ordinate directions at level ℓ,
   - At upstream edges interpolate intensities from level $\ell - 1$,
   - At downstream edges of level $\ell + 1$, increment intensities by the quantities stored in the flux registers from the previous sweep,
2. Do a multilevel sweep for levels $\ell + 1, \ldots, \ell_{\max}$,
3. Accumulate difference between level $\ell + 1$ and ℓ intensities in flux registers at downstream edges of level $\ell + 1$.

The "flux registers" mentioned here are data structures at the downstream edges of the fine grids, as shown in fig. 1. When transport sweeps are iterated to convergence, three quantities must be updated after each sweep: the scattering source, boundary

reflection source, and this new "refluxing source" associated with AMR. Fortunately, the refluxing source tends to converge more rapidly than either of the other two sources, so that the AMR algorithm does not require significantly more iterations than a similar single level calculation.

One minor difference with the current algorithm is that the sources are applied at cell edges, while in [JFHCP98] they were applied at cell centers. This only makes a difference for discretizations like diamond difference [CL68] and step characteristic [Mat99] that distinguish between fluxes entering different edges of a cell.

The same flux register data structures are used for accumulating differences in intensities for the synchronization phase of the multilevel timestep algorithm. In this case, the fine intensities are averaged in time (over the fine steps making up the coarse step), as well as in space (over the fine faces making up each coarse cell face). The goal for both cases is conservation, but for the multilevel transport solver it is conservation in an instantaneous sense, while for the timestepping scheme it is in a time-integrated sense.

4 Parallel Transport Sweep Algorithm

Parallel support for the code as a whole is provided by BoxLib [RBLCB00], This is a purely spatial decomposition, based on MPI, with different grids distributed to different processors. There is no provision for two or more processors to share a single grid. Each refinement level is distributed across the entire processor set, since in most parts of the code only one level is active at a time. No attempt is made to place physically-adjacent coarse and fine grids on the same processors. For reasonably efficient load balancing, it is best to divide the coarsest level into the same number of equal-sized grids as there are processors, and to arrange for there to be at least three times as many fine grids as processors on each level.

The radiation algorithm can influence the regridding operations by tagging cells for refinement in future timesteps, but it is not possible for any module of the code to adaptively alter the grid structure according to error criteria applied during the current timestep. Questions of load balancing, grid layout, error estimation, and refinement criteria are therefore outside the immediate scope of the radiation module. My emphasis in designing the radiation algorithm has been to take the grid structure as given, and compute the best and fastest radiation solution using the data provided on that grid.

The standard BoxLib primitives provide adequate support for most parallel operations, but transport requires more detailed control due to the dependencies between grids during sweeps. The left side of fig. 2 shows a simple 4×4 array of grids. An ordinate in the first quadrant must start in the grid marked "1" since all other grids are downstream of this one. After grid 1 is swept, that same ordinate can sweep through both grids marked 2 simultaneously—assuming these grids are on different processors—while another ordinate in the first quadrant can sweep grid 1. It takes seven steps to sweep a single ordinate through all of the grids, but with a large enough

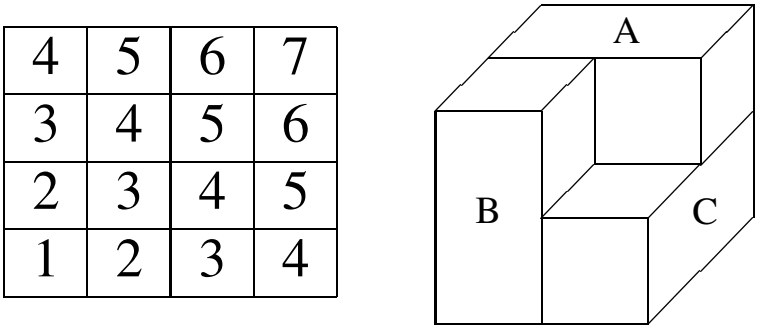


Fig. 2. Left: "Waves" of grids in 2D. Grids with the same number are able to sweep the same ordinate simultaneously; with seven different ordinates all 16 grids can be kept busy at once. Right: In 3D it is not always possible to sort grids into waves. Each of these three grids is in front of one of the others.

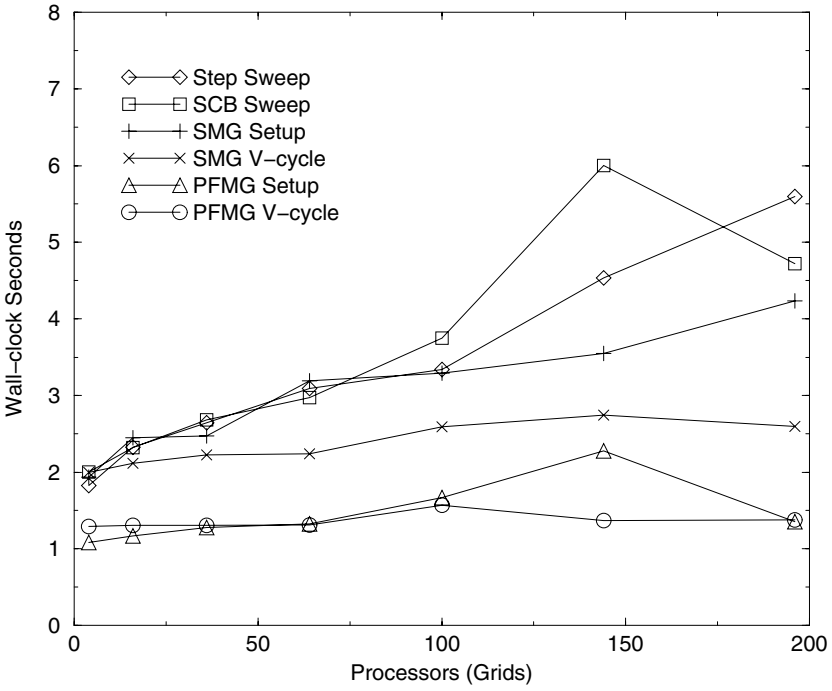


Fig. 3. Timings for 2D grids arranged in a square array, one grid per processor, each grid is 400×400 cells. S_n transport sweeps (Step and SCB) are for all 40 ordinates of an S_8 ordinate set.

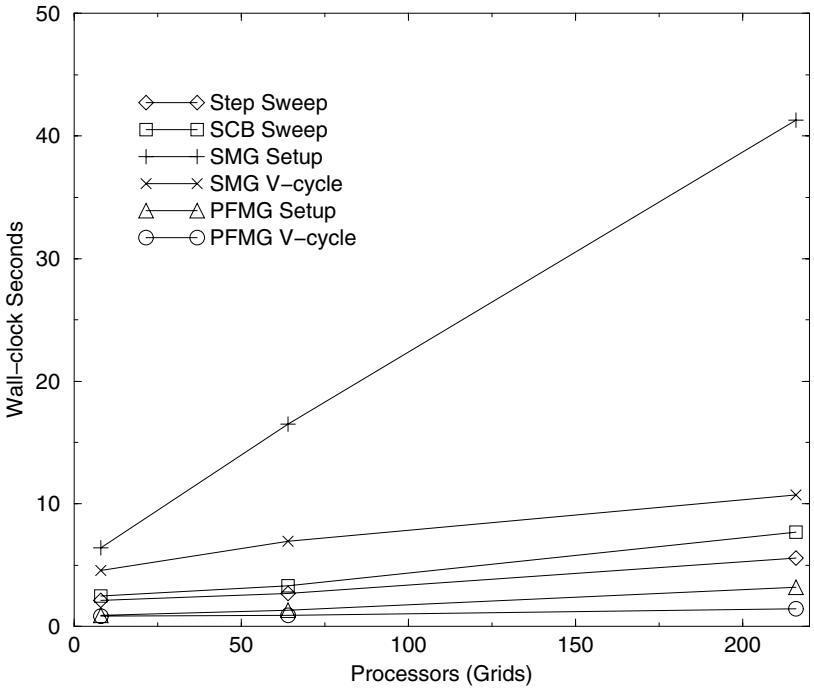


Fig. 4. Timings for 3D grids arranged in a cubical array, one grid per processor, each grid is $40 \times 40 \times 40$ cells. S_n transport sweeps (Step and SCB) are for all 80 ordinates of an S_8 ordinate set.

ordinate set the pipeline will be full for a while and all of the grids (processors) will be working most of the time.

The structures supporting the parallel sequencing are called *waves* and *stages*. Grids with no upstream dependencies make up the first wave, grids depending only on the first wave make up the second wave, and so on. Stages are the steps of the calculation; each stage specifies which wave is sweeping which ordinate at a particular time, and controls the communication of information from each grid to the next. During a single stage each grid only sweeps one ordinate, so there must be at least as many stages as there are ordinates to complete a transport sweep.

Not all ordinates are in the first quadrant, of course. In Cartesian coordinates the first stage for the 4×4 array of grids actually has the grids on all four corners active (cf. the algorithm in [DS96]). The angular differencing needed for axisymmetric coordinates introduces additional dependencies, so for that case only ordinates pointing towards the axis are able to sweep in the first stage, so only two corners can be active at first. 3D calculations can start inward from all eight corners. Adaptive mesh problems have more complicated grid layouts on each level, so the waves have a more irregular appearance, but the basic ideas work the same way.

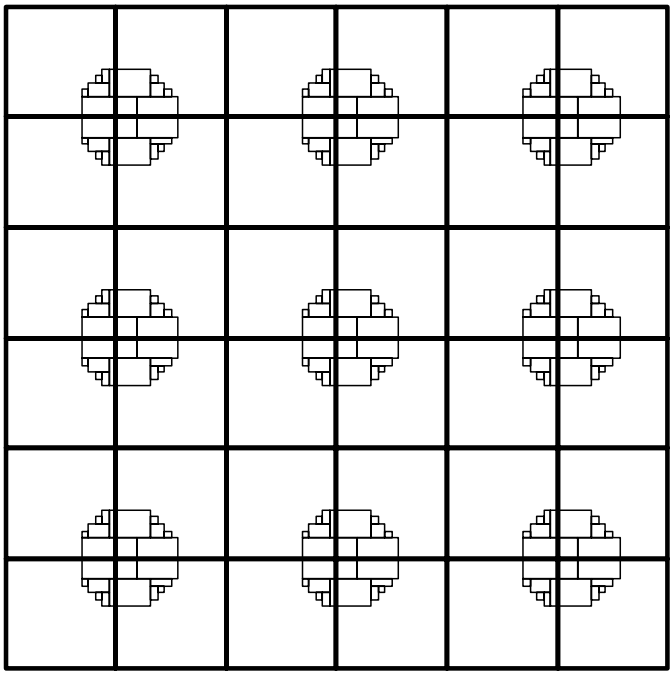


Fig. 5. This is the tiling pattern used in the 2D AMR scaling tests. Grids are arranged in a square array, with 4 coarse grids and 18 fine grids for every four processors. Each coarse grid is 256×256 cells.

Each quadrant of ordinates has its own wave structure. On a regular mesh the waves for opposing quadrants—1 and 3, for example—are clearly identical, just taken in the opposite order. I have found it beneficial to do the same thing for AMR meshes (that is, for quadrant 3 use the waves for quadrant 1 in reverse order), instead of building separate wave structures from the farthest upstream grids in all cases. The reason is so that ordinates from these opposing quadrants can sweep different waves of grids during the same stage of the calculation with minimal interference.

Choices must often still be made about which of two intersecting waves to sweep during a particular stage, with the goal of minimizing the number of stages required to sweep the entire ordinate set. I don't have a perfect algorithm for this sequencing problem, but I do have a number of heurstic rules that work fairly well. These can be considered tradeoffs between "order" and "evenness". "Order" is characterized by full pipelines: some quadrants take precedence over others, so those ordinates sweep through the mesh without interference. The drawback of order is that some quadrants finish well before others and thus leave large regions idle in later stages. "Evenness" is an attempt to treat all quadrants with equal priority, so that they finish in step and late stages have all corners of the mesh still active. The drawback of evenness is that pipelines are disrupted, leaving gaps. The additional constraints introduced by

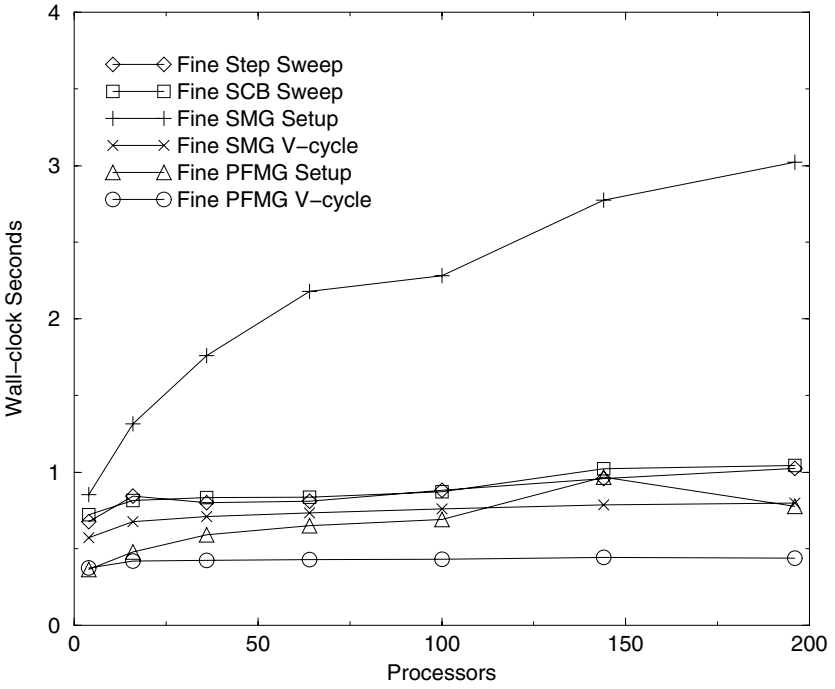


Fig. 6. Transport sweep and multigrid times for fine grids only using the tiling pattern shown in fig. 5. S_n transport sweeps (Step and SCB) are for all 40 ordinates of an S_8 ordinate set.

axisymmetric coordinates put a premium on order, while for Cartesian coordinates some form of evenness often gives a more efficient sequence of stages.

Since no single heuristic gives consistently shorter sequences, the code currently checks several schemes and picks the best for each particular grid layout. The expense of this is small but, as the next section suggests, may become a scaling issue when hundreds of processors are involved. In [How02] I give some more information about the sequencing tradeoffs, including figures and tables for various mesh sizes and ordinate sets.

One final sequencing difficulty that only arises in 3D problems is illustrated by the right side of fig. 2. Dependency loops can form for 3D adaptive grid layouts, so that each grid of a loop depends on one of the others. In order to sweep this kind of configuration at least one grid must be split into pieces to break the loop. Any of the three grids in the figure could be split. In my code grids are split only in the z direction—this is always possible—so that the resulting fragments have contiguous data. So the choice for this figure would be to split grid B along the plane separating grids A and C. The resulting four grids could then be sorted and swept in order.

The discussion so far in this section has focused on sweep patterns for a single level of grids, but these sweeps can be combined for multilevel problems in the manner presented in [JFHCP98] and summarized in the previous section. The parallel

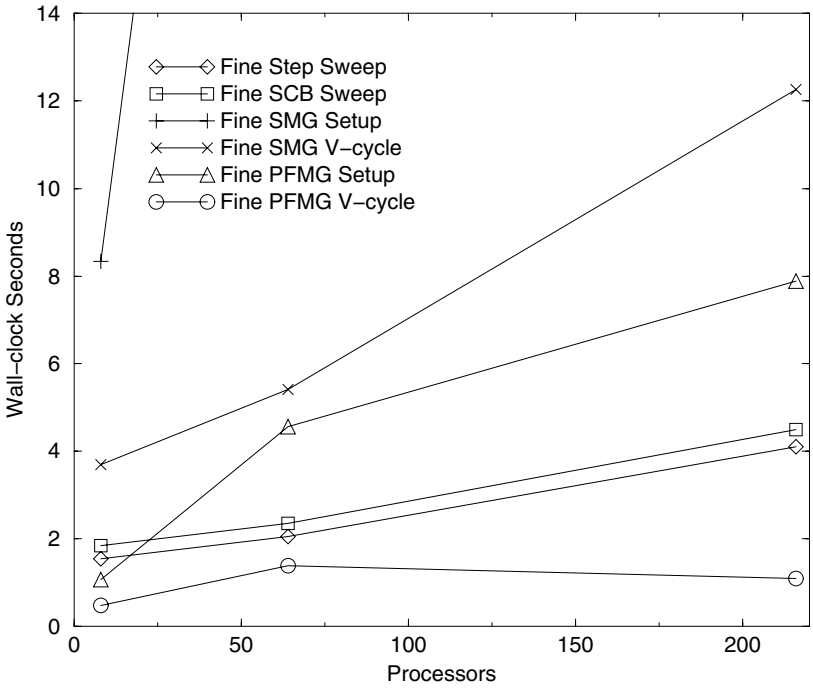


Fig. 7. Transport sweep and multigrid times for fine grids only using a 3D tiling pattern similar to that shown for 2D in fig. 5. Grids are arranged in a cubical array, with 8 coarse grids and 58 fine grids for every eight processors. S_n transport sweeps (Step and SCB) are for all 80 ordinates of an S_8 ordinate set. Fine SMG Setup times are off the scale, taking 41 and 126 seconds for 64 and 216 processors, respectively.

communication operations that must be implemented for the full multilevel transport scheme are then

1. Transfers from grid to grid on the same level according to the pattern of stages,
2. Transfers from a coarse level to upstream edges of a fine level,
3. Transfers from a coarse level to downstream edges of a fine level, to initialize flux registers,
4. Transfers from a fine level back to a coarse level as a refluxing source.

Some optimizations are possible. The third type of transfers (coarse data initializing flux registers) are only needed because at present the flux registers are associated with the fine grids. Implementation of coarse-processor data structures for holding this data would eliminate these transfers entirely. Also, when the communication operations are used during the single-level solves of an AMR timestep, it is not necessary to perform any transfers between levels while iterating a level to convergence, only when preparing to transfer converged solution information to another refinement level.

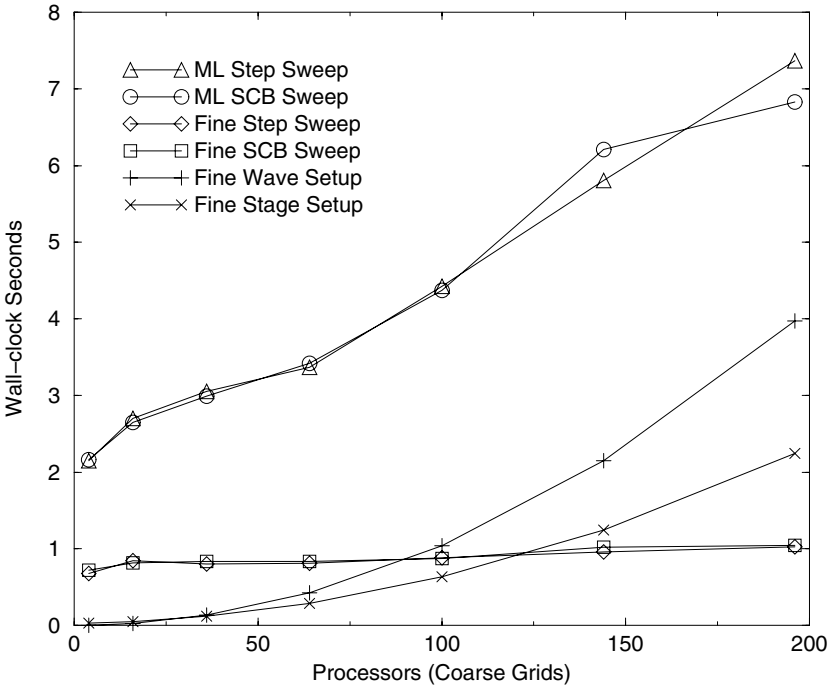


Fig. 8. Multilevel sweep timings compared to sweeps on the fine level alone. Fine level setup costs are also included for comparison. 2D grids are arranged in a square array of tiles, with 4 coarse grids and 18 fine grids for every four processors. Each coarse grid is 256×256 cells. S_n transport sweeps (Step and SCB) are for all 40 ordinates of an S_8 ordinate set.

5 Parallel Scaling

The runs presented in this section were all performed on ASCI Blue-Pacific. This IBM SP system has four processors per node, so the numbers of MPI processes used in the tests were always multiples of four. The number of cells per processor was the same for all the runs compared in each figure, so perfect scaling would be represented by flat horizontal lines on each plot.

The first tests (figs. 3 and 4) examine performance on a single uniform level of grids, with one grid per processor. In 2D the grids are arranged in square arrays of various sizes, and in 3D in cubical arrays. The times shown are for sweeping all ordinates of an S_8 ordinate set (40 ordinates in 2D, 80 in 3D) across each array of grids. For comparison, the times for the setup phase and a single V-cycle are shown for two geometric multigrid algorithms in the *hypre* library [CCF98], used in this code for radiation diffusion calculations [HG03]. Note that in comparing single sweeps and single V-cycles, I am comparing building blocks of solvers rather than complete solution times. Comparing solution times would bring in questions of algorithm efficiency, and for discrete ordinates would require an examination of acceleration tech-

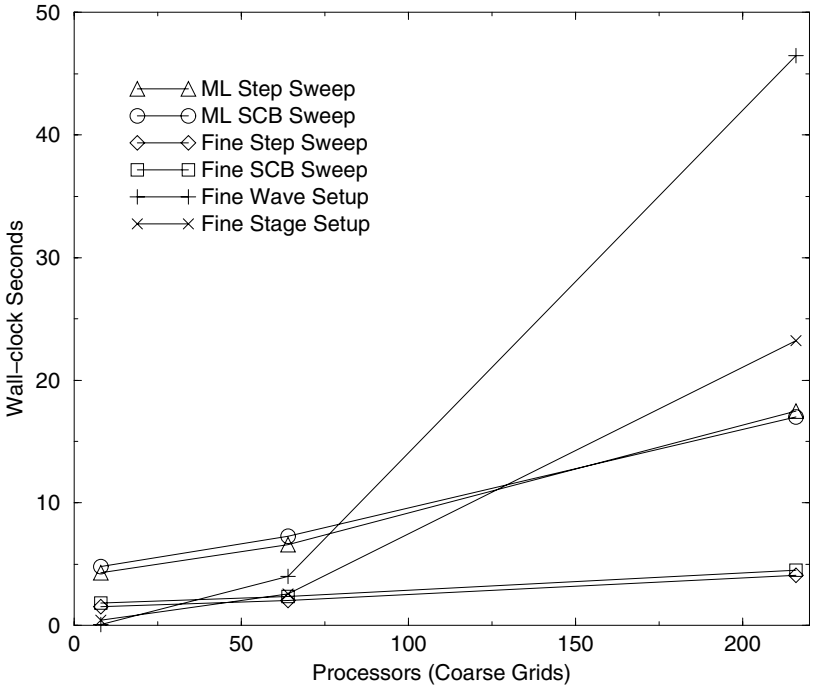


Fig. 9. Multilevel sweep timings compared to sweeps on the fine level alone. Fine level setup costs are also included for comparison. 3D grids are arranged in a cubical array of tiles, with 8 coarse grids and 58 fine grids for every eight processors. Each coarse grid is $32 \times 32 \times 32$ cells. S_n transport sweeps (Step and SCB) are for all 80 ordinates of an S_8 ordinate set.

niques and convergence behavior that is beyond the scope of this paper. (I show preliminary single-processor comparisons of acceleration schemes in [How02].)

In these two figures, as well as in the AMR results presented later in the section, the sweep times for the step and simple corner balance (SCB) discretizations are very similar. This may indicate that these routines are constrained more by memory access times than by floating point performance. (SCB does more arithmetic per cell, but both discretizations access the same amount of data.) It is also interesting to observe in these results that all 40 or 80 ordinates for an S_8 problem can be swept in a time comparable to that of a single multigrid V-cycle for a diffusion problem involving a single unknown.

The increase in times for the transport sweeps on larger arrays of grids can be partially accounted for by the greater numbers of stages required for the parallel computation. In 2D this ranges from 40 stages for 4 processors to 74 stages for 196 processors, while in 3D the increase is from 85 stages for 8 processors to 113 stages for 216 processors. (Remember that perfect efficiency would be 40 stages in 2D, 80 in 3D, the same as the number of ordinates in the S_8 set.) Another factor is the increase in communication per grid: In the smallest cases only half of the faces of

each grid require communication (the others being on the boundary). As the arrays of grids become larger, nearly all grid faces require communication. This latter effect applies to the multigrid cycles as well as to the transport sweeps.

To extend the scaling study from uniform mesh to AMR, we need a way to generate more interesting grid layouts while keeping the work per processor fixed. Figure 5 shows a tiling pattern I have used for these tests in 2D—the tiles are replicated to preserve the ratio of one coarse grid per processor. In this pattern, and its 3D counterpart, the refined regions in neighboring tiles do not touch each other. This means that the number of S_n stages, and the communication per fine grid, remain fixed as the pattern grows larger. (Other patterns where the fine regions do touch would also be interesting to look at, but this one allows me to eliminate these two effects from the comparison.)

The results for single-level sweeps across the fine level only are shown in figs. 6 and 7. The S_n results scale fairly well, and are flatter than those in figs. 3 and 4, showing the effect of the fixed number of stages and constant communication costs. The pattern chosen for 3D was complicated enough to require several grid splits to break dependency loops, as discussed in the previous section.

The scaling of the *hypre* solvers is not the main subject of this paper, but it is worth mentioning that the PFMG algorithm is showing good results, while the scaling for the SMG setup phase seems poor (particularly in 3D). I would like to stress that these tests were done only weeks before the workshop, and that the *hypre* developers had not had a chance even to reproduce and diagnose the effects shown, much less correct them. Since that time they have begun to look for the problems and work on improvements. A partial explanation for the adaptive mesh results is that *hypre* has previously been optimized mainly for problems with a single grid per processor.

The final two figures (8 and 9) show performance of the multilevel algorithm coupling coarse and fine levels. Multigrid comparisons are not shown since we do not currently use *hypre* for multilevel solves. The curves for the fine level alone are reproduced from the previous two figures. Note that the times for the multilevel solves rise with the number of processors, while the times for a single level remain relatively flat. This is because the tiles of the grid layout are in contact at the coarse level.

I also include fine level setup times in these last two plots. (Setup times for the coarse level are smaller and would be visible at this scale only for the largest numbers of processors.) The curve labeled “wave setup” shows costs independent of the ordinate set, while “stage setup” costs are those that depend on the ordinate set used. Since neither are scaling well, this is not a terribly important distinction. Though the setup costs can be amortized over the number of transport sweeps required to reach convergence, it seems clear that I will have to develop better setup implementations in order to push the code much beyond 200 processors.

6 Future Work

Further development of the discrete ordinates capability will proceed along at least three fronts: multigroup support, acceleration schemes, and applications. Experience with applications will feed back into the development work through code validation and through testing the practical limits of the implementation in various ways. Just as the scalability tests in the previous section indicate a need for further work in order to run on more than a few hundred processors, it is likely that applications tests will reveal a need for improved convergence acceleration methods beyond those already implemented [How02]. Additional physics may be needed once support for multiple frequency groups is in place.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

References

Ada97. Adams, M.L.: Subcell balance methods for radiative transfer on arbitrary grids. Transp. Theory Stat. Phys., **26**, 385–431 (1997)

BBSW94. Bell, J., Berger, M., Saltzman, J., Welcome, M.: Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. SIAM J. Sci. Comput., **15**, 127–138 (1994)

BC89. Berger, M.J., Colella, P.: Local adaptive mesh refinement for shock hydrodynamics. J. Comput. Phys., **82**, 64–84 (1989)

BO84. Berger, M.J., Oliger, J.: Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys., **53**, 484–512 (1984)

CL68. Carlson, B.G., Lathrop, K.D.: Transport theory—the method of discrete ordinates. In: Greenspan, H., Kelber, C.N., Okrent, D. (eds) Computing Methods in Reactor Physics. Gordon and Breach, New York (1968)

CCF98. Chow, E., Cleary, A.J., Falgout, R.D.: Design of the *hypre* preconditioner library. Proceedings of the SIAM Workshop on Object Oriented Methods for Interoperable Scientific and Engineering Computing. Yorktown Heights, NY, October 21–23, 1998

DS96. Dorr, M.R., Still, C.H.: Concurrent source iteration in the solution of three-dimensional, multigroup discrete ordinates neutron transport equations. Nucl. Sci. Eng., **122**, 287–308 (1996)

How02. Howell L.H.: A discrete ordinates algorithm for radiation transport using block-structured adaptive mesh refinement. Proceedings of the Nuclear Explosives Code Development Conference. Monterey, CA, October 21–24, 2002

HG98. Howell, L.H., Greenough, J.A.: A block-structured adaptive mesh refinement algorithm for diffusion radiation. Proceedings of the Nuclear Explosives Code Development Conference. Las Vegas, Nevada, October 25–30, 1998

HG03. Howell, L.H., Greenough, J.A.: Radiation diffusion for multi-fluid Eulerian hydrodynamics with adaptive mesh refinement. J. Comput. Phys., **184**, 53–78 (2003)

HPCJF99. Howell, L.H., Pember, R.B., Colella, P., Jessee, J.P., Fiveland, W.A.: A conservative adaptive mesh algorithm for unsteady, combined-mode heat transfer using the discrete ordinates method. Num. Heat Transfer B, **35**, 407–430 (1999)

JFHCP98. Jessee, J.P., Fiveland, W.A., Howell, L.H., Colella, P., Pember, R.B.: An adaptive mesh refinement algorithm for the radiative transport equation. J. Comput. Phys., **139**, 380–398 (1998)

LM93. Lewis, E.E., Miller, W.F. Jr.: Computational Methods of Neutron Transport. American Nuclear Society, La Grange Park, IL (1993)

Mat99. Mathews, K.A.: On the propagation of rays in discrete ordinates. Nucl. Sci. Eng., **132**, 155–180 (1999)

RAN97. Ramone, G.L., Adams, M.L., Nowak, P.F.: A transport synthetic acceleration method for transport iterations. Nucl. Sci. Eng., **125**, 257–283 (1997)

RBLCB00. Rendleman, C.A., Beckner, V.E., Lijewski, M., Crutchfield, W., Bell, J.B.: Parallelization of structured, hierarchical adaptive mesh refinement algorithms. Comput. Visual. Sci., **3**, 147–157 (2000)

Radiation Transport in AMR

P. Velarde[1] and F. Ogando[2,1]

[1] Instituto de Fusión Nuclear. C/ José Gutiérrez Abascal, 2. 28006 Madrid
pedro@din.upm.es
[2] Universidad Nacional de Educación a Distancia. C/ Juan del Rosal, 12. 28040 Madrid
fogando@ind.uned.es

1 Radiation transport in code ARWEN

1.1 Introduction

Radiation transport calculations have been implemented in the ARWEN code [14] as a means of energy transportation. For that purpose, the Boltzman equation (eq 1) is solved for the radiation intensity. The algorithm used includes finite differences for the spatial component, discrete ordinates for the angular dependence and multigroups of energy for the energy dependence of the radiation.

$$\frac{1}{c}\frac{\partial I}{\partial t} + \vec{\Omega}\cdot\nabla I + \kappa I = \varepsilon \tag{1}$$

The radiation calculations are coupled to the fluid dynamics. That means that the mutual interaction (through absorption and emission processes) with matter is taken into account in the calculations. The implementation of the interaction with matter results in a emission term dependent on the radiation intensity itself. The conventional algorithms present an important problem for achieving convergence in that case, resulting in the necessity of the introduction of more sophisticated acceleration techniques. The method of Diffusion Synthetic Acceleration (DSA) [3] has been implemented in our calculations, to overcome those difficulties.

All these calculations are performed under Adaptive Mesh Refinement (AMR) environment, to achieve a most optimum scheme of calculations. The algorithms concerning AMR involve both the resolution of single mesh calculations and the exchange of data between meshes and levels. This exchange of data, as commonly done in CFD [5], implies transmission of inward boundary conditions from coarse levels to finer ones, and correction of the coarse solutions by means of the finer ones.

All the handling of data specific from AMR has been treated using the BoxLib library [6], programmed in C++ and designed to simplify the programming of codes working with AMR. The library allows for parallel run in the code due to its internal structure [15] simplifying parallelization tasks to the programmer.

Another special feature of the transport of thermal radiation lays in the nonlinear dependence of the emissivity with the temperature. The resolution of the equation on the same domain, but using different levels of refinement, may lead to highly inconsistent values of emissivity, that results on the necessity of strong corrections among levels. A special method for calculations of emissivities has been introduced, to reduce the inconsistencies as much as possible. This method applies to single mesh calculation, but it is specially introduced to solve problems arising just in AMR calculations.

The above mentioned points have been the key points in the development of new algorithms for the resolution of the radiation transport equation using AMR. This points have been implemented in the ARWEN code, resulting in a proper treatment of the equation and leading to convergence.

1.2 The ARWEN code

The ARWEN code was designed to perform calculations of high temperatue and density fluids. These thermodynamic conditions are commonly reached in laser produced plasmas, with laser intensities over $10^{10}\mathrm{W}/\mathrm{cm}^2$. This kind of plasmas are obtained in inertial confinement fusion (ICF) experiments, astrophysical laser laboratories, extreme ultraviolet amplificators, etc, and examples of simulations with ARWEN are given at the end of the paper.

The code ARWEN solves numerically the compressible fluid dynamics equations with electron heat conduction and radiation transport. We separate this three main calculations by time splitting. The CFD part is a Godunov type scheme, with the Riemann problem solved in one or two temperatures (electron and ion) and with linear or parabolic reconstraction of profiles [8, 10]. For the electron heat conduction is solved by a multigrid technique with parabolic interpolation at boundaries [9]. Because heat conduction coeficients are flux-limited, special care must be given to handle the non linear behaviour of the equation. For instance, convergence is very slow sometimes, and the switching between different methods helps to accelerate the convergence in this case.

The equation of state (EOS) and opacities is another important part of the code, and we have paid much attention to that topic. The thermodynamic data is storaged in tables both in direct form, with all data dependent of density and temperature, and in inverse form, with all data dependent of density and internal energy. In one step both kind of tables are used, and some precaution is taken to assure this back and forward interpolation is consistent. Other conditions to the EOS that are enforced when possible are the Bethe ($\partial^2 p/\partial v^2\big|_S > 0$) and Smith medium condition ($v\partial p/\partial v \leq \gamma + \frac{1}{2}pv/e$). These conditions are related to the existence of a solution of the associated Riemann problem. The code Arwen can not handle phase transitions or elastic-plastic flow. When possible the EOS data is a tabular representation of a analytical EOS named QEOS[13], with a multiplier of pressure dependent of density and temperature. For opacities we use a tabular data obtained with the Jimena[12] code.

1.3 Coupling between radiation and matter

In radiation hydrodynamics, both the fluid and radiation fields are closely coupled to each other. Mainly that happens by the energy conservation equation, although at high temperatures the momentum exchange becomes also important. Radiation is emitted and absorbed depending on the thermal state of matter. Specially in optically thin media, that produces an non-local coupling of thermal states. Emission depends most strongly on temperature, and it can be modified as an effect from distant regions of the system. Those variation of temperatures of course introduce variations on the emission. These considerations have to be taken into account when designing a radiation fluid dynamic code. There are two ways of considering the proper emission: either the timestep is reduced in such a way that there are no appreciable temperature changes, or the emissivity changes are properly treated in the equation.

$$\varepsilon_\nu(T^1) \approx \varepsilon_\nu(T^0) + \frac{\partial \varepsilon_\nu}{\partial T}(T^1 - T^0) \tag{2}$$

Accordingly to equation 2, the emissivity is considered in an implicit way that is approximated by its linealized form. This expression is introduced in equation 1, and by a simple energy balance of matter following reference [3], the expression 3 is reached.

$$\vec{\Omega} \cdot \nabla I_\nu + \kappa I_\nu = \chi_\nu \int_E \sigma_{f,\nu} I_\nu \, \mathrm{d}\nu + S_\nu \tag{3}$$

where the coefficient of the new equation depend on both the atomic and thermal properties of the contituent matter. Other atomic properties, as is absorption coefficient, are the same way influenced by the temperature variations, but they are not considered, since the overall effect is smaller than the originated from emissivity. The consideration of variable coefficients with temperature would lead to a non-linear transport equation, increasing enormously the complication of its resolution.

The resolution of equation 3 not only increases the required computational effort, but it introduces a serious problem in the convergence rate of the variable source. The problem has already been studied for neutronics, and special algorithms, named as *acceleration techniques* have been developed [1]. From those algorithms, DSA has been chosen for producing best results. The algorithm has been implemented in the calculations at a single-mesh level, that is, inside mesh calculations, but not specifically on the AMR algorithms.

The improvements regarding this modification appear in situations where the temperature suffers from sudden rise. That may happen in illumination by laser or thermal radiation, collision of hypervelocity systems, etc...

1.4 Interaction among meshes in the AMR scheme

Block structured AMR [4] involves both calculations in homogeneous single meshes, and special exchange of information among them. AMR meshes are structured in a hierarchycal tree involving different levels of refinement. Inside every level of the tree, all meshes are non-overlapping and have the same cellsize. Some of the meshes

inside the same level share at least part of the boudaries, so that common information has also to be shared among them.

The interaction ways among meshes may be organized in three main sets:

- Interaction between meshes in the same level.
- Fine boundary conditions from coarse meshes.
- Correction of coarse calculations from finer results.

and each of these classification sets has to be treated in a special way for the radiation transport equation.

The transmission of information among meshes belonging to the same level provides boundary conditions from neighbour contiguous meshes. Accordingly to [7], this transmission of information has to be done following a downstram propagation. That involves ordering the meshes belonging to the same level, given a direction of propagation. That process means no special problem for bidimensional calculations, although it need special treatment in 3D cases. An additional problem arising in discrete ordinates lays in the different directions of propagation of the different angular discretized directions. In the 2D case it results on four possible orderings. There are two ways of treating the calculations:

- Handling each direction of propagation separately.
- Performing all four sweeps for all directions.

In the case of this code, in order to implement DSA properly, the kernel of calculations has been extracted from the DANTSYS system [2] for neutronics. That makes impossible to do separate sweeps for the different directions solved. The solution applied is therefore to perform four different sweeps following the four possible orderings. That ensures that every direction has been swept at least once following the right direction.

The transmission of intensity values between meshes in the same level is straightforward. Using the discrete ordinates algorithm, intensity values are calculated both in cell body and in cell sides. The latter values and directly transportable as boundary conditions to neighbouring meshes.

During the full AMR calculation, fine meshes take under certain circumstances boundary conditions from coarser meshes already calculated. That transmission of information from coarse to fine meshes is produced in an alike manner as in fluid dynamics. A conservative procedure is followed, assuring that the total flux crossing the coarse-fine boundary has the same magnitude measured from both the coarse or fine side.

The transmission of data, as shown in figure 1, implies interpolating the coarse data. From that process, fine values are obtained, and used as inward boundary conditions for the fine mesh. It is important to remark that the process is a per-direction interpolation, that is, radiation comes into the fine mesh only for some of the directions of propagation (exactly for half of them). It is just for those directions pointing into the fine mesh, that the process must be done.

Once performed the finest calculations the so produced data, believed to be the most accurate, replaces and corrects the data obtained from coarser meshes. The

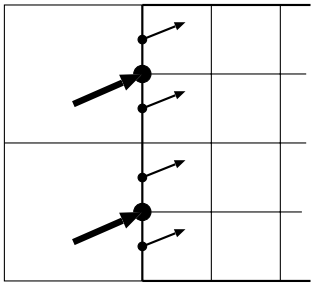


Fig. 1. Data transfer from coarse to fine mesh.

replacement takes place in the coarse regions overlapped by finer meshes. That replacement is done in a conservative way, averaging properly the values of intensity over a coarse cell. Another correction from those finer values take place by the outgoing flux coming from fine meshes. In the case of CFD, due to restrictions on the Courant number, that influence is restricted to the neighbouring line of cells surrounding the finer meshes. In radiation transport, on the contrary, the differences in boundary fluxes may influence distant regions. That point forces the recalculation of coarse meshes having into account the outgoing fluxes from finer meshes.

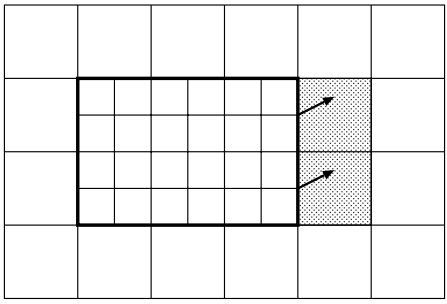


Fig. 2. Flux correction from finer to coarser meshes.

That correction applies to intensity values on cell sides of the coarse mesh (see figure 2). The introduction of precalculated fluxes is straightforward to implement in a normal algorithm of discretes ordinates but again the DSA implementation makes it more difficult to introduce. Having a starting point of a solver code that admits volumetric sources with dependance on the direction, the method applied consists of a way of converting the flux correction into a volumetric source.

The volumetric source, as shown in figure 3, is calculated to produce the same outgoing flux in the coarse cell neighbouring the fine mesh. Once it is done so, the radiation propagates the same way in both cases (contour or volumetric corrections). Using a one-dimensional aproximation for the propagation of the corrections under the assumption of no coupling with matter, the expression 4 is reached:

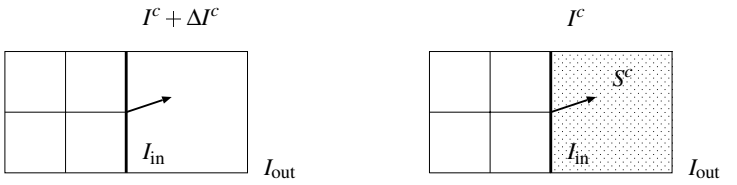


Fig. 3. Equivalence between flux correction and volumetric source.

$$\Delta S = \max\left(\frac{\mu}{\Delta x} - \frac{\kappa}{2}, 0\right) \cdot \Delta I^c \tag{4}$$

where ΔI^c represents the correction in incoming boundary intensity.

Once introduced the commented corrections, the outgoing flux of the neighbouring coarse cells is changed into the correct value, but even so the cell centered value remains incorrect. In figure 4 there is a simple scheme of the linealized intensity profiles inside a cell, in both the cases of contour correction (exact case) and the result of applying a volumetric source. It is easily noticed, that imposing a common outgoing intensity (right edge) does not mean at all that the cell-centered intensity takes also the proper value. It is therefore necessary to introduce another correction, since cell-centered intensities are used for energy balances. Checking figure 4 it is clearly seen that the necessary correction takes the value of $\Delta I_0^c = \Delta I^c/2$, half the value of the correction of the incoming intensity. This expression is valid in both the cases of coupling with matter or not.

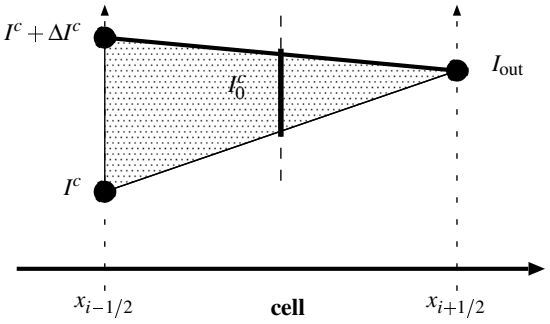


Fig. 4. Correction to the cell-centered intensity.

1.5 Calculation of sources

As it was pointed out before, in block structured AMR, some parts of the domain are calculated with different cellsizes, corresponding to different levels in the hierarchy. Regarding radiation transport, the coefficients of the equation are calculated using the thermodynamic states of matter calculated in every of the levels. Those thermodynamical states are granted to be consistent, since the conservative algorithms

used in CFD keep balances in some keys magnitudes (density, energy and momentum). However, even under the assumption of total conservation of those magnitudes (what is ensured by conservative projection of values), there may appear important inconsistencies in derived magnitudes. In highly nonlinear derived magnitudes, the obtained result is far from being conservative. That may result in big inconsistencies in the final solution, when it applies to sources that affect the balance of radiation in the system. Of course this undesired effect applies to every coefficient, since atomic properties of matter shows typically a highly nonlinear dependency with temperature (being closer to linearity in respect to density). All these inconsistencies can be treated by the interlevel flux correction, that have been presented in previous sections. However, and due to the application of approximation, it is desirable that those corrections are as small as possible. This is why there is also an aim of reducing the source of the inconsistencies, and not only correcting them (which will also be done).

In order to reduce those differences of values corresponding to the same region of the system, two main techniques have been implemented:

- Projection and replacement of values to ensure conservation of desired values.
- Consideration of first order profiles of thermodynamical properties to produce more accurate values on the regions not covered by finer meshes.

The first of this technique has no difference from the application it has in CFD calculations. It produces consistency on the coefficients and source of the transport equation, with a result of more consistent radiation fields in the different levels of refinemente. However still arises one question when doing calculation without extremely fine meshes. If the existing levels present inconsistencies, is the finest level reliable enough to believe its values? In a infinitely refined mesh, of course calculated data is reliable, but specially in rough meshes there may still be wrong values in the finest level.

Those regions where the inconsistencies appear take place where there are bug gradients of the thermodynamical data, specially temperature. Those areas may be localized in simulations applied to nuclear fusion around shock waves or ablation fronts, where temperature may rise up to five orders of magnitude in relatively short distance (few cells in the mesh). A special method must be implemented to take this particularities into account. Of course, in the bulk of the system with smaller gradients, this correction will have little effect. The same way AMR works, the correction (which means a calculation overhead) applies just in the regions where it is needed, thus optimizing the calculation efforts. However those regions, for example in the boundary of cold solid matter and hot plasma, may emit most of the existing radiation in the system.

Several correction methods may be developed to take into account the internal profiles of atomic values. In our code, calculations have been focused just to the emissivity. The correct value of the average emissivity in a cell is the following:

$$\varepsilon_i = \frac{1}{\Delta x_i} \int_i \varepsilon(x)\, \mathrm{d}x = \frac{1}{\Delta x_i} \int_i \varepsilon\left(\rho(x), T(x)\right)\, \mathrm{d}x \tag{5}$$

while the commonly accepted aprroximation is:

$$\hat{\varepsilon}_i \approx \varepsilon(\rho_i, T_i) = \varepsilon\left(\frac{1}{\Delta x_i}\int_i \rho(x)\,\mathrm{d}x, \frac{1}{\Delta x_i}\int_i T(x)\,\mathrm{d}x\right) \tag{6}$$

An analytical approximation for the emissivity has been considered [11], as well as a exponential spatial profile of the thermodynamical variables.

$$\varepsilon = a \cdot \rho^b \cdot T^c \tag{7}$$

$$\log\hat{\rho}(x) = \log\rho^0 + \rho^x \cdot x \tag{8}$$

$$\log\hat{T}(x) = \log T^0 + T^x \cdot x \tag{9}$$

Using this approximations, the actual value of the emissivity may be integrated using expression 5, leading to a result of the form:

$$\varepsilon = \hat{\varepsilon} \cdot F(\rho^x, T^x) \tag{10}$$

where $F(\rho^x, T^x)$ reflects the effect of internal profiles in the thermodynamical variables. It is straightforward to see that $F(0,0) = 1$, corresponding to a flat spatial profile. Function F depends on the approximation done in the emissivity and typically will vary from material to material, and in certain cases even for different regions. In figure 5 can be seen the values of F for different values of the slope parameters, that in a normal simulation may have values up to 3 or 4.

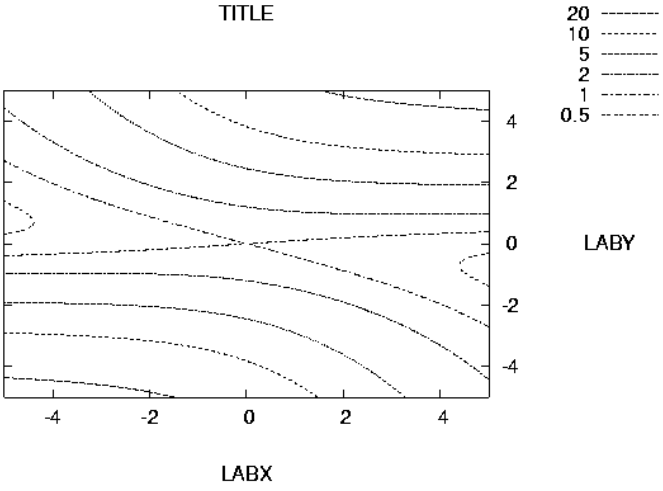


Fig. 5. Values of the correction factor for gold

1.6 Examples

In the following we will give an actual example of running the ARWEN code. In the figure 6 is represented the density evolution of a ICF guided target. First a X ray thermal radiation drives the compresion of the shell and afterward the same thermal radiation produces a jet by acumulation phenomena in the laser axis. This jet collides with the compressed shell, increasing the temperature to the ignition conditions. Typically three levels of AMR are used, being the finest one dedicated only to the laser energy source when is present. It allows a good precision for the critical density, where the most part of the laser energy is deposited. In this simulation, radiation burn throw the shell, heating the inner surface of the target and filling the interior with a medium temperature plasma.

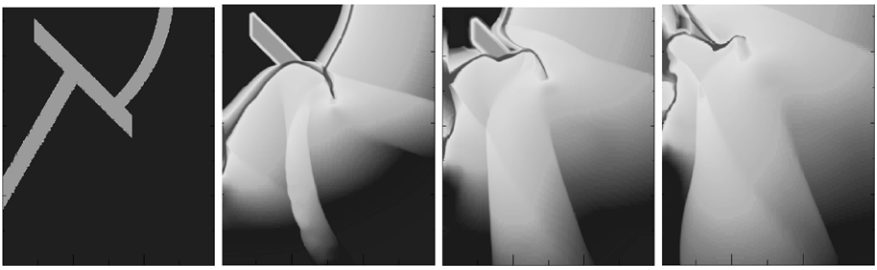


Fig. 6. Density evolution of a ICF guided target, driven by thermal radiation of 300 eV.

2 Acknowledgements

This work has been supported by Spanish Ministerio de Ciencia y Tecnología, with project number FTN2001-3885.

References

1. M.L. Adams and E.W. Larsen. Fast iterative methods for discrete-ordinates particle transport calculations. *Progress in nuclear energy*, 1:3–159, 2002.
2. R.E. Alcouffe, R.S. Baker, F.W. Brinkley, D.R. Marr R.D. O'Dell, and W.F. Walters. DANTSYS a diffusion accelerated neutral particle transport code system. Technical Report LA-12969-M, LANL, 1995.
3. R.E. Alcouffe, B.A. Clark, and E.W. Larsen. *Multiple time scales*, chapter The Difussion-Synthetic acceleration of transport iterations, with applications to a radiation hydrodynamics problem, pages 73–111. Academic Press, Orlando, Florida, 1985.
4. M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comp. Phys.*, 82:64–84, 1989.

5. M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53:484–512, 1984.
6. W.Y. Crutchfield and M.L. Welcome. Object-oriented implementations of adaptive mesh refinement algorithms. *Scientific Programming*, 2:145–156, 1993.
7. L. Howell, R. Pember, P. Colella, J.P. Jessee, and W. Fiveland. A conservative adaptive-mesh algorithm for unsteady, combined-mode heat transfer using the discrete ordinates method. *Numerical Heat Transfer Part B: Fundamentals*, 35:407–430, 1999.
8. P. Colella J.B. Bell and J. Trangenstein. Higher order godunov methods for general systems of hyperbolic conservation laws. *J. Comput. Phys.*, 82:362–397, 1989.
9. D. Martin. Solving poisson's equation using adaptive mesh refinement (amr). Technical Report see http://seesar.lbl.gov/anag/staff/martin/AMRPoisson.html, LBNL, 1999.
10. G.H. Miller and E.G. Puckett. A high order godunov method for multiple condensed phases. *J. Comput. Phys.*, 128:134, 1996.
11. E. Mínguez, R. Ruiz, P. Martel, J. M. Gil, J. G. Rubiano, and R. Rodríguez. Scaling law of radiative opacities for ICF elements. *Nuc. Instr. and Meth. in Phys. Res. sect A*, 464(1-3):218–224, 2001.
12. E. Mínguez, J.F. Serrano, and M.L. Gámez. Analysis of atomic models for the extinction coefficient calculation. *Las Part Beams*, 6:265–275, 1998.
13. R.M. More, K.H. Warren, D.A. Young, and G.B. Zimmerman. A new quotidian equation of state (QEOS) for hot dense matter. *Phys. Fluids*, 31:3059, 1988.
14. F. Ogando and P. Velarde. Development of a radiation transport fluid dynamic code under AMR scheme. *JQSRT*, 71:541–550, 2001.
15. C.A. Rendleman, V.E. Beckner, M. Lijewski, W.Y. Crutchfield, and J.B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Comp Vis in Science*, 3, 2000.

Part III

Software

HERA: A Hydrodynamic AMR Platform for Multi-Physics Simulations

Hervé Jourdren

CEA/DAM - Ile de France, Dpartement Sciences de la Simulation et de l'Information, BP 12 - 91680 Bruyres-Le-Chtel, France

Summary. The development at CEA/DAM of a new AMR multi-physics hydrocode platform led to convincing results on a wide range of applications, from interface instabilities to charge computations in detonics.

In this paper, we focus on:

1. A selection of numerical results illustrating gains to be expected from AMR in such fields, including precise comparisons between AMR and uniform grids (up to 100 millions cells in 2D using CEA's teraflops machine TERA-1).

2. An introduction to the hyperbolic framework and resulting suite of consistent multimaterial compressible flow solvers (hydrodynamics, hypo-elasticity, nT hydro and nT MHD).

3. A presentation of an innovative hydrocode architecture, allowing three different parallel modes at runtime: (i) a MPI mode for uniform or well-balanced AMR grids, (ii) a multithread mode on SMPs and (iii) a hybrid MPI/multithread mode on clusters of SMPs. Multithreading is used there to diminish grain sizes, to control memory cache effects and dynamic load balancing.

4. Finally, an overview of the user-model API is given, in both C++ and Python vector modes, for platform extensions using Strang-type operator splitting.

1 Multifluid hydrodynamics

For many users of traditional Lagrangian, ALE or Eulerian hydrocodes, one of the first objections to AMR is usually that "soon or later, AMR grids have to be refined everywhere to capture the growing complexity of most unsteady flows". Detailed numerical comparisons between AMR and uniform grids of same finest resolutions are then quite helpful. Up to now, such comparisons are not widely available in the literature. The next two examples have been used for a number of years, among others, to get support for the HERA[1] project. The third example involves a more complex flow, making use of CEA's teraflops machine TERA-1 to get a reference uniform grid result, for comparison to AMR in 2D.

[1] French acronym for Hydrodynamique Euler Raffinement Adaptatif.

Single-mode Richtmyer-Meshkov

The goal of the first test problem is to illustrate that cell-by-cell AMR, pionieered more than a decade ago in multifluid context using interface tracking [3] or concentration equation [5], may reveal surprisingly efficient when used with interface reconstruction and elementary grid strategies.

As illustrated Fig. 1, the finest level of 2x2 AMR refinement is applied on the incident shock, on the transmitted shock during the first instants, and on the interface during the whole computation. The comparison of AMR and uniform grid results clearly indicates that the interface instability is well-reproduced using such a simple AMR strategy, leading to a substantial reduction of the total number of cells and CPU time.

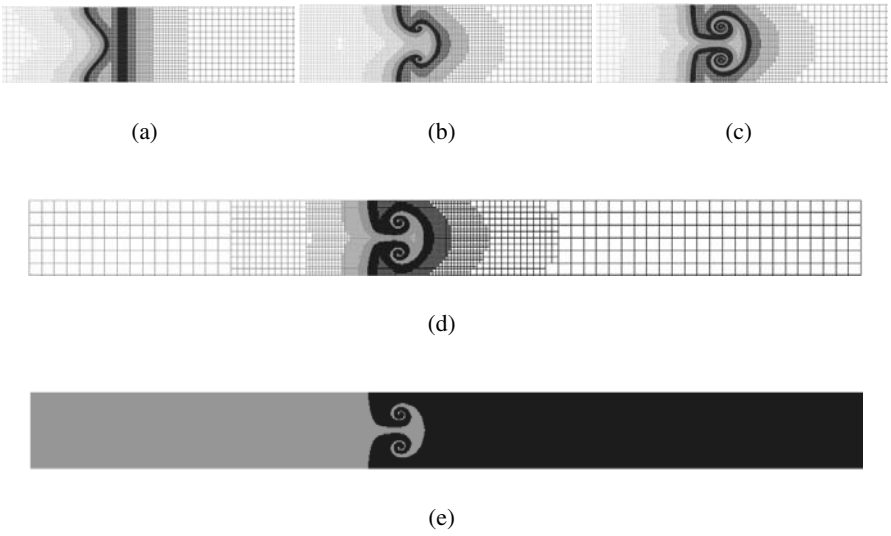
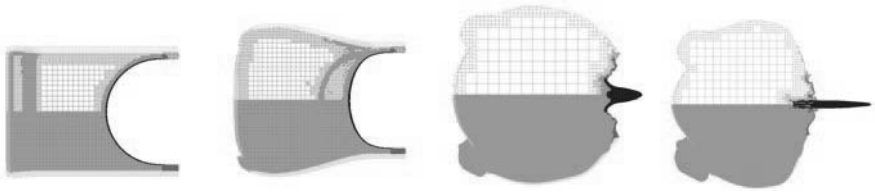


Fig. 1. Single-mode Richtmyer-Meshkov instability using a cell-by-cell 2D/AMR grid with interface reconstruction (StonyBrook #1 benchmark, 96 cells per wavelength). The AMR computation is about 7 times cheaper than the uniform grid one, without any significant difference on the development of the interface instability.

Shaped charge computation

In a different context – armor-anti armor – but with a very same AMR strategy, the finest 3x3 refinement level is here imposed on the detonation wave treated by a programmed burn approach, on the reflected shock in the detonation products, and on the liner during all the computation.

Again, the agreement between the AMR computation and the uniform grid one is quite satisfactory (Fig. 2), leading to a 24 reduction factor in CPU time (Alpha processor: 5 minutes instead of 2 hours). Such a reduction factor is representative of this type of computation in 2D. The 3D/AMR result is provided Fig. 3 with a comparison to the 2D/AMR axisymmetrical one, exhibiting an excellent agreement on the shape of the jet and associated velocity distribution on the axis.



(a) (b) (c) (d)

Fig. 2. 2D/AMR hemispherical shaped charge computation (a) 2.5 microseconds, (b) 12.5 microseconds, (c) 27.5 microseconds, (d) 40 microseconds. Four-level 3x3 AMR (top) *vs.* uniform reference grid (bottom).

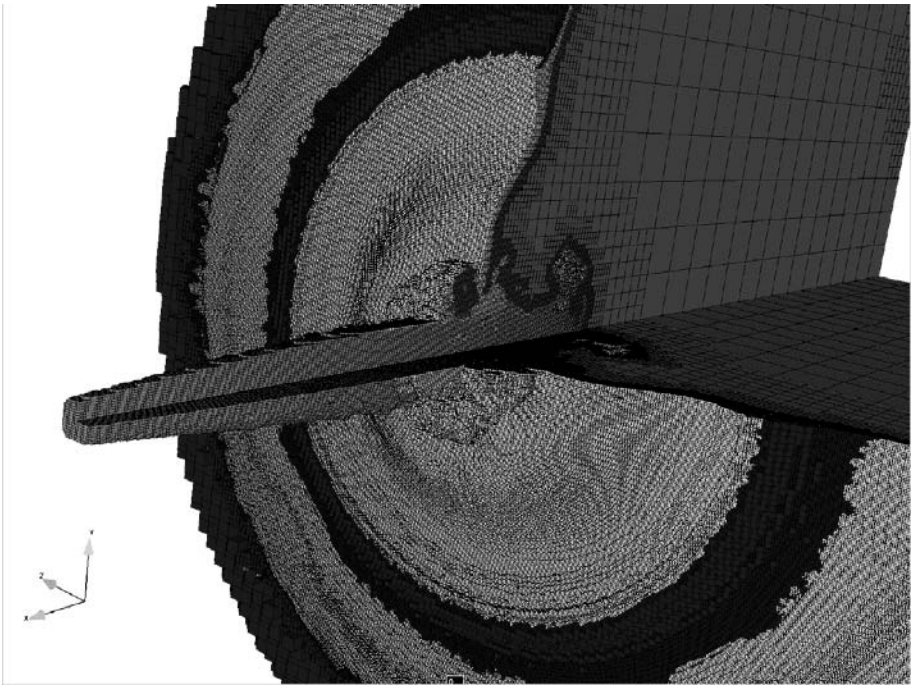
Multi-mode Richtmyer-Meshkov

Let's consider finally a more complex flow, combining features of the two previous ones: multi-mode instabilities in a cylindrical implosion, involving a succession of HE and inert materials as sketched Fig. 4.

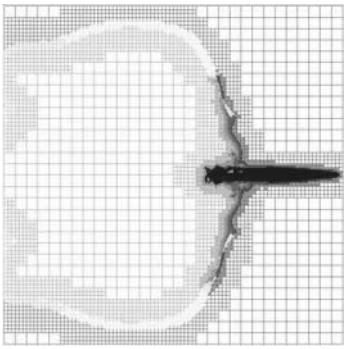
The grid strategy is exactly the same as before, with the finest grid resolution on the detonation waves – treated now by a reaction rate model – and on the perturbated liner. A difference with the two previous examples is the initialization of the liner, performed when it's just about to be reached by the flow, such a capability being used here to save a *very* significant amount of computer time. With a spatial resolution of 15 microns, the reference uniform grid computation involves 100 millions cells, requiring 20,700 hours of CPU time with CEA's teraflops machine TERA-1 (256x80 hours, Alpha EV68 1Ghz processor). The AMR computation requires 3.5 millions cells by the end of the computation and about one week in mono-processor mode. As illustrated Fig. 5, the positions of the liner are extremely close in both computations, along with development of the interface instabilities.

2 Hyperbolic solvers

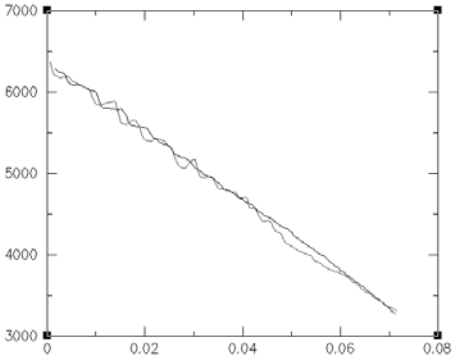
The choice of AMR *orthogonal* grids has been motivated in this project by a number of numerical issues, including:



(a)



(b)



(c)

Fig. 3. 3D/AMR hemispherical shaped charge computation (a) final geometry at 40 microseconds, (b) 3D/AMR geometry (top) *vs.* 2D/axi (bottom) at 38 microseconds, (c) jet velocity profile on the axis (solid line for 3D/AMR, dash line for 2D/axi).

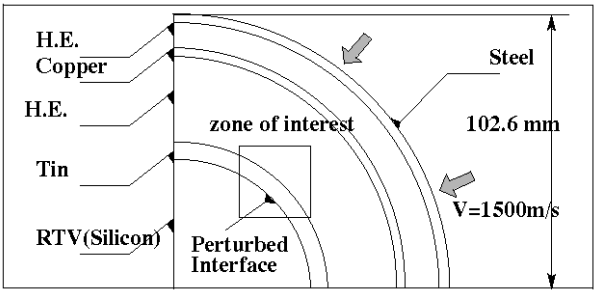
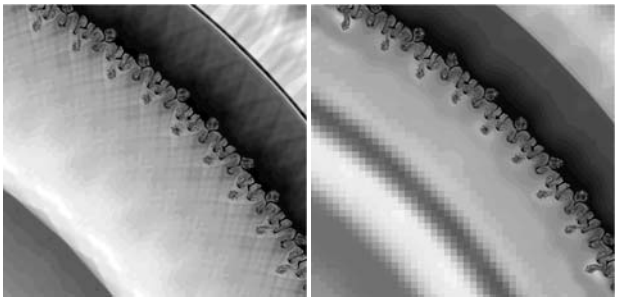
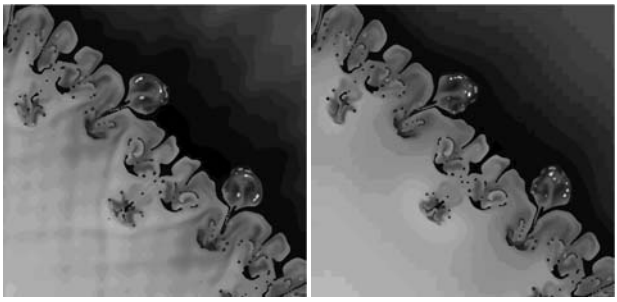


Fig. 4. Multi-mode Richtmyer-Meshkov test problem. Initial 2D plane geometry.



(a) (b)



(c) (d)

Fig. 5. Multi-mode Richtmyer-Meshkov test problem. Zooms at 10 microseconds of the 100 millions cells (15 microns) uniform grid (a, c), and AMR grid of about 3.5 millions cells, with same spatial resolution on the reaction zones and perturbated liner (b, d). A complex acoustic wave structure is clearly noticeable with the uniform grid. It's not the case in the AMR case, without any visible influence however on the position of the interface and the development of the instability.

i) the nice properties of the resulting diffusion operator, called by most plasma physics models (positivity, convergence, etc)

ii) the natural AMR extension of advanced ADI interface reconstruction algorithms for multifluid advection [4], [6]

iii) and also efficiency criteria, with simple formulae for volumes, surfaces, particules trajectography, etc.

In a truly multi-physics context, a general treatment of discontinuous solutions of hyperbolic systems of conservation laws is still challenging (i.e. derivation of a general shock capturing scheme [1], [2]). The choice of a conservative approach seemed quite natural there[2], but also appeared as a major source of scepticism since "extensions of Godunov-type hydro schemes to systems as elasticity, radiation hydrodynamics or MHD are still in their infancy, when compared to VNR-type staggered grid schemes".

The resulting mathematical framework worked-out at CEA/DAM for this purpose is detailed in [7], providing a consistent suite of multi-physics AMR flow solvers[3].

3 Parallelism and dynamic load balancing

It's often said that "cell-by-cell AMR may look attractive from a user point of view, but leads to severe difficulties on the parallel side".

The parallel strategy explored with the HERA platform involves advanced programming techniques. Portability being a crucial issue, a combination of message passing and multithreading has been selected[4], with a strict domain decomposition SPMD approach. The grain size of a parallel simulation is controled by using an arbitrary number of subdomains per processor (overloading factor). Each subdomain is then treated by a thread or eventually by a process. Thanks to the choice of the C++ programming language, *object oriented* polymorphism is used for all operations related to subdomains (process or thread) and message passing (MPI or shared memory), communications within same address spaces being then handled automatiquely via shared memory, otherwise via MPI.

As a consequence, with completely thread-safe and reentrant AMR solvers and physical modules, the HERA architecture supports three different modes of parallelism at runtime:

[2]Rankine-Hugoniot conditions are satisfied automatically with conservative schemes; AMR implementations are also much easier than VNR staggered grid ones, most notably in 3D with arbitrary refinement factors as 2x2x2 or 3x3x3.

[3]Recovering by construction, for example, the basic hydro scheme when deviatoric stresses (resp. magnetic field components) are set to zero in the elastic solver (resp. MHD solver).

[4]OpenMP has been disregarded for portability reasons, implementations being platform dependant, without *any* control on the underlying thread scheduling algorithm (a crucial issue when used in pure SPMD domain decomposition mode along with MPI).

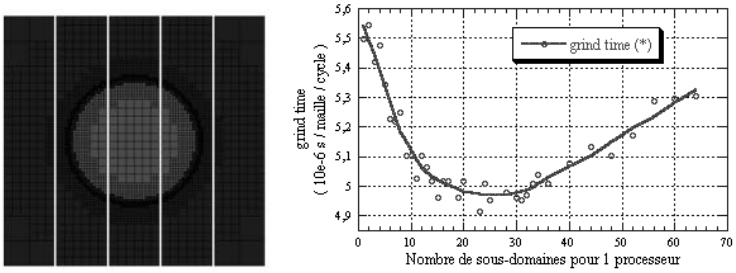
i) a MPI mode, for uniform or well-balanced AMR grids;

ii) a "pure" multithread mode on SMPs;

iii) and a hybrid MPI/multithread mode on clusters of SMPs.

In the third mode, a user-space non-preemptive multithreading approach is used, with a specific scheduling policy tuned to the needs of AMR computations. Within a HERA process, the thread scheduler is then modified so that all number crunching operations are performed first, communication threads being given the CPU resource only when *all* computing threads of the process are blocked on a communication phase. Such a thread scheduling policy helps keeping CPU busy as long as possible, and minimizes the number of context switches to benefit from cache effects.

As a first illustration of the capabilities of such a parallel code architecture, grind times for a 2D/AMR hydro test problem of about 1.4 millions cells on *one* processor are given Fig 6 with increasing overloading factors. Up to 30 computing threads per process, a significant grind time reduction appears, resulting directly from cache effects on the Alpha EV68 processor.



(a) (b)

Fig. 6. Multithreaded hydrodynamics in mono CPU mode (a) structure of the four-level 1.4 million cells 3x3 AMR grid and underlying 1D domain decomposition (cylindrical Sod's shock tube problem) (b) AMR hydrodynamic grind times. Up to 30 threads per process, gains from cache effects are clearly noticeable.

Using several processors, the efficiency of the strategy is illustrated Table 1 by speedup results for a 2D/AMR nonlinear electronic conduction problem, using an implicit scheme and a conjugate gradient method with diagonal preconditionning (Fig. 7). With an overloading factor of 8 multithreaded subdomains per process and the same type of domain decomposition given Fig. 6, a speedup of about 14.4 on 16 processors is achieved, to be compared to 7 in pure MPI mode.

In practice, diminishing grain size is usually not sufficient to achieve good speedups.

Dynamic load balancing has also to be done, performed automatically by the AMR platform, currently by subdomain migrations using the standard check-point/restart procedure.

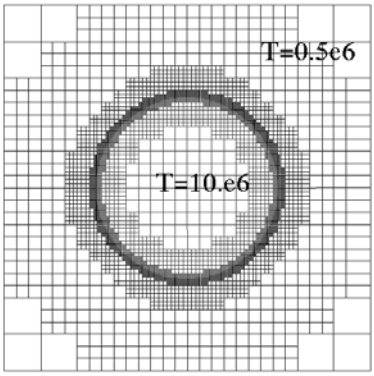


Fig. 7. Nonlinear electronic conduction. Structure of the four-level 3x3 AMR grid.

Table 1. Nonlinear electronic conduction (600,000 cells, four-level 3x3 AMR).

mode	threads/processor	processors	speedup	total time
serial	1	1		1556.
MPI	1	16	7.04	221.
MT/MPI	2	16	10.24	152.
MT/MPI	4	16	10.81	144.
MT/MPI	8	16	14.41	108.
MT/MPI	16	16	11.04	141.

As final example, a speedup curve for the Top Hat #1 benchmark [8] – a radiation hydrodynamic problem treated here with an equilibrium diffusion model – is reported Fig. 8. For a converged Eulerian computation, the reference 4 millions cells uniform grid simulation costs 3,330 CPU hours (128x20 hours) for 10 microseconds of physical time. The AMR computation, offering the same resolution on the Marshak wave and on the interface, just requires 50 CPU hours in monoprocessor mode, with virtually the same temperature profile, exhibiting a spectacular AMR efficiency of about 70.

In parallel mode, with dynamic load balancing & subdomain migrations, the wall clock time is reduced to 3 hours 10 minutes, a 15.6 speedup on 16 processors. With more than 16 processors, the average number of cells per multithreaded subdomain gets to low (3,000 cells) for the cost of extra-communications.

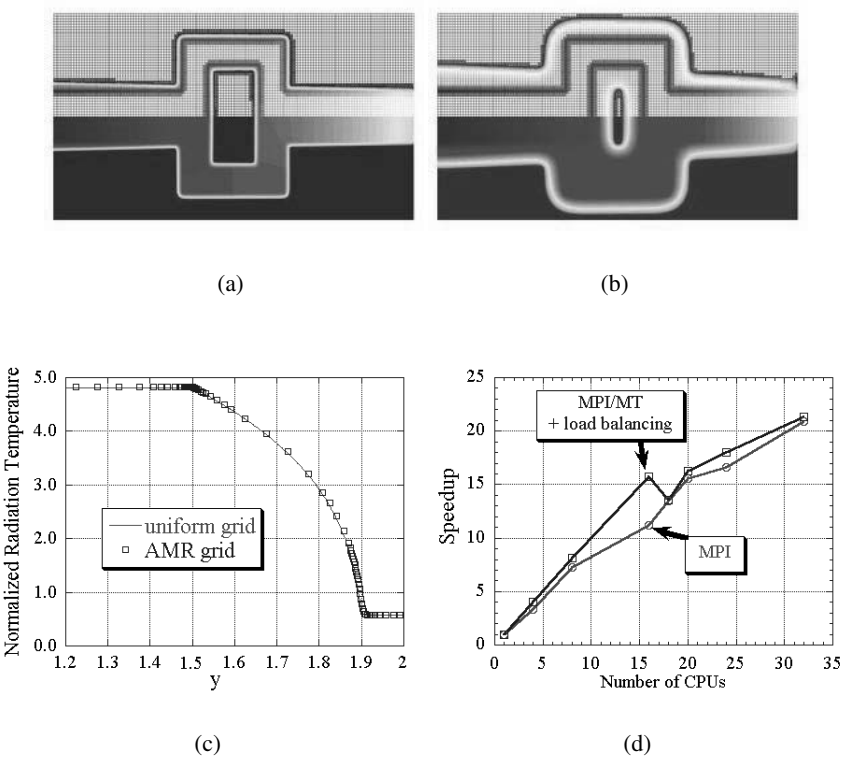


Fig. 8. Top Hat #1 benchmark, constant opacitites. Equilibrium diffusion model at 2 microseconds (a) and 10 microseconds (b). Temperature profiles at 10 microseconds (c). Speedup curve with MPI and MPI/multithread parallel modes, using dynamic load balancing and subdomain migrations (d).

4 User-model API

The C++ programming language, selected for the development of the AMR parallel kernel code, evolved favorably over the last few years, with compilers offering a much better support of the full norm and shorter compilation times [5].

Consequently, advanced C++ features are also used now within AMR solvers and physical modules, most notably with the introduction of *templates* for all variables located on cells and faces ('global' or per material). This insures a complete *source-code decoupling* of AMR solvers and physical modules from memory management, an issue handled dynamically at runtime by the AMR kernel, allowing for example dynamic switches on a per-material basis from direct addressing to compressed

[5] It's also well-established now that C++ may compete favorably with Fortran in terms of CPU performance.

memory-chunks or vice versa, depending upon the current number of cells of each material.

Full feature objet-oriented APIs are also provided:

i) in C++, AMR solvers and user-models being *objects* loaded at runtime or statically linked to the kernel[6], vector syntax being eventually available using PETE Portable Expression Template Engine;

ii) or in Python vector mode, for user-models defined within data sets and interpreted at runtime[7], implemented via SWIG wrapping, the underlying AMR C++ vectors being the same "templatized" vectors as in PETE compiled mode.

The two APIs are quite close, with the same entry points to the AMR solvers. An example of extension of the platform in combustion using Python is provided Listing 1 with full source code, data set and associated gaphical output Fig. 9.

In Python vector mode, such local models are 15-20 % slower than in compiled C++.

Introduction of just in time compiling techniques is on the way, along with comparisons to C# mechanisms, to determine whether AMR solvers may also be implemented that way, in interpreted vector mode on the top of the AMR C++ parallel kernel.

Such an issue may be relevant for future use of the platform by numericists.

```
#! /usr/bin/env python
#
#   HERA - V0.1  CEA 2003 (C)
#
#   Example of reactive flow extension via Python user-model
#

pythonExtensionModule = """from heraAPI import *

class MyModel (HydroAPI):

   def CreeGdrInt(self):
         self.createVariable("burn", self.VAR_SORTIE
          | self.VAR_INTENSIVE | self.VAR_PROJETABLE_MASSE)

   def EqEtatInit(self):  #  EOS at initialisation
         rho = self.var("rho"); pmat = self.var("pmat")
         eint = self.var("eint"); cmat2= self.var("cmat")
         burn = self.var("burn"); burn.assign(0. * burn)

         gamma, pzero = 3.8060, 28.445e8

         eint.assign((pmat+gamma*pzero) / ((gamma- 1.)*rho))
         cmat2.assign(gamma * (pmat + pzero) / rho)

   def EqEtat(self):     #  EOS at each time step
         rho = self.var("rho"); eint = self.var("eint")
         pmat = self.var("pmat"); cmat2= self.var("cmat")

         # MIXTURE STIFFENED-GAS EOS

           gamma1, pzero1 = 3., 0.
           gamma2, pzero2 = 3.8060, 28.445e8
           dcj = 8600.;  q = dcj**2/(2.*(gamma1**2-1.))

           burn  = self.var("burn")
           gamma = self.temporaryVariable()
           pzero = self.temporaryVariable()

           gamma.assign(1. + (gamma1 - 1.) * (gamma2 - 1.)
            /((burn*(gamma2-1.))+((1.- burn)*(gamma1-1.))))

           pzero.assign(((gamma - 1.) / gamma)
            * (burn * (gamma1 * pzero1) / (gamma1 - 1.)
            + (1.-burn)*(gamma2*pzero2) / (gamma2 - 1.)))

         #  EOS OUTPUT

           pmat.assign((gamma-1.)*rho*(eint+burn*q)-gamma*pzero)
           cmat2.assign(gamma * (pmat + pzero) / rho)

   def Calcul(self):

         # BURN FRACTION INCREMENTATION - FOREST FIRE MODEL

           pmat = self.var("pmat"); burn = self.var("burn")

           pref,pmin,a,b,n = 35.e8,1.e8,0.07e6,4.2e-5,2.85
```

[6]Different versions of the platform can be easily administrated, with solvers and physical modules linked statically to the kernel or available via dynamic libraries.

[7]In Python mode, no user-access to the 'include' files of the platform is necessary.

```
        p = self.temporaryVariable()
        p = maximum(1.e-99, pmat)

        burn.assign(burn + where ( pmat <= pmin , 0. ,
          self.deltat()*((1.-minimum(burn, 1.)) *a
         *pow(p/pref,n) * (1.+b*pow(p/pref,7. - n)))));

        burn.assign(minimum(1., burn))

herausermodel_HE = MyModel()

"""

import hera

mat1 = hera.Mat("HE", ext="PythonUserModel"
                    , amr_ini=2, amr_min=2)
mat1.ini (rho = 1844, pmat = 1.0e5)
mat1.frt ( (0.21, 0.0), (0.21, 0.1), (0.9, 0.1), (0.9, 0.0) )

mat2 = hera.Mat("Imp", gamma=3.5, pzero=35.e9
                     , amr_ini=3, amr_min=3, amr_max=3)
mat2.ini (vitesse = 1.0e3, rho = 7882, pmat = 1.0e5);
mat2.frt ( (0.2, 0.0), (0.21, 0.0), (0.21, 0.1), (0.2, 0.1) )

mat3 = hera.Mat("Conf", gamma=3.5, pzero=350.e8
                                , amr_ini=3, amr_min=3, amr_max=3)

mat3.ini (rho = 7882, pmat = 1.0e5);
mat3.frt ( (0.2, 0.1), (0.9, 0.1), (0.9, 0.), (0.91, 0.)
                               , (0.91, 0.11), (0.2, 0.11) )

s1 = hera.Solver("Godunov", lag=2, advect=2 , limiter="superbee"
                             , amr_crit = "shlieren")

NX =  11; NY =  4

for i in range(3):

        amrData = hera.Data (dim = 2, geom = 1
           , xmin = 0. , ymin = 0., xmax = 1.1, ymax = 0.40
           , nx = NX * 2**i, ny = NY * 2**i
           , amr_fact = 3, amr_max = 4, amr_extra=1, amr_frt=0
           , amr_plus = 0.10, amr_minus = 0.060
           , tfinal = 90.e-6, tmpost = "0. 5.e- 6/100.e-6 10.e-6"
           , dt_ini = 4.e-8
          )

        simu = hera.Simu ("ForestFire"+str(i), amrData, (s1,)
             , (mat1, mat2, mat3), ext = pythonExtensionModule)

        simu.run()
```

Listing 1. A complete Python user-model with AMR data set (mixture EOS and shock to detonation reaction rate).

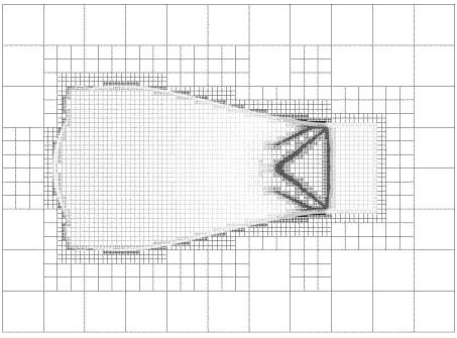


Fig. 9. Graphical output corresponding to the Python data set.

5 Conclusion

A long-going effort has been devoted at CEA/DAM to the evaluation of multifluid AMR hydrodynamics, on a wide range of unsteady fluid flow problems. Patch-based AMR being essentially seen as a natural extension of current Eulerian hydrocodes on MPPs, a cell-by-cell approach has been considered. The evaluation revealed successful on cartesian grids, with impressive CPU time reductions when compared to uniform grids, as illustrated in this paper by several examples. Interestingly, the choice of a cell-by-cell approach also led to various advances, modifying the traditional view one may have of hydrocode development. Staggered grid hydrodynamic

schemes being especially difficult to implement, most notably with arbitrary refinement factors, cell-centered Godunov-type schemes have been considered. In a truly multi-physics perspective, this led to the development of a mathematical ADI framework to cover consistently hyperbolic systems as hydrodynamics, hypo-elasticity, nT hydro and nT MHD. On the computer programming side, cell-by-cell AMR being also quite difficult to parallelize, a new type of parallel code architecture has been explored, using advanced C++ techniques. The choice of such a language, and related development tools, now opens the way for new types of interaction with final users, that may be able to develop code extensions by their own. A version of the AMR platform has been delivered recently to physicists for research work in the field of Laser Plasma Interaction.

6 Acknowlegments

P. Ballereau, D. Dureau, B. Despres, M. Khelifi and S. Jaouen, all associated to this project, are gratefully acknowledged. D. Besnard and D. Bouche are also thanked for their early support.

References

1. J. Von Neumann, R. D. Richtmyer, J. Appl. Phys., 21, p. 232 (1950)
2. S. K. Godunov, Mat. Sb., 47, p. 271 (1959)
3. B. L. Gerasimov, S. A. Semushin, Lecture notes in Physics, V. 170, p. 211 (1982)
4. D. Youngs, Internal report AWRE/44/92/35
5. M. L. Gittings, "SAIC's adaptive grid Eulerian hydrocode". In Defense Nuclear Agency Numerical Methods Symposium, 28-30 April 1992.
6. F. Lagoutire, PhD Thesis, Universit Paris VI, (2000)
7. B. Despres, "Invariance properties of Lagrangian systems of conservation laws, approximate Riemann solvers and the entropy condition", Numer. Math. (2001) 89, p. 99
8. Gentile N. A., Journ. Comp. Phys., 172 (2001), p. 543-571

Parallel Multi-dimensional and Multi-material Eulerian Staggered Mesh Schemes using Localised Patched Based Adaptive Mesh Refinement (AMR) for Strong Shock Wave Phenomena.

A.S. Dawes[1]

Computational Physics Group, AWE plc, Aldermaston, RG7 4PR, UK
Alan.Dawes@awe.co.uk

1 Introduction

We are interested in problems containing several different materials that become highly distorted, after some impulse, with time. Numerical simulation of these phenomena requires sophisticated numerical techniques and a high level of computing power. Currently this takes the form of a 3TFlop IBM SP2 MPP (massive parallel processing) machine known as BLUEOAK. For applications containing High Explosive (H.E.) sub-millimetre zoning will be required in and around the detonation front. However, it would be prohibitively computationally expensive to have it everywhere. To circumvent this an Adaptive Mesh Refinement (AMR) capability has been developed to dynamically localise the mesh where greater resolution is required.

In an earlier paper[3] we presented an overview of the development of the respective two and three dimensional Eulerian multi-material codes SHAMROCK and HYDRA. In this paper we discuss the parallelisation based upon the in-house parallel/communications library TYPHON. Finally, some applications are presented.

2 Governing Equations

For multi-material flows we make the assumption that each material has a clear interface between its neighbours. Each material is assumed to be fluid like (and inviscid) but may have material strength. We also assume that there is no heat conduction between the adjacent materials. The applicability of these assumptions are discussed further within the introduction to the paper by the author[2].

The conservation equations for mass, momentum and energy of a given material component within the flow are,

$$\frac{d\rho}{dt} + \rho \nabla \cdot \underline{u} = 0 \tag{1}$$

$$\rho \frac{du}{dt} = \nabla \cdot \underline{\underline{\sigma}} \tag{2}$$

$$\rho \frac{d\varepsilon}{dt} = -(\underline{\underline{\sigma}} \cdot \nabla) \cdot \underline{u} \tag{3}$$

where ρ is the density, $\underline{u}$ is the velocity vector, ε is the specific internal energy and $\underline{\underline{\sigma}}$ is stress tensor. The stress tensor can be divided up into a sum of the hydrostatic pressure p and deviatoric stress tensor thus $\underline{\underline{\sigma}} = -p\underline{\underline{i}} + \underline{\underline{s}}$ where $\underline{\underline{i}}$ is the unit tensor. For closure an Equation of State (EoS) for the pressure in terms of the other thermodynamic variables (usually ρ and ε) must be supplied as well as differential equations for the incremental changes in the deviatoric components. A good background to these topics can be found in the article by Benson[1] and the book by Wilkins[12].

3 Numerical Method

The numerical method is second order in space and time and is based on dual staggered and non-staggered Cartesian grids. Density, specific internal energy, stress tensor components and pressure are stored at cell centres, while the velocity components are stored at the vertices. The method consists of two phases. The first is a Lagrangian predictor/corrector step producing an intermediate solution. The second phase remaps this solution back onto the original fixed Eulerian mesh employing operator splitting together with Van Leer's second order (and a "third" order) monotone technique. A full description of this approach can be found in the paper by Youngs[13]. For multi-materials we employ the volume-of-fluid (VOF) technique, and in two dimensions, is also discussed in the paper by Youngs. A comprehensive review of the VOF technique can be found in the paper by Rider and Kothe[9]. The extension to three dimensions is complicated by the fact that the material interfaces have to be reconstructed from the volume fractions during the remap stage. For planar surfaces these can be broken down into five simple shapes and are illustrated in Figure 1.

We have adopted Quirk's patched based philosophy[8] for adaptive mesh refinement. Patches are intuitively easier to implement, than cell-by-cell, for multi-physics models.

4 Parallelisation

Parallelisation was based upon the Single Program Multiple Data (SPMD) model[4] [5]. A single copy of the hydrocode program is executed on every processor with the mesh domain decomposed across the processors. This latter stage has been accomplished using the TYPHON library[11].

Using the SPMD model each processor acts in isolation (from all others) until data is required from its neighbours. This then takes the form of a communication. The library was developed for HYDRA to simplify these tasks and to abstract the "common" parallel/communication events into subroutines or functions. To use the library a communication phase must be first registered. Once this is done the respective variables that will be communicated/exchanged are registered. A typical code fragment for both these events are illustrated in Figure 2.

The TYPHON library was originally developed for uniform three dimensional meshes. However, the success of the SHAMROCK code has brought about its redevelopment. To bridge the gap between this library and the hydrocodes Jones et al[7] have come up with the concept Generic, Adaptive and Parallel (GAP). Essentially, for parallel AMR there are generic functions or events which are common to the codes. For example, the flagging of cells, for sub-division, is generic. The subsequent subdivision to create the patch distribution is also generic. The replacement library TYPHON GTI (GAP Technology Included) is under development.

5 Results and Discussion

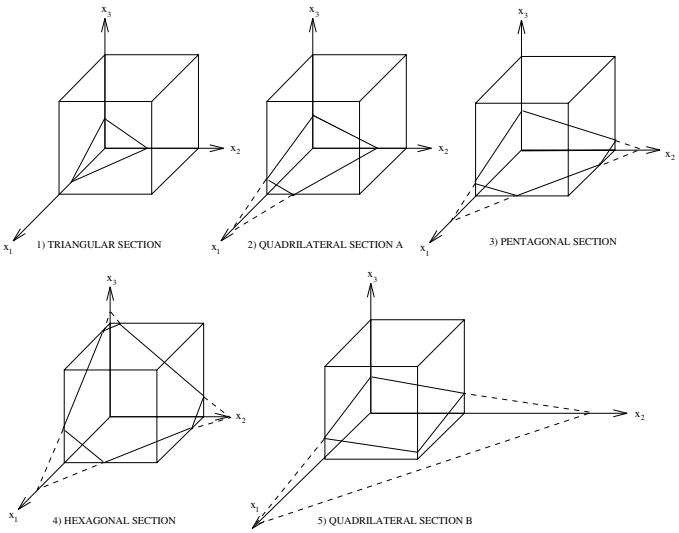


Fig. 1. Interface intersecting Cell.

5.1 Sod Shock Tube Problem

Sod's shock tube problem[10] consists of a high pressure/density region and a low pressure and density region separated by a diaphragm. It is broken resulting in a

```
Communication Phase
CALL REGISTER_COMM_PHASE("Half Time Step",      &
                         ONE_GHOST,             &
                         XYZ_X_DIR,             &
                         PURE + MIXED,          &
                         HALF_TIME_STEP)

Quantity
CALL REGISTER_VARIABLE("Cell Centred Pressure", &
                       PURE + MIXED,            &
                       CENTRED,                 &
                       ONE_GHOST,               &
                       HALF_TIME_STEP,          &
                       RDATA=cell%pressure)
```

Fig. 2. TYPHON Registration Phase.

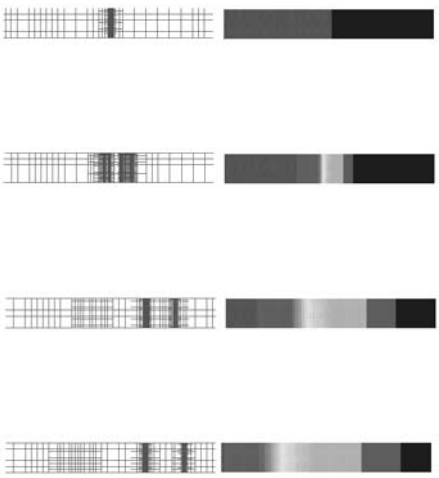


Fig. 3. Sod Shock Tube Problem. Dynamically changing mesh (left) and Pressure Results (right).

shock wave propagating into the low pressure region, a rarefaction propagating into the high pressure region and both are separated by a contact wave. The calculation was performed using SHAMROCK and results are shown in Figure 3. It is observed that the code correctly adapts around the shock wave, contact discontinuity and rarefaction; as the rarefaction weakens so the mesh de-refines around it.

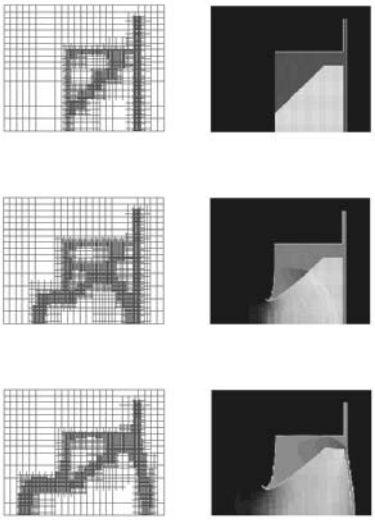


Fig. 4. Plane Wave Generator. Dynamically changing mesh (left) and Pressure Results (right).

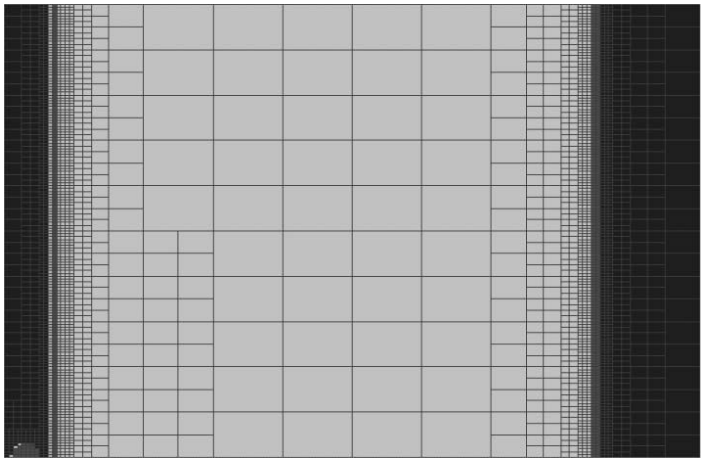


Fig. 5. Particle and AMR before impact.

5.2 Plane Wave Generator

This problem consists of a HE insert in a aluminium case and detonated at one point in the centre of the rear face. The diverging detonation wave propagates outwards and hits a varying thickness barrier which is used to non-uniformly attenuate it to produce a planar propagating wave. The calculation was performed using SHAMROCK and results are shown in Figure 4. The calculation correctly follows the advancement of the shock wave and progression of the material interfaces.

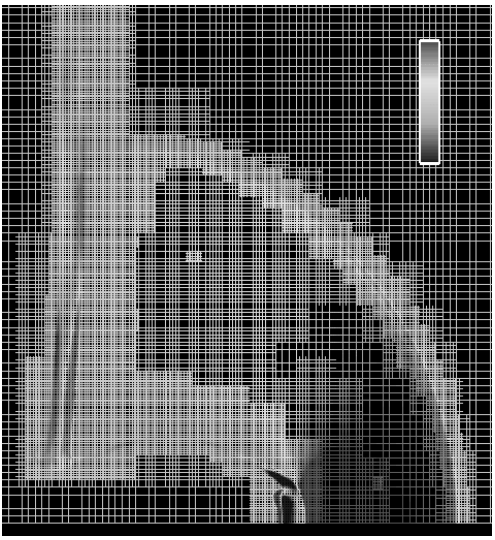


Fig. 6. Particle, AMR and Density Contours after Impact.

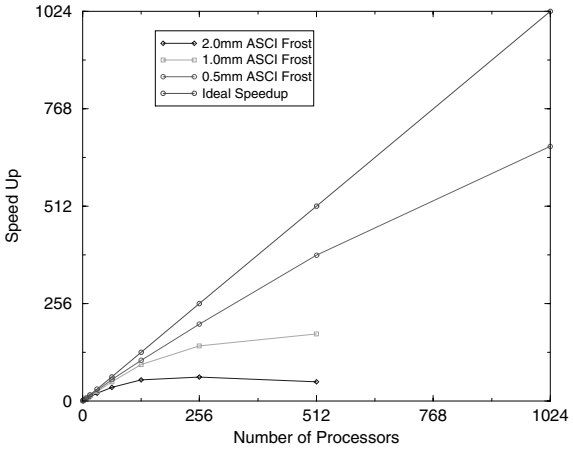


Fig. 7. Scalability Results from ASCI Frost.

5.3 Hydrodynamic Ram

This problem consists of a 12mm diameter steel sphere travelling at 2km/s. The sphere hits a water tank made of 3.2mm thick Aluminium plate. The particle and initial SHAMROCK patch distribution is illustrated in Figure 5. The particle subsequently penetrates the plate and into the water. In Figure 6 the particle, AMR distribution and density are illustrated. The bow shock wave ahead of the distorted particle is clearly visible.

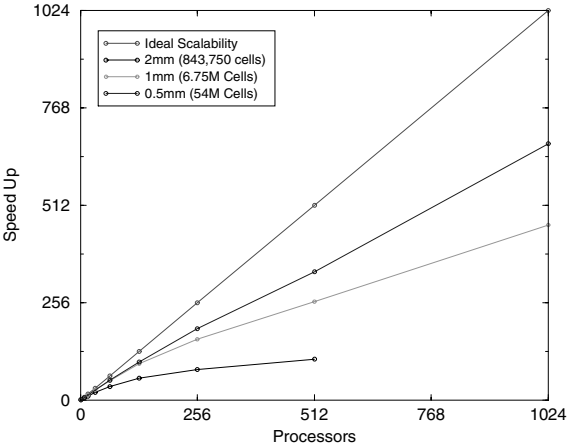


Fig. 8. Scalability Results from BlueOak.

5.4 Freund Problem

This problem consists of a 3mm spherical shell of Aluminium with inner radius of 5.7 cm surrounded by 3cm thick H.E. with inner radius 6cm. The HE is detonated on the axis on its outer surface. The HE compresses the metal shell with the outer surface expanding outward. Further details of the experimental setup can be found in the reference by Freund et al[6]. A uniform mesh calculation was run on HYDRA using different numbers of processors and at different mesh resolutions to show the scalability of the code. Results are presented for runs performed on LLNL [1] ASCI[2] Frost machine (during 2001) in Figure 7. Results are also presented for runs performed on AWE's BLUEOAK machine (during 2003) in Figure 8. It is observed that the scalabilities are very similar between the two machines. This is because ASCI Frost and BLUEOAK are based on similar hardware. At the time of writing this article the 3D AMR option for HYDRA was under development.

6 Acknowledgements

I would like to thank J.A.Herdman (TYPHON Team Leader), Matt Street (TYPHON GTI) and W.P. Gaudin (SHAMROCK Team Leader) for their input into this paper.

References

1. D.J. Benson. Computational Methods in Lagrangian and Eulerian Hydrocodes. *Computer Methods in Applied Machanics and Engineering*, pages 235–394, 1992.

[1] Lawrence Livermore National Laboratory.
[2] Accelerated Strategic Computing Initiative.

2. A.S. Dawes. A Three Dimensional Contact Algorithm for Sliding Surfaces. *Int. J. Numer. Meth. Fluids, Vol 42*, pages 1189–1210, 2003.
3. A.S. Dawes, W.P. Gaudin, and B.J. Parker. Eulerian Hydrocode Development at AWE. *22nd International Symposium on Shock Waves, Imperial College, London, UK*, 1999.
4. K. Dowd and C. Severance. *High Performance Computing*. O'Reilly, 1998.
5. I. Foster. *Designing and Building Parallel Programs*. Addison-Wesley Publishing Company, 1995.
6. H.U. Freund, W. Geiger, and G. Honcia. Acceleration of spherical metal shells by high explosive: Detonation velocity along shell surface and efficiency of energy transfer. *6th International Symposium on Detonation,*, 1976.
7. B. Jones, D.M. Turland, and W.P. Gaudin, 1997. Unpublished AWE Report.
8. J. Quirk. An Adaptive Grid Algorithm for Computational Shock Hydrodynamics, 1991. PhD thesis, College of Aeronautics, Cranfield University, Bedford, UK.
9. W.J. Rider and D.B. Kothe. Reconstructing Volume Tracking. *J. Comp. Phys.***141**, pages 112–152, 1998.
10. G. A. Sod. A numerical study of a converging cylindrical shock. *J. Fluid Mech.,* **83** , pages 785–794, 1977.
11. D.M. Turland. TYPHON: A Parallel Communications Library for 3-D Staggered Cartesian Mesh Hydrocodes, 1997. Unpublished AWE Report.
12. M.L. Wilkins. *Computer Simulation of Dynamic Phenomena*. Springer, 1999.
13. D.L. Youngs. Time-dependent multi-material flow with large fluid distortions. in *Numerical Methods for Fluid Methods*, edited by K.W.Morton and M.J.Baines (Academic Press, New York), 1982.

A general adaptive multi-resolution approach to ocean modelling: experiments in a primitive equation model of the north Atlantic

Laurent Debreu[1], Eric Blayo[1], and Bernard Barnier[2]

[1] Projet IDOPT, Laboratoire de Modélisation et Calcul, 38041 Grenoble Cedex 09, France, {Laurent.Debreu,Eric.Blayo}@imag.fr
[2] Laboratoire des Ecoulements Géophysiques et Industriels, 38041 Grenoble Cedex 09, France, Bernard.Barnier@hmg.inpg.fr

Summary. Following the work of [BD99], the present paper presents a general approach to adaptive mesh refinement for ocean models. The numerical procedure is briefly described, as well as a software package which easily allows for the actual implementation of multi-resolution within any existing finite difference model. The effectiveness of this approach for ocean modelling, even at a basin-scale, is illustrated in the context of a primitive equation numerical model of the north Atlantic. Several experiments are presented, which demonstrate the potentialities of mesh refinement and emphasize the role of the refinement criterion.

1 Introduction

The need for a detailed description and understanding of local phenomena gives rise to increasing interest of ocean modellers for local refinement techniques. As a matter of fact, it is often impossible, due to computer limitations, to define an uniformly very high resolution in a global (or basin-scale) ocean general circulation model (OGCM). On the other hand, a high resolution local model is extremely dependent of the specification of the boundary conditions, which are often only very crudely known. A convenient and frequently used way to overcome these difficulties is the nested grid approach, in which a high-resolution local model covering the area of interest is embedded within a regional or global lower resolution model. The interaction between the two models can be either one-way or two-way. In the first case, the global model solution is interpolated to provide boundary conditions for the local model, but there is no retroaction of the local solution onto the global one. At the opposite, in the two-way case, the local solution is used to update (and hopefully improve) the global solution.

In the context of ocean modelling, a number of studies have already used the nested grid techniques. [SH91] developed a nested grid system and validated it on two analytic problems: the barotropic modon and an anticyclonic vortex. These

two test cases were also treated by [LAM96] to compare the performances of different nesting methods. Nested models have also been used for realistic regional oceanic studies : [OC92] for the Norwegian coastal current, [FM95], [FM96] for the Island-Faroe front; [PSBHW97] and [PS98] illustrated numerical and physical considerations about nested boundary conditions in models of the Greenland-Iceland-Norwegian sea and of the Mediterranean sea; [GHMY03] investigated the role of horizontal resolution on JEBAR in a nested simulation of the Kuroshio, [GRR98)] developed a multiply-nested ocean circulation model to study the response of the ocean to a westerly wind burst and to a tropical cyclone.

While the fine grids were always static in these studies, the notion of mesh movement appeared recently in the context of ocean modelling [BD99] [RG99]. In order to be able to follow an evolving oceanic feature, [RG99] implemented a mesh movement scheme within their multiply-nested ocean model. The mesh movement can be specified a priori or determined in the course of the model run. It has been successfully applied to several idealized and realistic test cases.

In a previous paper ([BD99], henceforth denoted BD99), we addressed the interest for ocean modelling of the general adaptive mesh refinement (AMR) algorithm introduced by [BO84]. In this method, the number, the size and the resolution of the fine grids can evolve during the run according to any specified criterion. Moreover, our implementation is independent of the model, which allow our tools to be easily implemented within any existing finite-difference model. The effectiveness of the method was demonstrated in BD99 in the classical case of the barotropic modon, and for a multilayer quasigeostrophic box model. Following this work, the aim of the present paper is to show that these previous results can be extended to realistic OGCMs for basin-scale simulations, by describing an application to a primitive equation model of the north Atlantic. The paper is organized as follows: the AMR method and the way we implement it are briefly recalled in section 2, and the OGCM and its north Atlantic configuration are presented in section 3. Section 4 is devoted to simulations with adaptive refinement, and emphasizes the peculiar importance of the choice of the refinement criteria for the simulation of the general circulation. Finally some conclusions are drawn in section 5.

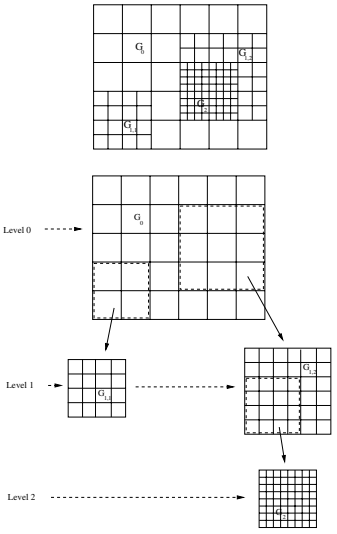
2 The AMR method and its implementation

2.1 General description

Our work is based on the AMR method proposed by [BO84] for the solution of partial differential equations using finite difference techniques. A complete description of this method can be found in this latter paper, in [BC89], or in BD99, and we will only give here a brief overview.

The AMR strategy features a hierarchy of embedded grids, with different levels of resolutions, and which interactions are specified by a time-integration scheme. More precisely, the level 0 contains one only coarse grid G_0, of resolution h_0, covering the whole domain. This root grid contains a set of n_1 finer grids $G_{1,i}$ $(i = 1, \ldots, n_1)$

of resolution h_1, forming the level 1. This nesting is recursive, each grid at level l containing eventually some finer grids at level $l+1$. A maximum number of levels L must of course be defined by the user. The refinement ratio h_l/h_{l+1} is a constant integer r (typically between 2 and 5). An example of grid hierarchy and the associatesd time integration scheme are symbolized on Fig. 1.



Recursive Procedure INTEGRATE (l)

If l == 0 **Then** *nbstep* = 1
Else *nbstep* = *refinement-ratio*
Endif
Repeat *nbstep* times
  Step on all grids at level l
  If level l+1 exists **Then**
    Compute boundary conditions at level l+1
    INTEGRATE$(l+1)$
    update level l
  Endif
End Repeat
End Procedure INTEGRATE

Fig. 1. An example of grid hierarchy and the time integration scheme

We will note k_l the time step on grids at level l. A usual way to define k_l is to impose $k_l/k_{l+1} = r$, which ensures that the CFL number h_l/k_l is constant and does not depend on the resolution level. To advance the whole grid hierarchy by one coarse time step k_0 (from time T to time $T+k_0$), we begin by advancing the coarse grid by one time step k_0. Then boundary conditions are computed for grids at level 1 at time steps $T, T+k_1, \ldots, T+rk_1 = T+k_0$. This is usually done by some interpolation in time and space of the coarse grid solutions at time T and $T+k_0$. Given these boundary conditions, the grids at level 1 can then be advanced by r time steps k_1. This procedure is of course recursive, and grids at finer resolution levels are advanced in the same manner. For every resolution level l except the finer one, the solutions on the different grids at the end of each time step are updated (and hopefully improved) by using the solutions on the embedded grids at level $l+1$ at the same physical time. This fine-to-coarse grid transfer of information can be performed in several ways (simple copy of corresponding values, averaging or other filtering operators...) and can take into account different constraints (e.g. conservation of mass across coarse/fine grid interfaces).

The number, the size and the location of the different grids can either be fixed or evolve during the simulation. In this last case, a criterion is applied at regular time intervals (every N coarse time steps k_0) at each grid point to detect if a finer grid is necessary, or if an existing fine grid can be removed. Such a criterion can rely on a mathematical basis (some estimate of the truncation error for instance), on physical basis (e.g. some turbulence indicator), or on a combination of both. Since this criterion detects only a set of points, an algorithm must then be applied to cluster these points into rectangular subgrids. An efficient regridding procedure is described for example in [BR91].

2.2 Implementation

In order to implement efficiently and easily this method into any existing finite difference ocean circulation model, we have developed the Fortran-90 software package AGRIF (Adaptive Grid Refinement In Fortran), available to the scientific community. In our approach, the model is seen as a black box. The package must only know some of its features (e.g. the name of the global variables, their position in the cells...), which are described by the user in a configuration file. The user also indicates in this file some AMR parameters (e.g. the refinement ratio r, the maximum number of resolution levels...) and provides the refinement criterion if any. A complete description of this package is given in [DB02]. AGRIF is already used in different fixed grid refinement applications: in the ROMS model for high resolution modelling of the Los Angeles Bay currents, or in the OPA model for simulation at 1/15 ° resolution of the Labrador sea.

3 Configuration of the numerical experiments

3.1 Model description

The numerical simulations which are presented in section 4 were performed with the OPA OGCM [MDIL99]. This model solves the primitive equations on the sphere, with the rigid-lid hypothesis. It uses horizontal curvilinear coordinates and vertical z-coordinate. The equations are discretized using standard second-order centered finite difference schemes, on an Arakawa C-grid. Discretization in time uses a leapfrog scheme for non-diffusive terms, and an Euler scheme for diffusive ones. Subgrid scale processes are parameterized both horizontally and vertically by a laplacian diffusion operator. For the vertical part, the values of the coefficients are computed at each time step by a turbulent closure model (TKE). A no-flux condition is imposed for temperature and salinity at the ocean bottom, and a slip condition is chosen for the velocity at the solid boundaries.

3.2 AMR implementation

We have implemented in this model the capability for mesh refinement and mesh adaptivity using the AGRIF package mentioned previously. In the present applications, the refinement ratio between successive resolution levels is chosen equal to 3.

Boundary conditions for embedded grids are linearly interpolated in time and space. The variables on the different grids at the end of each time step are updated with values of corresponding variables on finer grids, using a classical 'full-weighted' filter. Moreover, a correction step is added to ensure conservation of mass through grid interfaces. Some specific difficulties were encountered, due to the fact that we address realistic applications. We can cite for instance the management of islands, or of irregular coastlines and bottom topography at different resolutions. This last point can lead for example to different values for the volume of the ocean, depending on the resolution levels, or to grid points being located on the boundary for a particular resolution level, but one mesh inside or outside the domain for another level. Some conventions and specific near-boundary schemes were included in the package to deal with those difficulties.

In the present study, several refinement criteria were tested and compared. Since the ocean circulation is essentially horizontal, we have defined two-dimensional criteria $R(x,y)$, the points corresponding to the highest values of $R(x,y)$ being detected for refinement. In the present implementation of the method, the refinement itself is also only horizontal (the number of vertical levels remains the same on every grid).

Let $V = (u,v)$ a 2D velocity field. We can define the following quantities :

- $Q_1(V) = \|V\| = \sqrt{u^2+v^2}$ (Euclidean norm of V)
- $Q_2(V) = \|\mathrm{Curl}V\| = \left|\dfrac{\partial v}{\partial x} - \dfrac{\partial u}{\partial y}\right|$ (absolute value of the relative vorticity)
- $Q_3(V) = \dfrac{\partial u}{\partial x}\dfrac{\partial v}{\partial y} - \dfrac{\partial v}{\partial x}\dfrac{\partial u}{\partial y}$. This expression is the restriction to the 2D incompressible case of the more general 3D quantity $\dfrac{1}{2}\sum_{i,j}\left(A_{ij}^2 - S_{ij}^2\right)$, where A_{ij} and S_{ij} are respectively the antisymmetric and symmetric parts of the deformation tensor :

$$A_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) \qquad S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

for any velocity field u. This quantity, used in the context of turbulence studies [HWM88] [Del99], can be seen as a measure of the balance between vorticity and shearing.

In our numerical experiments, we have compared three refinement criteria : $R_1(x,y) = Q_1(V(x,y))$, $R_2(x,y) = Q_2(V(x,y))$ and $R_3(x,y) = \max(0, Q_3(V(x,y)))$, with V being a vertical average of the velocity field on the upper levels of the model (corresponding to the first 1000 meters).

Fine grids initialization

When a new grid is created, the 3D velocities and the 2D barotropic streamfunction are interpolated independently. So there is no reason for the resulting barotropic

transport to be non divergent ($\mathrm{div} H\bar{U}_h = 0$) as it is required by the rigid lid approximation. A correction step must then be added.
A first solution can be that the barotropic velocities are replaced using the streamfunction Ψ. The corrected velocities $(u^\star, v^\star)$ are given by :

$$\begin{cases} u^\star = u - \dfrac{1}{H}\dfrac{\partial \Psi}{\partial y} - \dfrac{1}{H}\displaystyle\int_{-H}^{0} u \\ v^\star = v + \dfrac{1}{H}\dfrac{\partial \Psi}{\partial x} - \dfrac{1}{H}\displaystyle\int_{-H}^{0} v \end{cases} \tag{1}$$

In this solution, the whole correction acts on the barotropic part of the velocity. Thus this can lead to large variations $|U - U^\star|$, which destabilizes the algorithm in actual simulations.

In this study, we have chosen an algorithm that also modifies the streamfunction. We have to solve the following problem : find three correction terms u', v', Ψ' so that

$$u^\star = u + u', \quad v^\star = v + v', \quad \Psi^\star = \Psi + \Psi'$$

with the condition that

$$\begin{cases} \bar{u}^\star = -\dfrac{1}{H}\dfrac{\partial \Psi^\star}{\partial y} \\ \bar{v}^\star = +\dfrac{1}{H}\dfrac{\partial \Psi^\star}{\partial x} \end{cases}$$

or equivalently

$$\begin{cases} \bar{u} + \bar{u'} = -\dfrac{1}{H}\dfrac{\partial \Psi}{\partial y} - \dfrac{1}{H}\dfrac{\partial \Psi'}{\partial y} \\ \bar{v} + \bar{v'} = +\dfrac{1}{H}\dfrac{\partial \Psi}{\partial x} + \dfrac{1}{H}\dfrac{\partial \Psi'}{\partial x} \end{cases}$$

or

$$\begin{cases} \bar{u'} + \dfrac{1}{H}\dfrac{\partial \Psi'}{\partial y} = -\bar{u} - \dfrac{1}{H}\dfrac{\partial \Psi}{\partial y} \\ \bar{v'} - \dfrac{1}{H}\dfrac{\partial \Psi'}{\partial x} = -\bar{v} + \dfrac{1}{H}\dfrac{\partial \Psi}{\partial x} \end{cases}$$

If we assume that the field $\bar{u'}, \bar{v'}$ has a zero curl (which implies that the barotropic vorticity of the corrected field equals the barotropic vorticity of the interpolated field), we have to solve for Ψ' the following equation

$$\frac{\partial}{\partial x}\left[\frac{1}{H}\frac{\partial \Psi'}{\partial x}\right] + \frac{\partial}{\partial y}\left[\frac{1}{H}\frac{\partial \Psi'}{\partial y}\right] = \frac{\partial \bar{v}}{\partial x} - \frac{\partial \bar{u}}{\partial y}$$

This equation has exactly the same structure that the barotropic vorticity equation already solved in the model and so the same solver can be used. On the boundary of the high resolution domain, we take $\Psi' = 0$. Once this solution is found, the new barotropic streamfunction $\Psi^\star = \Psi + \Psi'$ is computed and we deduce the velocity as in the first solution (eq (1)).

3.3 North Atlantic configuration

The computational domain covers the whole north Atlantic ocean, from 20 °S to 70°N. The grid is isotropic, with a constant mesh size $\Delta\lambda$ in longitude, and a varying mesh size $\Delta\Phi = \Delta\lambda.\cos\Phi$ in latitude. Two buffer zones are implemented at the northern and southern boundaries. In these areas covering 5 ° of latitude, the temperature and salinity are relaxed towards climatological values.

4 Numerical experiments

The resolution of the root coarse grid is equal to 1 ° in longitude, with 20 vertical levels. In the following numerical experiments we used a refinement factor of 3 leading to a resolution of 1/3° on the fine grids. The 1° model corresponds to a non eddy-permitting coarse resolution whereas the 1/3° resolution model is eddy-permitting. Starting from rest, a 10 years spin up is done and the next three years are used for the diagnostics.

A series of four experiments have been conducted, which are summarized on table 1. The first one is a reference run with a uniform resolution of 1/3 ° on the whole domain. The second one is a coarse grid run with a uniform resolution of 1°. During the adaptive run, we chose the refined areas in order to cover a proportion of approximatively 15-20% of the domain. The space and time refinement factors are equal to 3, leading to a computational cost approximatively equal to $(1 + (0,15 - 0,20) \times 3^3) \approx 5 - 6.5$ times the cost of the uniform coarse resolution run. It is interesting to note that the closest run at a uniform usual resolution, in term of computational cost, is a 1/2° model (cost equal to 8 times the coarse model cost). That is why such an additional experiment has been conducted.

Table 1. Computational costs of the different simulations

resolution	cost
uniform 1°	1
adaptive run 1-1/3°	5-6.5
uniform 1/2°	8
uniform 1/3°	27

4.1 Influence of the refinement criterion

In order to compare the different criteria, we conducted a one year experiment (starting from rest). Instantaneous plots of the streamfunction are showned on figure (2). In all these experiments, the detection of areas to be refined is done every four days. When new grids are created, interpolations are done everywhere except at points which were already inside a high resolution grid (in the previous grid hierarchy).

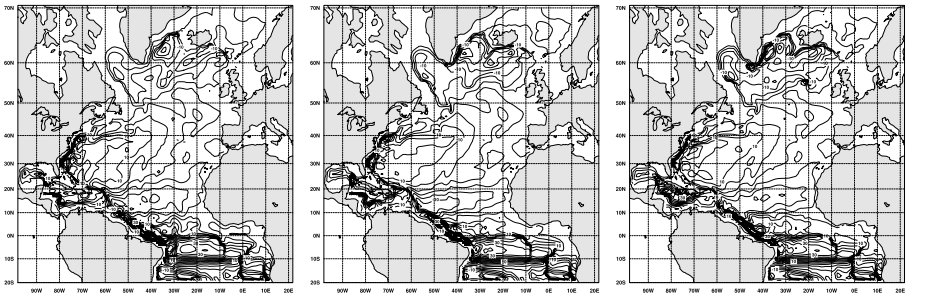


Fig. 2. Instantaneous streamfunction after one year, using different refinement criteria, left: norm of velocity, middle: vorticity, right: Hunt criterion

The values of the different state variables at these points are simply "restored" from the existing ones. An indicator of the grid movement from one refinement stage to another is the percentage of restored points, which is given by table 2.

Table 2. Percentage of restored points

refinement criterion	restored points in percentage
norm of velocity	88%
vorticity	75%
Hunt criterion	80%

It appears that the velocity criterion is the more stable (with actually a small difference in refined areas between winter and summer).
Comparing the Hunt's and the vorticity criteria, differences appear mainly in the north of the domain (subpolar gyre and Greenland-Iceland-Scotland ridge). We believe that the buffer area creates a strong shear that produces vorticity detected by the vorticity criterion but not by the Hunt criterion. The Hunt criterion will be kept for the experiments described next.

4.2 Comparison of the AMR solutions with fixed resolution solutions

For all these experiments, the 1/3° will be our reference run. All results presented here correspond to a three-year average after the ten-year spin up time. The barotropic streamfunction and the temperature field at 100m depth corresponding to the four different simulations are plotted on figures (3) and (4).

Comparing the uniform 1/3° and 1° resolution runs (top left and bottom left pictures), we observed usual differences. Strongest ones appear in the representation of the subpolar gyre both in its transport and its southward extension. The intensification of the Gulf stream is clearly visible on the temperature field with an enhancement of the warm water eastward penetration. Strong equatorial currents are also visible

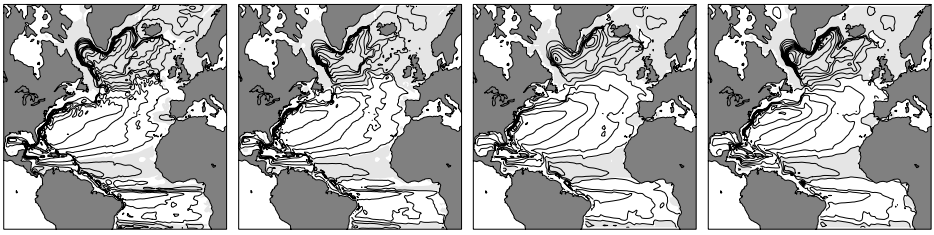


Fig. 3. Barotropic streamfunction : from left to right: uniform 1/3° simulation (reference simulation), uniform 1/2° simulation, uniform uniform 1° simulation, adaptive 1°-1/3° simulation. Contour interval is of 5 Sverdrup

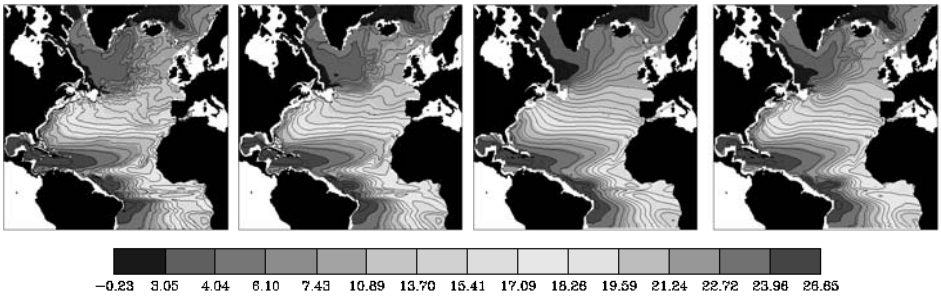


Fig. 4. Temperature at 100 m depth: from left to right: uniform 1/3° simulation (reference simulation), uniform 1/2° simulation, uniform uniform 1° simulation, adaptive 1°-1/3° simulation.

in the 1/3° run on the temperature field. However these currents are essentially baroclinic and have no real impact on the barotropic streamfunction.

If we now compare the two other simulations, it appears that the adaptive simulation is nearer to the 1/3° simulation than the 1/2° simulation is, with an exception at the equator and in the Antilles current. This is however not surprising since we have seen that our criterion nether refines in this area (expect near the west coast). Two different reasons can explain this absence of refinement. First it may be approximatively in the jump from 1° to 1/2° resolution that these equatorial currents appear. Additionnaly, even if these currents were present in the 1° resolution run, as already said these currents are mainly baroclinic and this can explain why our refinement criterion (which is always computed by taking an average on the first 1000m) does not detect this area.

In other areas, the adaptive simulation seems better reproduce the 1/3° reference run than the 1/2° run does: this is clearly visible in the subpolar gyre (where the differences were the most marked between the 1° and the 1/3° simulations) and in the gulf stream region.

5 Summary and conclusions

In this paper, we discussed the application of an adaptive mesh refinement method to ocean modelling. Integration of adaptive mesh refinement capabilities was done with the use of the AGRIF tool [DB02].

To our knowledge, the application of full adaptive mesh refinement applications in the context of ocean modelling has not been studied before. The goal of this paper was to give some first insights in the context of realistic simulations.

The configuration with a 1° resolution is a non eddy-permitting simulation. This is clearly a drawback, especially for the evaluation of the different refinement criteria. However its low cost lets us perform several long term simulations.

The results highlights the importance of the choice of the refinement criterion. The Hunt criterion, a balance between vorticity and shearing, seems to lead to the best refinement location. However its use here was only on vertically averaged currents and so does not take into account the baroclinicity of the currents as may have an important role in the detected area, as for example in equatorial regions.

Comparing the reference run (a $1/3^\circ$uniform resolution run) with the adaptive (AMR) run we saw that for a global cost in CPU time and memory divided by a factor of 4-5, we can achieve quite similar results in the main turbulent areas like the subpolar or subtropical gyres. The comparison AMR - $1/2^\circ$run (which was the closest run for a uniform resolution run) shows that AMR gives in general better results except in regions where refinement never takes place.

Even if this adaptive mesh refinement method seems potentially very attractive, it still requires a lot of improvements. An increase of the root grid resolution is clearly needed, a resolution of $1/3^\circ$ or $1/4^\circ$ should lead to an improvement of the meaning of the refinement criteria, and consequently of the grids location. Among the other possibilities, the vertical refinement is the most promising, allowing both an increased vertical resolution where and when needed and also to refine horizontally only some parts of the water column, leading in an additional gain in CPU.

Acknowledgments

The authors would like to thank Dr Hervé Knochel for his help in the management of the numerical simulations. The calculations were made possible by the computer facilities at the Centre Grenoblois de Calcul Vectoriel (CGCV-CEA) and on the MIRAGE cluster of the University of Grenoble. IDOPT is a joint CNRS/University of Grenoble/Institut National Polytechnique de Grenoble/INRIA project.

References

BO84. Berger, M., Oliger, J.: Adaptive mesh refinement for hyperbolic partial differential equations, J. Comp. Phys., **53**, 484-512 (1984)

BC89. Berger, M., Colella, P.: Local adaptive mesh refinement for shock hydrodynamics, J. Comp. Phys., **82**, 64-84 (1989)

BR91. Berger, M., Rigoutsos, I.: An algorithm for point clustering and grid generation, IEEE Trans. on Systems Man and Cybernetics, **21**, 1278-1286 (1991)

BD99. Blayo, E., Debreu L.: Adaptive mesh refinement for finite difference ocean models: first experiments, J. Phys. Oceanogr., **29**, 1239-1250 (1999)

DB02. Debreu, L., Blayo, E.: AGRIF: Adaptive Grid Refinement in Fortran, INRIA Technical Report, RT-0262, (2002) [Available online http://www.inria.fr/rrrt/rt-0262.html].

Del99. Delcayre, F.: Etude par simulation des grandes échelles d'un écoulement décollé : la marche descendante, PhD thesis, Institut National Polytechnique de Grenoble, (1999).

FM95. Fox, A.D., Maskell, S.J.: Two-way interactive nesting of primitive equation ocean models with topography, J. Phys. Oceanogr., **25**, 2977-2996 (1995)

FM96. Fox, A.D., Maskell S.J.: A nested primitive equation model of the Iceland-Faeroe front, J. Geophys. Res., **101**, 18259-18278 (1996)

GRR98). Ginis, I., Richardson, R., Rothstein, L.M.: Design of a multiply nested primitive equation ocean model, Month. Weath. Rev., **126**, 1054-1079 (1998)

GHMY03. Guo, X., Hukuda, H., Miyazawa, Y., Yamagata, T.: A Triply Nested Ocean Model for Simulating the Kuroshio - Roles of Horizontal Resolution on JEBAR, J. Phys. Oceanogr., **33**, 1, 146-169 2003

HWM88. Hunt, J.C.R., Wray, A.A., Moin, P.: Eddies, stream and convergence zones in turbulent flows, Center for Turbulence Research report, CTR-S88:193. (1998)

LAM96. Laugier, M., Angot, P., Mortier L.: Nested grid methods for an ocean model: a comparative study, Int. J. Num. Meth. Fluids, **23**, 1163-1195 (1996)

MDIL99. Madec, G., Delecluse, P., Imbard, M., Levy, C.: OPA release 8.1, Ocean General Circulation Model reference manual, Internal report, LODYC/IPSL, France (1999)

OC92. Oey, L.-Y., Chen, P.: A nested-grid ocean model with application to the simulation of meanders and eddies in the Norwegian coastal current, J. Geophys. Res., **97**, 20063-20086 (1992)

PSBHW97. Perkins, L., Smedstad, L.F., Blake, D.W., Heburn, G.W., Wallcraft, A.J.: A new nested boundary condition for a primitive equation ocean model, J. Geophys. Res., **102**, 3483-3500 (1997)

PS98. Perkins, L., Smedstad, L.F.: Scale-related aspects of nested finite difference ocean models, Theoret. Comput. Fluid Dyn., **10**, 311-322 (1998)

RG99. Rowley, C., Ginis, I.: Implementation of a mesh movement scheme in a multiply nested ocean model and its application to air-sea interaction studies, Month. Weath. Rev., **127**, 1879-1896 (1999)

SH91. Spall, M.A., Holland, W.R.: A nested primitive equation model for oceanic applications, J. Phys. Oceanogr., **21**, 205-220 (1991)

An Overview of the PARAMESH AMR Software Package and Some of Its Applications

Kevin M. Olson[1] and Peter MacNeice[2]

[1] GEST Center, University of Maryland, NASA/GSFC, *kevin.olson@gsfc.nasa.gov*
[2] Drexel University, NASA/GSFC, *peter.macneice@gsfc.nasa.gov*

Summary. The latest release of the PARAMESH parallel, adaptive mesh refinement software package is discussed. Its features and some of the applications it is being used with are highlighted. Further, we discuss the philosophy of the design of the package as well as some the problems and solutions we have found when importing into various applications.

1 Introduction to PARAMESH

PARAMESH [1] was written primarily by Peter MacNeice and Kevin Olson at NASA's Goddard Space Flight center as part of the NASA/ESTO-CT project (formally HPCC). Other contributors to date include C. Mobary, R. deFainchtein, C. Packer (NASA/GSFC), R. DeVore (NRL), M. Zingale, J. Dursi, A. Siegel, K. Riley (U. of Chicago), and R. Loy (Argonne Lab).

This paper is divided into 2 parts: A short introduction to PARAMESH and a series of brief descriptions of some of the applications it is being used for.

1.1 PARAMESH: What is it ?

PARAMESH is a set of subroutines which are written in Fortran 90. The package is fully parallel and communications between processes are handled using calls to the MPI communications library. PARAMESH has been tested using a number of different Fortran 90/95 compilers and different computer architectures, with an emphasis on using Beowulfs. Some of these include the Portland Group compiler, the NAG compiler, the Intel compiler, and the Lahey/Fujitsu compiler. Architectures which have been used to run PARAMESH include the IBM SP, SGI, HP-Compaq, and Cray. The current released version of PARAMESH is version 3.0. We expect to release version 3.01 before the end of 2003.

The kind of application developers we are targeting with PARAMESH are those who already have a pre-existing, uniform-mesh, serial code. PARAMESH is designed to enable such users to easily incorporate both parallelization and dynamic adaptive mesh refinement (AMR) into their application. We provide templates that

help them to do this. Further, we distribute the source code, allowing users to modify it for their specific application. Alternately, we collaborate with users to make modifications to PARAMESH for their specific application if they request this.

PARAMESH is a subset of the block-adaptive technique described by Burger & Oliger [2] and Burger & Collela [3]. In our case we bi-sect the region of space covered by the computational domain, forming a set of child blocks. Child blocks can themselves be bi-sected, their children bi-sected, and so on. This process is carried out recursively until the desired resolution in a certain region of space is reached. Only jumps in refinement of a factor of 2 in resolution are allowed. This naturally leads to the use of a tree data structure to organize the blocks and their relationships to one another.

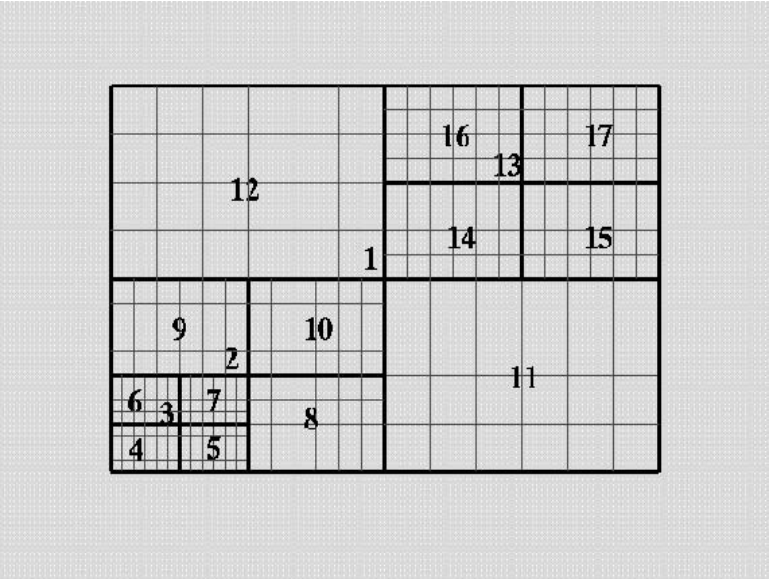


Fig. 1. PARAMESH blocks are created by recursively bisecting space as shown here.

For the purposes of parallelization, the blocks are ordered according to a morton space-filling curve [4]. This curve is 'cut' into a number of pieces equal to the number of processors. The length of each piece is determined by a user selectable work weighting which can be specified at runtime and enables some control over the load balance of the algorithm. This type of curve has the property that blocks which are nearby in physical space also tend to be nearby along the morton curve.

The refinement-derefinement process is very simple from the users' point of view. Blocks are simply marked with logical flags which indicate whether the blocks are to refine or derefine. Calling the PARAMESH routine for refinement-derefinement then results in the appropriate set of new blocks being created (or de-

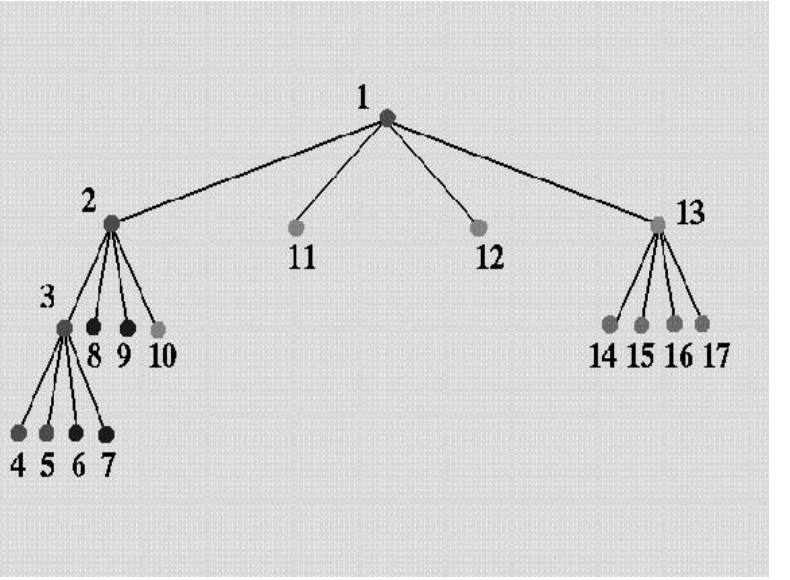


Fig. 2. PARAMESH blocks are arranged according to a tree data structure. We show in the figure above the tree relationships of the blocks originally shown in Figure 1. The numbers shown at the 'nodes' of the tree refer to the numbers of the blocks shown in Figure 1.

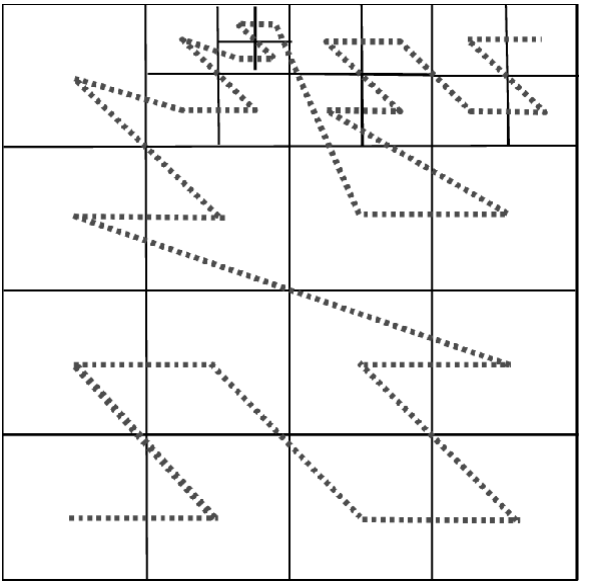


Fig. 3. Morton Space Filling Curve. This figure shows an example of a 'morton space filling' curve threading a set of PARAMESH blocks.

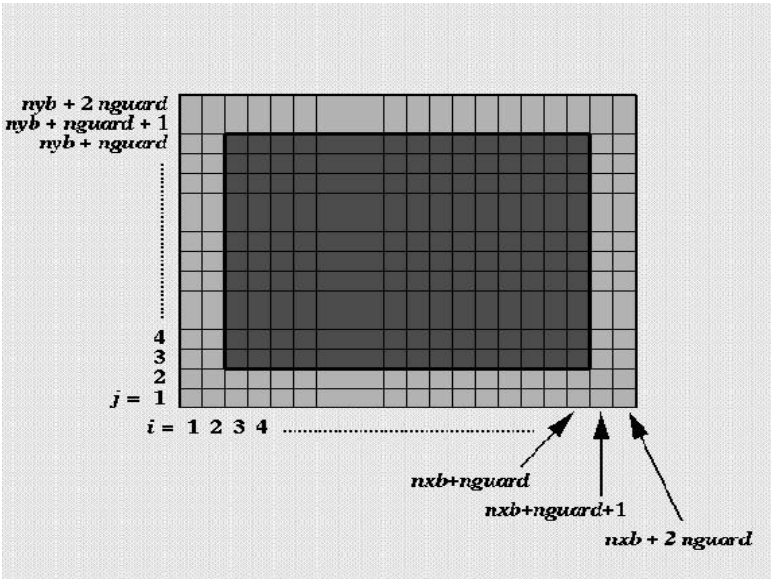


Fig. 4. Each block in PARAMESH is a logically Cartesian mesh of cells. This figure shows an example of a single PARAMESH block. Interior cells of the block are shown in red and the 'guardcells' (cells which overlap with neighboring blocks) are shown in blue. A user of PARAMESH is free to choose the number of guardcells and the number of interior cells in a block.

stroyed), the tree being reconstructed, the blocks being redistributed according to the morton space filling curve, and all internal control information being reconstructed and stored for later use by other PARAMESH functions.

Each PARAMESH block is itself a Cartesian mesh of cells. Each block is surrounded by a 'guard' or 'ghost' cell region. The guardcells are filled by either copying data from neighboring blocks at the same refinement level or by interpolation if there is a refinement jump. The PARAMESH package provides a routine which takes care of this interpolation procedure using Lagrange polynomials to a user chosen, arbitrary order. The user is also free to modify this routine to satisfy any special interpolation requirements they might have for their application.

The user has the option of storing cell, face, edge, or corner data, or any combination of them, on a single computational cell within a PARAMESH block.

We provide support for 'flux' conservation or 'refluxing' at refinement jumps where fluxes (or any quantity) that are computed at cell faces are averaged (or summed) from the fine cells to course cell faces where two blocks of different refinement abut one another.

We also provide a similar procedure which can be applied to values computed at cell edges such that circulation integrals around cell faces are consistent at refinement jumps.

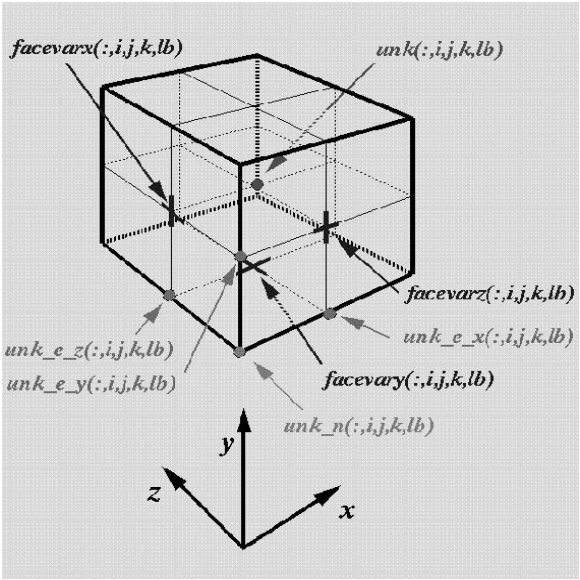


Fig. 5. This figure shows a single cell within a PARAMESH block. Data can be located in any combination at cell centers, cell edges, cell faces, or cell corners.

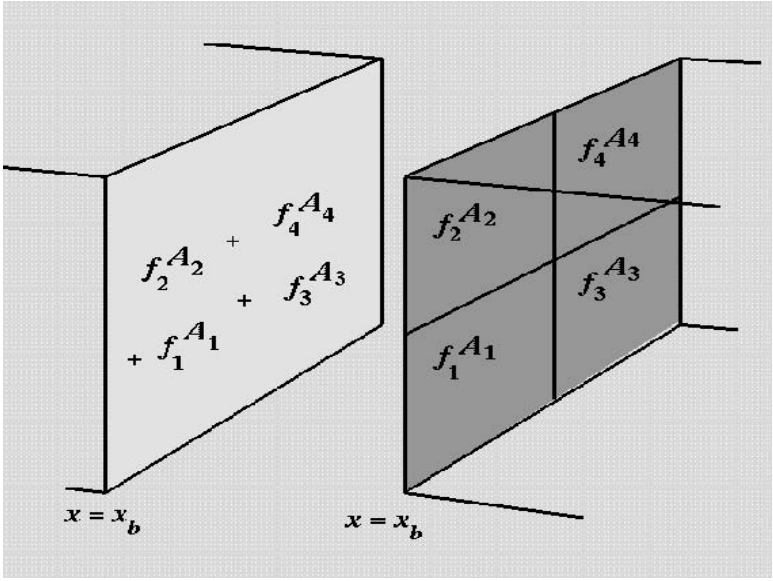


Fig. 6. PARAMESH supports 'refluxing' or flux averaging at jumps in refinement.

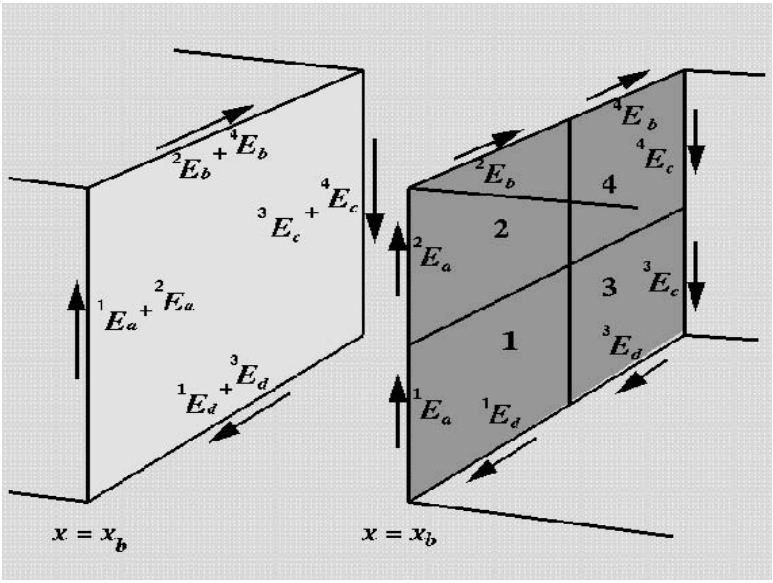


Fig. 7. PARAMESH supports averaging of edge based quantities which can be useful for algorithms which require the circulation integral of that quantity to be continuous at refinement jumps.

2 Applications

2.1 NRL AMR-MHDFCT

The Naval Research Lab's AMR-MHDFCT code was the application for which PARAMESH was originally developed. Many of the choices we made in designing PARAMESH were influenced by the requirements of this code and the requirement that it get a certain level of performance on the Cray T3E.

It is a Flux Corrected Transport MHD code, with magnetic fields stored at cell faces [6]. It uses 3 guardcells around each block. Electric fields are computed at cell edges and the edge averaging described earlier is used to ensure a consist update of the magnetic fields. All other variables are stored at the cell centers. It is mostly being used for Solar physics applications.

2.2 ATHENA

The second code PARAMESH was applied to was another MHD code known as ATHENA. It was written in-house at Goddard by a team lead by D. S. Spicer. It uses a High Order Godunov Technique with a corner-transport-upwind scheme [5], combined with a constrained transport algorithm [9] for advancing the magnetic fields. It can use different cell reconstructions and hence different numbers of guardcells

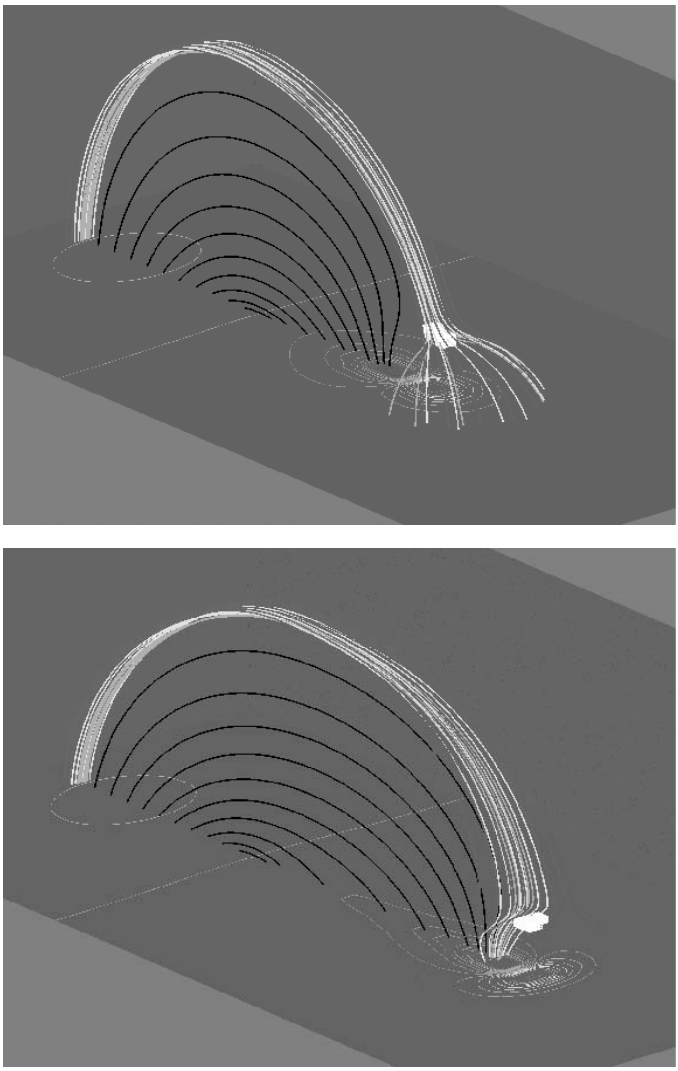


Fig. 8. Shown here are results of a simulation using the NRL AMR-MHDFCT code. The simulation is of an active region of the sun (which starts with closed field lines), testing if that region can move into a coronal hole region with open field lines [7]. In order for this to occur the field lines in the 'front' must reconnect and move to the 'back' of the active region. For this to be simulated successfully, regions with reconnection must be followed with very high resolution. For this calculation refinement is triggered by high values in J^2/B^2. The finest blocks are shown in white. The base mesh was 64x64x128 cells and refinement was allowed to 5 levels below that. This simulation, and many in solar physics, are uniquely suited to AMR because of the need to follow reconnection events, which usually occur on small scales relative to the overall volume of the computation.

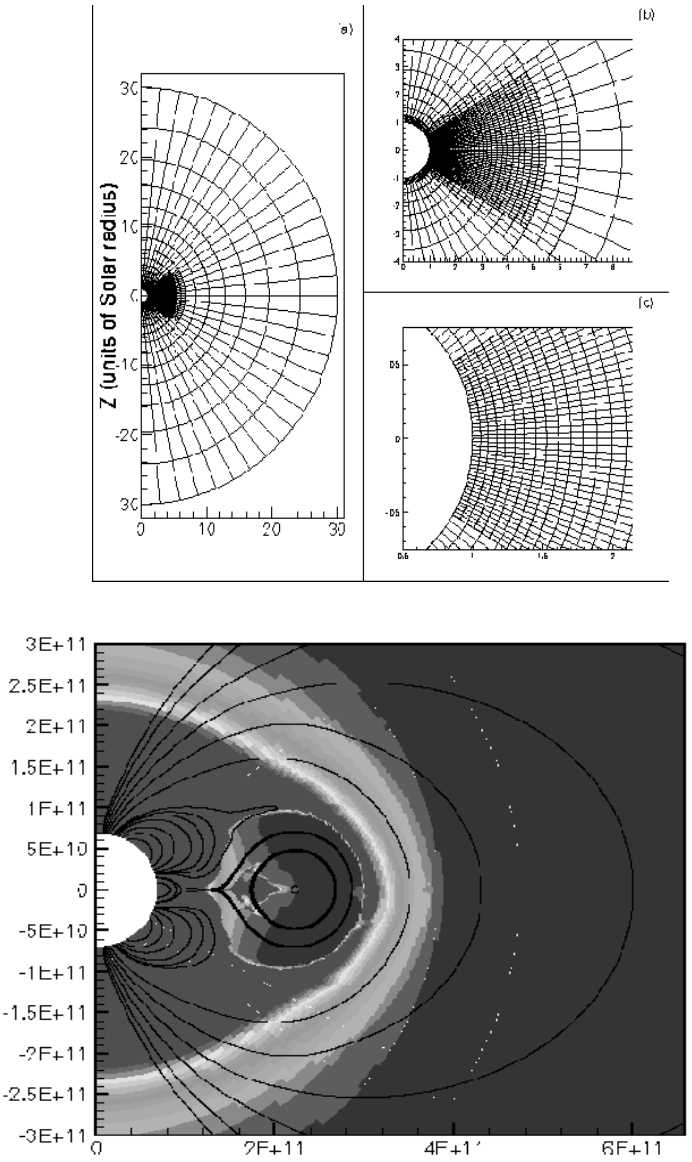


Fig. 9. Results of simulation using a variant of the NRL AMR-MHDFCT code [8]. It uses FCT in the same way as previously described except that it uses spherical coordinates and works in 2.5 dimensions (carries azimuthal B field). The mesh is refined near the inner boundary representing the solar surface as shown in panels a, b and c above. The initial field is a combination of the sun's dipole field superposed with an octopole field near the equator. Shear along the model solar surface is introduced near the equator. As result, a flux rope forms and propagates away from the surface. This is shown in the bottom panel where the magnetic field lines are shown superposed over the density.

around blocks can be used. It is being applied to problems in solar and magnetospheric physics.

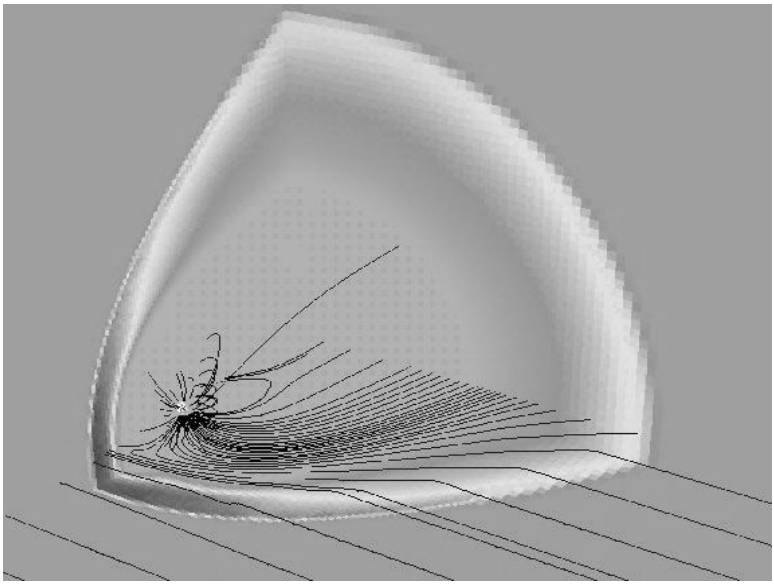


Fig. 10. Results of a simulation of the earth's magnetosphere using ATHENA. Density is plotted, along with the field lines being advected toward the 'earth' as well as the distorted field lines of the earth.

2.3 FLASH

The FLASH code [10, 11] is an astrophysics code which uses PARAMESH as its main AMR package. This code implements a number of CFD schemes, an MHD algorithm, a nuclear reaction network with energy feedback, stellar equations of state, self-gravity using multigrid, and particles (which can be passive or gravitationally interacting). It uses 4 guardcells around each block with all variables stored at cell centers. The main purpose for this code is the simulation of astrophysical thermonuclear flashes which occur in close binary star systems where one of the stars' sizes has exceeded its Roche limit and some of its material is accreting onto the companion star. Depending on conditions, it is believed that such a situation can give rise to observed X-ray bursts, classical novae, and Type I supernovae.

2.4 IBEAM

IBEAM is a project funded by NASA/ESTO-CT to build a modern software framework for Astrophysics. They are using as their starting point the FLASH code and

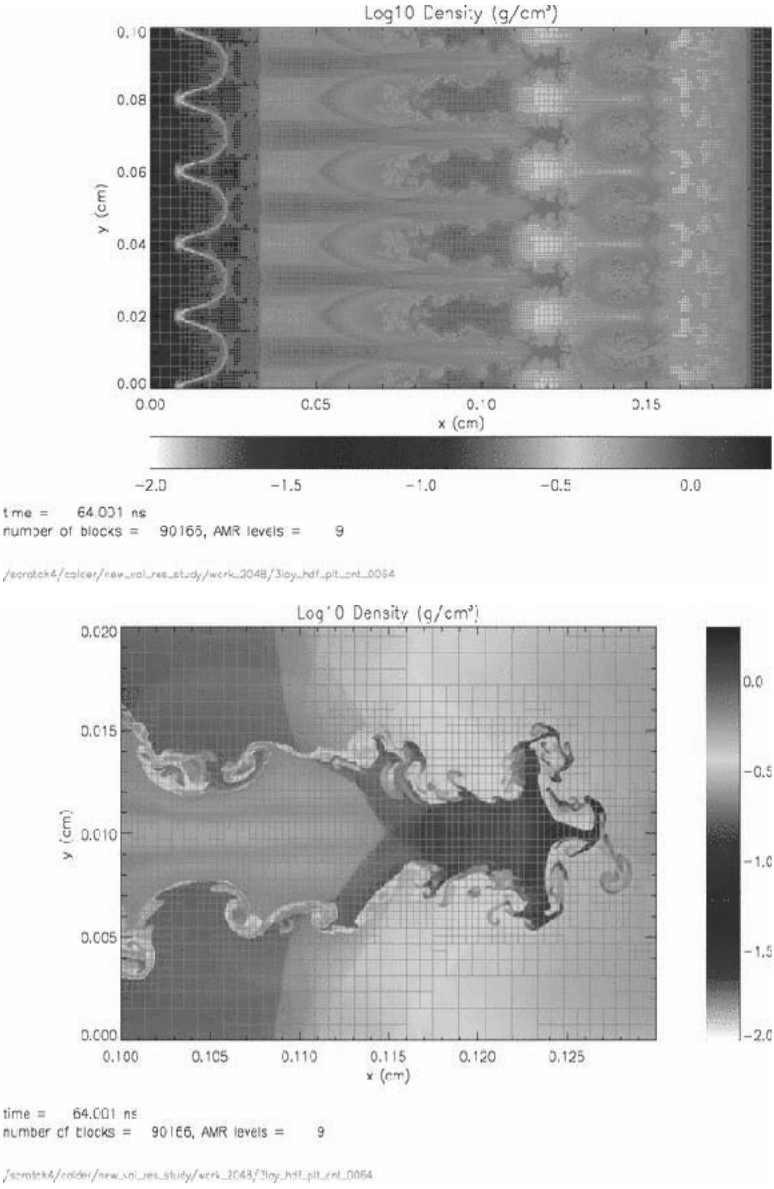


Fig. 11. Example of a calculation using FLASH. The simulation is part of an effort to validate the code by comparing it to laboratory experiments [12]. Shown are the results of a simulation of a layered, 3 material (copper, polymide plastic, and carbonized resorcinol formaldehyde foam) target being hit by a shock. The copper initially has a sinusoidal shape milled into one of its interfaces. As the shock propagates through this interface it excites Richtmeyer-Meshkov instabilities. The above figure shows the numerical experiment at a time well after the shock wave has passed through the interfaces between the three materials. The upper panel shows the entire computational area. The bottom panel shows a blow-up of one of the Richtmeyer-Meshkov fingers.

are collaborating with them and are adding functionality which is compatible with the FLASH code as well as their own framework. The physics they are attempting to model is radiation, hydrodynamics and they have developed relativistic hydro modules, Multigrid for radiation diffusion (single group), BiCGstab algorithm (for radiation). GMRES and Boltzmann transport are currently under development. Their target application is the simulation of Gamma Ray bursts.

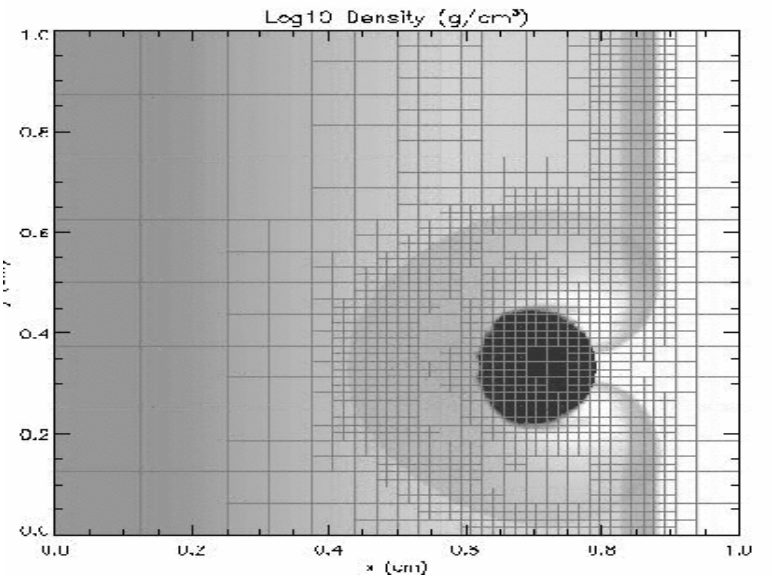


Fig. 12. This figure shows a relativistic shock hitting a dense blob of material is shown. Density is shown with the PARAMESH blocks overlaid.

2.5 Other MHD codes

PARAMESH has been incorporated into several other MHD applications similar to those discussed earlier. D. Odstrcil (NOAA) has used PARAMESH in combination with FCT and TVD schemes for MHD. He is using it to model interacting magnetic flux ropes propagating away from the sun.

Yet another MHD app. using PARAMESH was developed by a group from the University of California at Berkley [13]. They have combined PARAMESH with the Zeus code [14] and are using it for studying the emergence of magnetic flux from the solar surface.

2.6 General Relativity

A group at Goddard lead by Dr. Joan Centrella at Goddard Space Flight Center is developing an application which solves the Einstein equations in order to simulate

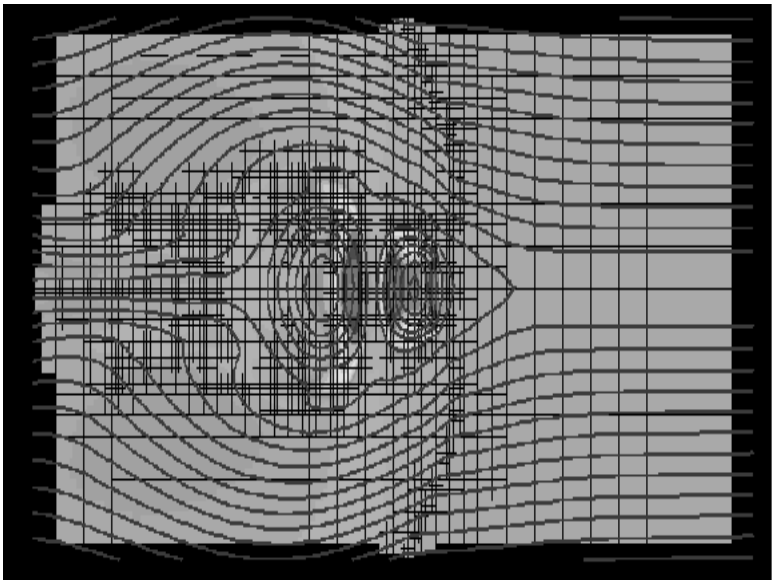


Fig. 13. Simulation of two, interacting Flux Ropes propagating away from the sun (courtesy D. Odstrcil (NOAA)).

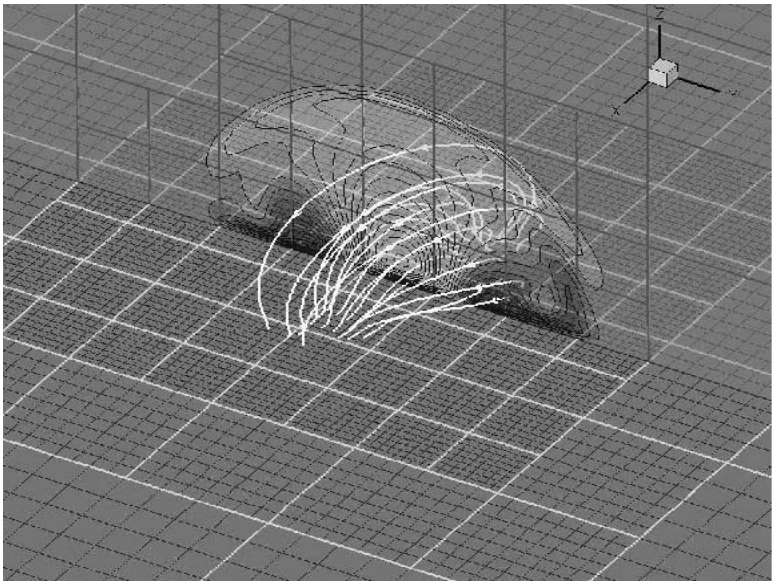


Fig. 14. Simulation of magnetic flux emerging from the solar surface using the ZEUS + PARAMESH code developed at the University of California, Berkley.

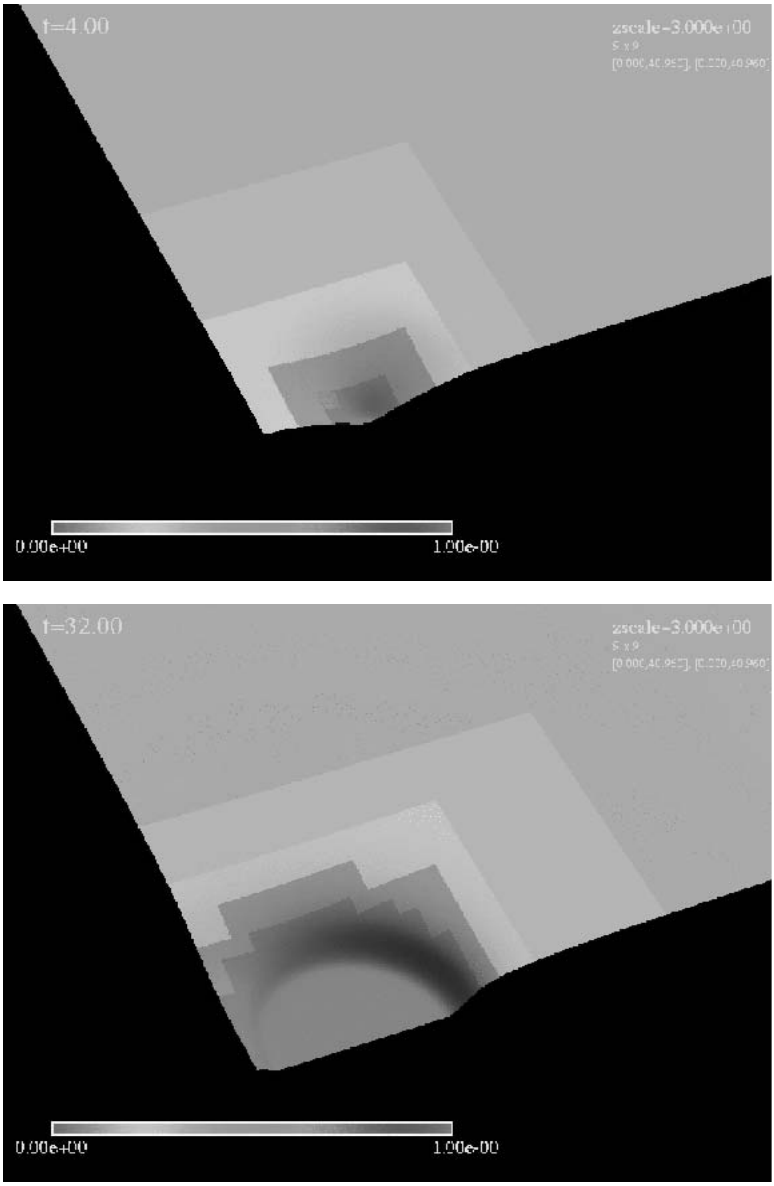


Fig. 15. This slide shows the results of a recent simulation of the head on collision of two black-holes. The meshes of different refinement are shown in different colors. The variable shown is the lapse function (courtesy D. Choi).

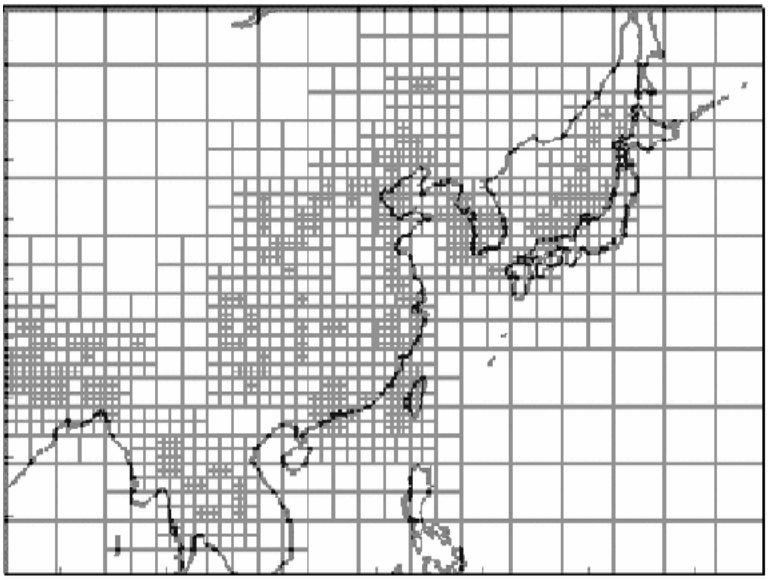


Fig. 16. This slide shows a typical mesh during one of the runs of the above data assimilation/model pollution transport code. The mesh is overlaid on a map of the south Asia.

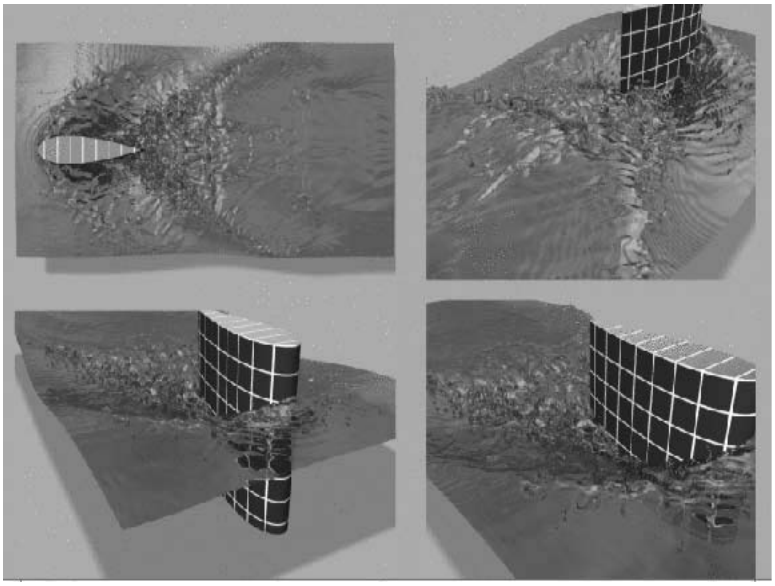


Fig. 17. This figure shows a simulation of vertical 'post' moving through water. The surface of the simulated water surface is shown.

the collision of super-massive Black Holes colliding in the centers of galaxies. They are interested in simulating events of this type since it is fundamental to predicting the amplitude of the propagating gravitational wave-forms which would result. Such simulations are therefore critical to the design and interpretation of measurements from the Laser Interferometer Space Antenna (LISA) to be launched in 2011 [15].

AMR is important for this app. since one needs to model the details of the collision (which occur on small, relative size scales) at the same time a large volume around the collision is modeled to simulate the wave forms which result from and propagate away from the collision site. This group is using finite difference techniques and multigrid. Plans are to also being made to incorporate a compressible fluid algorithm.

2.7 Pollution Transport and Data Assimilation

A group at Michigan Tech. and Virginia Tech. is using PARAMESH for adaptive mesh calculations of Pollution Transport with data assimilation. The model incorporates diffusion and advection of pollution constituents as well as a chemical reaction network which includes 80 chemical species.

A unique feature of the way this group uses PARAMESH is that they refine the mesh only in 2 spatial dimensions while the 3rd (perp. to earth's surface) remains fixed. The model also ingests actual meteorological data and interpolates it to the non-uniform, adapting mesh so that data is assimilated into the running computational model. A specialized routine for doing this was written for them by the PARAMESH developers.

2.8 Ship's Hulls Moving through water

Finally we show an application using PARAMESH for simulating Breaking Waves around ship hulls [16]. The model uses a Finite Volume technique to model water flow. They also use a volume of fluid technique to track the air-water interface and a cut-cell technique to model the interior boundary of the ship hull. They are using PARAMESH as a parallelization technique, but plan to incorporate AMR to follow breaking wave and 'spray' formation in more detail.

References

1. MacNeice, P., Olson, K. M., Mobarry, C., deFainchtein, R., & Packer, C. 2000, Comput. Phys. Comm., 126, 330
2. Burger, M. J. & Oliger, J. 1984, J. Comp. Phys., 53, 484
3. Burger, M. J. & Colella, P. 1989, J. Comp. Phys., 82, 64
4. Warren, M. S. & Salmon, J. K. 1993 in Proc. Supercomputing (Washington DC: IEEE Comput. Soc.), 12
5. Colella, P. 1990, J. Comp. Phys., 87, 171
6. DeVore, R. 1991 ???

7. Antiochos, S. K. 2003, Conference Presentation of the American Astronomical Society, SPD meeting #34, Abstract #01.02
8. MacNeice, P., Antiochos, S. K., Phillips, P., Spicer, D. S., DeVore, C. R., & Olson, K. 2003, in preparation
9. Balsara D. S. & Spicer, D. S. 1999, J. Comput. Phys., 149, 270
10. Fryxell, B., Olson, K., Ricker, P., Timmes, F. X., Zingale, M., Lamb, D. Q., MacNeice, P., Rosner, R., Truran, J. W., & Tufo, H. 2000, ApJS, 131, 273
11. http://flash.uchicago.edu
12. Calder, A.C., Fryxell, B., Plewa, T., Rosner, R., Dursi, L. J., Weirs, V. G., Dupont, T., Robey, H. F., Kane, J. O., Remington, B. A., Drake, R. P., Dimonte, G., Zingale, M., Timmes, F. X., Olson, K., Ricker, P., MacNeice, P., & Tufo, H. M. 2002, ApJS, 143, 201
13. Abbett, W.P., Ledvina, S.A., Fisher, G.H., & MacNeice, P., 2001, American Geophysical Union, Fall Meeting, abstract #SH11C-0727
14. Stone, J. M. & Norman, M. L. 1992, ApJS, 80, 791
15. http://lisa.jpl.nasa.gov
16. Hendricksen, K., Shen, L., Yue, D. K. P., Dommermuth, D. G., Adams, P. 2003, presented at DOD High Performance Computing Program 2003 Users Group Conference, Bellevue, WA

AstroBEAR: AMR for Astrophysical Applications - I: Methods

Poludnenko, A.[1], Varnière, P.[1], Cunningham, A.[1], Frank, A.[1], Mitran. S.[2]

[1] University of Rochester wma,pvarni,ajc4,afrank@pas.rochester.edu
[2] University of North Carolina at Chapel Hill mitran@amath.unc.edu

1 Introduction

Adaptive Mesh Refinement methods combined with modern shock-capturing techniques hold great promise for the simulation of many astrophysical fluid dynamics environments. This is particularly true of problems involving strong shocks followed by efficient cooling via radiative losses. The high degrees of compression achieved in such circumstances require high resolutions that can tax the computational resources of many research groups. The need to accurately compute various micro-physical processes (non-equilibrium ionization, radiation transport, etc.) place further constraints on acceptable resolution for a simulation.

In this contribution we present a new AMR code for astrophysical applications called AstroBEAR. The code is designed on the BEARCLAW framework (Boundary Embedded Adaptive Mesh Refinement for Conservation Laws) [M04] and it offers generalized adaptive facilities including mesh refinement in a grid-based formalism [BL98]. The code is flexible and efficient being designed specifically for multi-physics applications in which processes operating under different time and length scales can be simultaneously simulated.

Our design philosophy in the construction of AstroBEAR was to develop a code that would perform well for a targeted list of astrophysical applications. These include: the interaction of shocks and winds with heterogeneous environments; astrophysical jets; stellar wind blown nebulae; accretion disk physics including the accretion disk wind outflows and the interaction of disks with an embedded proto-planet. These applications are accessed via pre-defined modules. Various modules providing handling of physical processes (ionization dynamics, chemistry, radiative cooling, radiation pressure, central gravity, etc.) can be accessed without code re-compilation and can be switched on and off at runtime by the user.

In this paper we describe the general numerical method used including aspects of the code which are still under development (*i.e.* the formalism for treating flux conservation in MHD).

2 Hydrodynamic and MHD Solver Method

2.1 Integration Scheme

AstroBEAR is designed to solve problems in astrophysical hydrodynamics and magneto-hydrodynamics. The relevant equations we solve are

$$\partial_t \begin{bmatrix} \rho \\ \rho\mathbf{U} \\ E \\ \mathbf{B} \end{bmatrix} + \nabla \cdot \begin{pmatrix} \rho\mathbf{U} \\ \rho\mathbf{U}\otimes\mathbf{U} + \tilde{p}I - \mathbf{B}\otimes\mathbf{B} \\ \mathbf{U}(\varepsilon+\tilde{p}) - \mathbf{B}(\mathbf{U}\cdot\mathbf{B}) \\ \mathbf{U}\otimes\mathbf{B} - \mathbf{B}\otimes\mathbf{U} \end{pmatrix} = \begin{pmatrix} S_\rho \\ S_m \\ S_e \\ S_B \end{pmatrix} \tag{1}$$
$$\nabla \cdot \mathbf{B} = 0$$

The symbols have the standard definition, ρ is the density, $\mathbf{U} = (u_1, u_2, u_3)^T$ is the velocity field, E is the total energy, $\mathbf{B} = (B_1, B_2, B_3)^T$ is the magnetic field, $\tilde{p} = p + |\mathbf{B}|^2/2$ is the total pressure, p is the thermal pressure and $|\mathbf{B}|^2/2$ is the magnetic pressure. Using these definitions the equation of state is:

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho|\mathbf{U}|^2 - \frac{1}{2}|\mathbf{B}|^2\right),$$

where γ is the ideal gas constant.

The mass, momentum, energy and magnetic field source terms (S_ρ, S_m, S_e, S_B) can take a variety of forms depending on the problem including geometrical source terms for non-Cartesian geometries and source terms associated with the solution to the flux conservation constraint as in the 8-wave and GLM methods [P94, D01]. This $\nabla \cdot \mathbf{B} = 0$ condition requires special consideration and we have implemented a number of methods to deal with it.

For both Hydrodynamics and MHD, however, we have chosen to utilize a single integration scheme in the form of the Wave Propagation Algorithm of LeVeque [L97]. We note, though, that the BEARCLAW package includes a variety of integration schemes.

Many of the problems we are interested in studying involve extremely strong shocks, rarefactions and density contrasts. As we shall see in paper II a typical problem may involve flows at Mach numbers up to 200 and density jumps of $\rho_2/\rho_1 > 10^5$. Since all numerical methods have inherent strengths and weaknesses we have chosen to pursue an approach in which a set of Riemann solvers is precompiled in the code and the end user makes a particular choice of the Riemann solver to be used, moreover different Riemann solvers can be used on different subdomains of the computational domain. For example, efficient methods for solving the exact Riemann problem can be formulated [T99] for pure hydrodynamics. For the MHD equations the complexity of the flux Jacobian often leads to the use of linearized Riemann solvers.

In one dimension the Wave Propagation Algorithm takes the form [L97]

$$q_i^* = q_i - \frac{\Delta t}{\Delta x}(\mathcal{A}^+\Delta q_i + \mathcal{A}^-\Delta q_{i+1}) - \frac{\Delta t}{\Delta x}(\tilde{F}_{i+1} - \tilde{F}_{i-1}) \tag{2}$$

where $\tilde{F}$ are the correction fluxes which will be defined in terms of the waves $\mathcal{W}_i^p$ and their speeds λ_i^p arising from the i-th Riemann problem at the interface between cells i and i+1. The $\mathcal{A}^+$ and $\mathcal{A}^-$ terms describe the fluctuation splitting and they are defined as

$$\mathcal{A}^{\pm}\Delta q = \sum_{p=1}^{M} (\lambda^p)^{\pm} \mathcal{W}^p, \tag{3}$$

where $\lambda^- = \min(\lambda, 0)$ and $\lambda^+ = \max(\lambda, 0)$. Without limiters the correction fluxes take the form

$$\tilde{F}_i = \frac{1}{2} \sum_{p=1}^{M} |\lambda_i^p| \left(1 - \frac{\Delta t}{\Delta x} |\lambda_i^p|\right) \mathcal{W}_i^p. \tag{4}$$

Flux, slope or wave limiters ϕ are introduced into the method described above to reduce oscillations resulting from discretization. The functions ϕ adjust the fluxes, carried by the waves into and out of the cell, to reduce high frequency oscillations associated with the Gibbs effect. BEARCLAW provides the user with choices of a variety of standard limiters such as MinMod and Superbee (see [L97] and references therein).

An important component of the wave propagation algorithm is the implementation of transverse waves for multi-dimensional problems. For the example of 2-dimensional problems the governing equations include fluxes in both the x and y directions $F_{i,j}$ and $G_{i,j}$. Since the waves resulting from the solution of a Riemann problem in the normal direction and originating at interfaces should propagate in a multi-dimensional manner one should include their effect on other neighboring cell averages. Transverse fluctuations may be defined by decomposing each fluctuation $\mathcal{A}^*\Delta q$ into the "up-going" $\mathcal{B}^+\mathcal{A}^*\Delta q_{i,j}$ and the "down-going" $\mathcal{B}^-\mathcal{A}^*\Delta q_{i,j}$ components. Here * refers to the left or right propagating fluctuation. In practice the transverse fluctuations are found by determining the Jacobian matrix B of the fluxes G in the transverse dimension. This can be an exact Jacobian matrix as for the case of an exact Riemann solver or an approximate one as for the case of the linearized Riemann solvers. Subsequently eigenvalues μ^s and eigenvectors w^s of B are determined and the latter can be used to decompose $\mathcal{A}^*\Delta q$ into their linear combination. Based on this we finally get for the transverse fluctuations

$$\mathcal{B}^{\pm}\mathcal{A}^*\Delta q_{i,j} = \Sigma_s (\mu^s)^{\pm} \beta^s w^s. \tag{5}$$

Thus when updating the cells (i, j) and $(i+1, j)$ after solving the Riemann problem at the interface between them via an x pass, the transverse fluctuations are used to modify the four neighboring y fluxes below and above these two cells, namely $G_{i,j}$, $G_{i,j-1}$, $G_{i+1,j}$ and $G_{i+1,j-1}$.

2.2 Riemann Solvers

Hydrodynamic Solvers: We have constructed both linearized Roe solvers and exact Riemann solver modules for hydrodynamics. In spite of the additional computational

cost we find that the extreme conditions found in some simulations necessitate an exact Riemann solver. Moreover, the increased stability of the scheme resulting from the use of the exact as opposed to a linearized Riemann solver and, therefore, larger acceptable time steps compensate for the increase in the computational overhead due to the exact Riemann solver and decrease numerical viscosity. In simulations in which a clump is initiated with high velocity we found near vacuum conditions created for which the Roe solver failed to find a solution. In addition, we found strong oscillations in density and pressure behind the extreme shocks and cooling environments. When radiative cooling is included there was also a need to maintain accurate tracking of post-shock conditions to avoid run-away cooling though this issue also relied on the use of a robust ODE solver for stiff systems of equations due to source terms. Finally the formation of so-called carbuncles [Q94] requires particular care. These structures appear as randomly located sharply pointed corrugations in the shock and can completely dominate the behavior of the completed solution.

The use of the exact Riemann solver with the proper consideration of vacuum states and transonic rarefactions resulting from the solution of a Riemann problem resolved many of the issues listed above. However, we found that carbuncles could still form in some cases in particular in the presence of strong radiative cooling. Our solution then relied on two approaches. First, we included the implementation of transverse waves for the exact Riemann solver. This was done by extracting the interface state from the exact solution of the Riemann problem, i.e. the state between the two nonlinear waves, in the normal direction. These hydrodynamic primitives ρ, p, u_1, u_2, u_3 where then used for each transverse dimension to construct the exact Jacobian matrix of the fluxes and then to find its eigenvalues and eigenvectors. After that each fluctuation $\mathcal{A}^*\Delta q$ was decomposed into the "up-going" $\mathcal{B}^+\mathcal{A}^*\Delta q_{i,j}$ and the "down-going" $\mathcal{B}^-\mathcal{A}^*\Delta q_{i,j}$ components as discussed in the previous section.

We also implemented a spectrally differentiated numerical viscosity via splitting the contact discontinuity (CD) into 2 waves $\mathcal{W}^*{}_l$ and $\mathcal{W}^*{}_r$ moving with speeds $u_l^* = u^* - \varepsilon$ and $u_r^* = u^* + \varepsilon$, where u^* is the original speed of the CD. The amount of a conserved quantity Q carried into a cell by the CD is

$$\Delta Q = \Delta t u^* (Q_r^* - Q_l^*), \tag{6}$$

where the * refers to the subregion of the Riemann problem solution between the left- and right-going non-linear waves with r and l denoting the solutions on either side of the CD. Performing a conservative split of ΔQ, assuming here that the Riemann problem is being solved in the x-direction, one gets the state between the waves $\mathcal{W}^*{}_l$ and $\mathcal{W}^*{}_r$

$$\overline{\rho} = \frac{\rho_l^* + \rho_r^*}{2};\ \overline{\rho u_1} = \overline{\rho} u^*;\ \overline{\rho u_{(2,3)}} = \frac{\rho_l^* u_{(2,3),l} + \rho_r^* u_{(2,3),r}}{2};\ \overline{E} = \frac{E_l^* + E_r^*}{2}. \tag{7}$$

The extent of the diffused region then depends on the quantity

$$\varepsilon = \eta_{CD} \mathrm{MIN}(|u_l - u^*|, |u_r - u^*|), \tag{8}$$

where u_l (u_r), depending on the nature of the left (right) non-linear wave, are either the speed of the left (right) shock or the tail of the left (right) rarefaction and η_{CD}

is a user specified quantity between 0 and 1. Using this formalism we were able to eliminate the carbuncles while limiting numerical diffusion.

In Figure 1 we present two 1-D tests of the hydrodynamic method. The first is a Sod-type shock tube designed to assess the entropy satisfaction property of the code. The second tracks the evolution of a strong blast wave and is intended to assess the overall robustness and accuracy of the numerical scheme. Both tests were taken from [T99]. In both cases we find excellent agreement with the exact result with nominal spreading. 2-D results will be presented in Paper II.

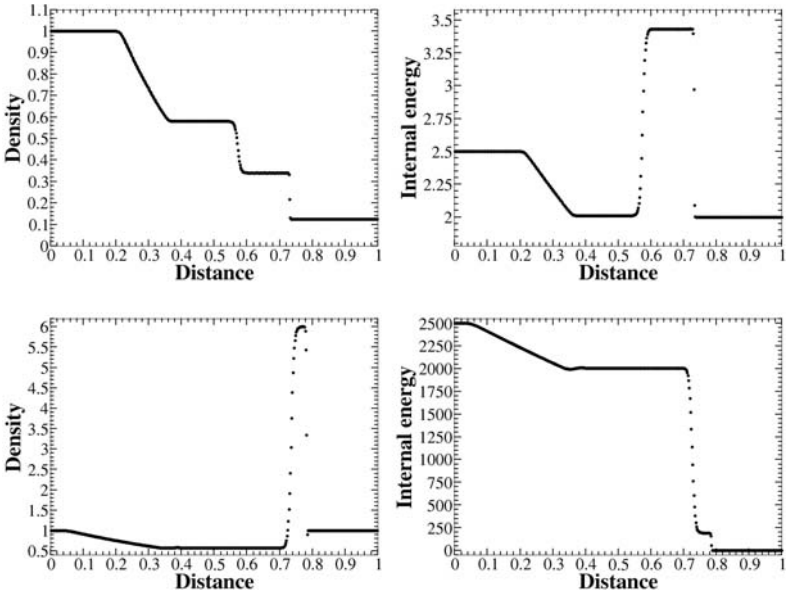


Fig. 1. 1-D tests of the method. Resolution 500 cells, the limiter used was MinMod. *Top:* Sod-type shock tube; time $t = 0.2$; initial discontinuity at $x = 0.3$. *Bottom:* Strong blast wave; time $t = 0.012$; initial discontinuity at $x = 0.5$.

MHD Solvers: To solve the MHD equations we used the Riemann solver developed by J. Rossmanith [R02] in conjunction with the Wave Propagation algorithm. Due to the complexity of the flux Jacobian this approximate Riemann solver uses arithmetic average, instead of Roe average, values to determine the state at cell interfaces. In general the two approaches produce comparable results [RJ95, RJF95].

2.3 Source Terms

In many cases of interest to astrophysical fluid dynamics the source terms will comprise a stiff system of ODE's. Radiative cooling in the limit $t_{cool} << t_{hydro}$ is an often

encountered case where the timescales for changes in the flow variables can have very different values for the different processes (flow vs. radiative cooling). Great care must be taken in integrating such source terms. We have chosen to use the 4th-order accurate implicit Rosenbrock method, in particular its implementation by Kaps and Rentrop [KR79, PTVF97]. This was implemented in the form of a generic module capable of integrating arbitrary systems of source terms that are functions only of temporal and spatial coordinates. For a specific choice of source terms a user simply must provide their description and the Jacobian matrix based on the source term functions. We note that in the presence of extremely strong source terms this method can also fail producing negative densities or pressures. In order to prevent this we included adaptivity in time in the implicit integration.

3 $\nabla \cdot \mathbf{B}$ treatment

The elliptic equation $\nabla \cdot \mathbf{B} = 0$ serves as a constraint for MHD. If the initial field is divergence free, the Maxwell equations retain the condition during subsequent evolution. When solving the MHD equations numerically, however, flux conservation can be violated via truncation errors. Having a non-divergence free magnetic field will create forces along the magnetic field of the order of $F_l \cdot \mathbf{B} = |\mathbf{B}|^2(\nabla \cdot \mathbf{B})$ which can render the simulation useless. Thus at each step the divergence must be corrected, or ''cleaned". Several methods are available and have been discussed extensively in the literature. For example in [T00] these methods are reviewed and compared. As part of an ongoing research effort we have implemented a number of methods in AstroBEAR. We discuss these below.

3.1 The 8 Waves Method

Powell [P94] derived the MHD equations without the constraint $\nabla \cdot \mathbf{B} = 0$. This creates an additional source term S in the system (1)

$$S = -(\nabla \cdot \mathbf{B}) \begin{pmatrix} 0 \\ \mathbf{B} \\ \mathbf{U} \\ \mathbf{U} \cdot \mathbf{B} \end{pmatrix}. \tag{9}$$

The equation for the divergence of $\mathbf{B}$ now becomes an advection equation. The non-vanishing divergence term will be advected with the flow instead of remaining at the location of its creation. This procedure can work well only with outflow boundary conditions. It does have the great advantage of being inexpensive compared to other methods. However, it may also create incorrect jump condition across strong shocks because we are not solving the "right" MHD equations.

3.2 The GLM Method

This method, introduced by Dedner *et al.* [D01] is a generalization of the 8 waves method of Powell. It introduces a generic set of equations for dealing with the divergence of **B**. These equations can be hyperbolic, elliptic or parabolic. We have coded a form of the advection-diffusion equation which advects the divergence of **B** error at the fastest speed in the system while also diffusing it away. Once again the system of equations that is solved is not the original MHD set and thus has the same problem as the 8 waves method around strong shocks.

3.3 The Projection Method

A number of tests were carried out with a projection method suitable for AMR computations. The full details of the method may be found in [MVF04]; a brief description is given here. It was shown by Tóth [T00] that the Poisson solver step, required in the projection method and proposed by Brackbill and Barnes [BB80], does not entail excessive computational penalties, especially if fast techniques are used. The problem that arises in applications to AMR is that the grids on different levels are coupled leading to the necessity of solving the Poisson equation on the entire grid hierarchy for each of the time subcycling steps. While this is indeed the case when applying the projection method to equations exhibiting a true elliptic subsystem (e.g., the incompressible Navier-Stokes equations), in the case of MHD the equation set is hyperbolic so a grid-by-grid correction procedure is feasible. The procedure is presented in the context of a 2D cell-centered finite volume method with the cell average magnetic field on a coarse grid cell $C^{2h}_{i,j}$ with edges of length $2h$ and cell area $A^{2h}_{i,j}$ defined by[3]

$$\mathbf{B}_{i,j} = \frac{1}{A^{2h}_{i,j}} \int_{C^{2h}_{i,j}} \mathbf{B}(x,y)\, dxdy\,. \tag{10}$$

The embedded fine grid field is respectively

$$\mathbf{b}_{i,j} = \frac{1}{A^{h}_{i,j}} \int_{C^{h}_{i,j}} \mathbf{B}(x,y)\, dxdy. \tag{11}$$

Three issues have to be addressed in constructing a projection correction for an AMR computation of MHD flows:

1. the interpolation operator used to initialize fine grid values from coarse grids should not introduce divergence errors;
2. the restriction operator used to update coarse grid values from more accurate fine grid values should maintain zero-divergence of the finer grid;
3. the fixup procedure employed to ensure conservation at coarse-fine interfaces should maintain zero divergence on the coarse grid.

[3]Here as an example we assume the refinement ratio of 2 though it can be any positive even number.

The main interest in this work is in 2nd-order accurate schemes so the divergence of the magnetic field is sought to be maintained zero to $O(h^2)$ accuracy. It is straightforward [MVF04] to show that directly injecting coarse grid values to embedded fine grids (constant function interpolation) leads to $O(h)$ errors in the divergence on the fine grid. Linear interpolation, either continuous or discontinuous, at adjoining cell faces does lead to $O(h^2)$ accuracy in the fine grid divergence and is a suitable interpolation procedure. For instance, in the case of piecewise continuous linear interpolation the child grid values are given by

$$\mathbf{b}_{2i-1,2j-1} = \mathbf{B}^1_{i,j}(x_i - \frac{h}{2}, y_j - \frac{h}{2}), \qquad \mathbf{b}_{2i,2j-1} = \mathbf{B}^2_{i,j}(x_i + \frac{h}{2}, y_j - \frac{h}{2}), \tag{12}$$

$$\mathbf{b}_{2i-1,2j} = \mathbf{B}^3_{i,j}(x_i - \frac{h}{2}, y_j + \frac{h}{2}), \qquad \mathbf{b}_{2i,2j} = \mathbf{B}^4_{i,j}(x_i + \frac{h}{2}, y_j + \frac{h}{2}), \tag{13}$$

with the interpolation functions defined by

$$\mathbf{B}^1_{i,j}(x,y) = (\mathbf{B}_{i,j} - \mathbf{B}_{i-1,j})\frac{x-x_i}{2h} + (\mathbf{B}_{i,j} - \mathbf{B}_{i,j-1})\frac{y-y_j}{2h} + \mathbf{B}_{i,j}\ , \tag{14}$$

$$\mathbf{B}^2_{i,j}(x,y) = (\mathbf{B}_{i+1,j} - \mathbf{B}_{i,j})\frac{x-x_i}{2h} + (\mathbf{B}_{i,j} - \mathbf{B}_{i,j-1})\frac{y-y_j}{2h} + \mathbf{B}_{i,j}\ , \tag{15}$$

$$\mathbf{B}^3_{i,j}(x,y) = (\mathbf{B}_{i,j} - \mathbf{B}_{i-1,j})\frac{x-x_i}{2h} + (\mathbf{B}_{i,j+1} - \mathbf{B}_{i,j})\frac{y-y_j}{2h} + \mathbf{B}_{i,j}\ , \tag{16}$$

$$\mathbf{B}^4_{i,j}(x,y) = (\mathbf{B}_{i+1,j} - \mathbf{B}_{i,j})\frac{x-x_i}{2h} + (\mathbf{B}_{i,j+1} - \mathbf{B}_{i,j})\frac{y-y_j}{2h} + \mathbf{B}_{i,j}\ . \tag{17}$$

Taylor series expansion of the discrete fine grid divergence shows that second order accuracy is maintained

$$\nabla_h \cdot \mathbf{b}_{2i-1,2j-1} = \frac{\partial B^x}{\partial x} + \frac{\partial B^y}{\partial y} - \frac{h}{2}\left[\left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y}\right)\left(\frac{\partial B^x}{\partial x} + \frac{\partial B^y}{\partial y}\right)\right] + O(h^2)\ , \tag{18}$$

as long as the coarse grid field itself has zero divergence to second-order accuracy

$$\frac{\partial B^x}{\partial x} + \frac{\partial B^y}{\partial y} = O(h^2)\ . \tag{19}$$

The natural restriction operator used in AMR computations of conservation laws is simple averaging of embedded fine grid quantities

$$\tilde{\mathbf{B}}_{i,j} = \frac{1}{4}\left(\mathbf{b}_{2i-1,2j-1} + \mathbf{b}_{2i,2j-1} + \mathbf{b}_{2i-1,2j} + \mathbf{b}_{2i,2j}\right) \tag{20}$$

This can be shown to maintain $O(h^2)$ accuracy for the updated coarse grid values at cells away from coarse-fine interfaces since a series expansion at such points gives

$$\nabla_{2h} \cdot \tilde{\mathbf{B}}_{i,j} = \frac{\partial B^x}{\partial x} + \frac{\partial B^y}{\partial y} + \frac{h^2}{8}\left[\frac{\partial^3 B^x}{\partial x \partial y^2} + \frac{\partial^3 B^y}{\partial x^2 \partial y} + \frac{19}{3}\left(\frac{\partial^3 B^x}{\partial x^3} + \frac{\partial^3 B^y}{\partial y^3}\right)\right] + O(h^3)\ . \tag{21}$$

However at coarse-fine interfaces (denoted, for instance, by an index pair (I,J)) the same procedure leads to a divergence

$$\nabla_{2h}\cdot\tilde{\mathbf{B}}_{I,J}=\frac{\partial B^x}{\partial x}+\frac{\partial B^y}{\partial y}-\frac{h}{32}\left[\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)(B^x+B^y)\right]+O(h^2)\,, \tag{22}$$

introducing a 1st-order error in divergence. Further errors are introduced by the conservative fixup procedures typically used in AMR computations. This leads to the necessity of re-establishing a divergence-free field on a coarse grid after time subcycling has been completed and fine grid values along with conservative fixups have been introduced on the coarse grid. It is relatively easy to include the appropriate correction in the sequence of AMR computations:

1. trial time steps taken on a coarse grid determine placement of fine grids; if the initial coarse grid magnetic field is divergence-free to $O(h^2)$ accuracy so is the field on the newly constructed fine grids;
2. the coarse grid is advanced to $t+\Delta t$ through some procedure of solving the MHD equations; typically this would introduce divergence errors which remain not corrected for now;
3. time steps are taken on the fine grid; assume that the fine grid is at the finest level of resolution allowed during the computation, then after each time step required for time subcycling a projection correction is applied individually on each fine grid;
4. at the end of fine grid time subcycling, fine grid values are injected onto the coarse grid and the conservative fixup is applied; the projection procedure is now applied on the coarse grid by solving

$$\triangle\phi=\nabla\cdot\overline{\mathbf{B}}, \tag{23}$$

where $\overline{\mathbf{B}}$ is the initial estimate of the magnetic field at $t+\Delta t$, and setting

$$\mathbf{B}^{n+1}=\overline{\mathbf{B}}-\nabla\phi\,. \tag{24}$$

The Poisson equation is discretized using a standard 5-point stencil and solved through an FFT-based method.

The above procedure is repeated for all embedded grid levels. One question that arises is the posing of boundary conditions for the fine grids. In most of AMR computations boundary conditions for fine grids are imposed by space-time interpolation of coarse grid quantities. This would introduce additional errors in the context of the above procedure since a divergence correction for the coarse grid at the new time $t+\Delta t$ has not been carried out when boundary values for fine grids are required. Though the errors would be eliminated during the course of the fine grid projection step, it is preferable not to introduce such errors at all. This is accomplished through an extended boundary ghost cell region, initialized from coarse grid values at t and which is divergence free to $O(h^2)$ by virtue of the linear interpolation procedure.

Additional problems arise when shocks form in the flow domain. In this case the stencil of the discrete approximation of the Laplacian operator is not valid when

crossing the shock. An additional source term must be included in the Poisson equation to account for this effect. For details see [MVF04].

Acknowledgments This research was sponsored by DOE/NNSA Cooperative Agreement DE-F03-02NA00057 with Cornell University by subcontract NO 41843-7012. as well as by NSF grant AST-9702484, NASA grant NAG5-8428 and the Laboratory for Laser Energetics under DOE sponsorship.

References

BB80. J.U. Brackbill & D.C. Barnes, *J. Comp. Phys.*, 35:426, 1980
BL98. M.J. Berger, R.J. LeVeque, *SIAM J. Numer. Anal.*, 35(6):2298, 1998
D01. A. Dedner, F. Kemm, D. Kroner, C.D. Munz, T. Schnitzer, M. Wesenberg, Albert-Ludwigs-Universitat Freiburg, preprint Nr. 13/2001, 2001
KR79. P. Kaps, P. Rentrop, *Numerische Mathematik*, 33:55, 1979
L97. R.J. LeVeque, J. Comp. Phys., 131:327, 1997
M04. S. Mitran, *these proceedings*
MVF04. S. Mitran, P. Varnière, A. Frank , *J. Comp. Phys.*, 2003 (submitted).
P94. K.G. Powell, Technical Report 94-24, ICASE, Langley, VA, 1994
PTVF97. W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in Fortran 77: The Art of Scientific Computing, Vol. 2, 1997, (2nd ed.; Cambridge University Press)
Q94. J.J. Quirk, *Intl. J. Numer. Meth. Fluids*, 18:555, 1994
R02. J.A. Rossmanith, *Ph.D. Thesis*, University of Washington, 2002
RJ95. D.S. Ryu & T.W. Jones, *Astrophys. J.*, 442:228, 1995
RJF95. D.S. Ryu, T.W. Jones & A. Frank, *Astrophys. J.*, 452:785, 1995
T99. E.F. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics: a Practical Introduction, 1999, (2nd ed.; Springer-Verlag)
TP99. Tagger & Pellat, *Astron. & Astrophys.*, 349:1003, 1999.
T00. G.Tóth, *J. Comp. Phys.*, 161:605, 2000.

Introducing Enzo, an AMR Cosmology Application

Brian W. O'Shea[1], Greg Bryan[2], James Bordner[1], Michael L. Norman[1], Tom Abel[3], Robert Harkness[4] and Alexei Kritsuk[1]

[1] Laboratory for Computational Astrophysics at the Center for Astrophysics and Space Sciences, University of California at San Diego, La Jolla, CA 92093, USA
bwoshea@cosmos.ucsd.edu
[2] Department of Physics, University of Oxford, Oxford Ox1 3RH
[3] Department of Astronomy and Astrophysics, Penn State University, University Park, PA 16802, USA
[4] San Diego Supercomputing Center, UCSD, La Jolla, CA 92093, USA

Abstract. In this paper we introduce Enzo, a 3D MPI-parallel Eulerian block-structured adaptive mesh refinement cosmology code. Enzo is designed to simulate cosmological structure formation, but can also be used to simulate a wide range of astrophysical situations. Enzo solves dark matter N-body dynamics using the particle-mesh technique. The Poisson equation is solved using a combination of fast fourier transform (on a periodic root grid) and multigrid techniques (on non-periodic subgrids). Euler's equations of hydrodynamics are solved using a modified version of the piecewise parabolic method. Several additional physics packages are implemented in the code, including several varieties of radiative cooling, a metagalactic ultraviolet background, and prescriptions for star formation and feedback. We also show results illustrating properties of the adaptive mesh portion of the code. Information on profiling and optimizing the performance of the code can be found in the contribution by James Bordner in this volume.

1 Introduction

In astrophysics in general, and cosmology in particular, any given object of interest can have many important length and time scales. An excellent example of this is the process of galaxy formation. When studying the assembly of galaxies in a cosmological context, one wants to resolve a large enough volume of the universe to capture enough large-scale structure (a box with length on the order of several megaparsecs[5]). However, in order to adequately resolve structure in an individual galaxy one wants to have resolution two orders of magnitude smaller than the ultimate size of the objects of interest (a dwarf galaxy is on the order of ~ 1 kiloparsec). This typical cosmological problem requires roughly five orders of magnitude of dynamical

[5] 1 parsec = 3.26 light years = $3.0857 \times 10^{18} cm$

range, which is prohibitively expensive when done using a single grid. Many other astrophysical phenomena, such as the study of molecular clouds and the formation of stars and galaxy clusters, require similarly large dynamical range. Many scientists have adopted Lagrangean techniques such as smoothed particle hydrodynamics (SPH)[1] to address these issues. However, this type of method suffers from several drawbacks, including poor shock resolution and fixed mass resolution in regions of interest. The use of grid-based techniques with structured adaptive mesh refinement avoids many of these problems, and additionally allows the use of higher-order hydrodynamics schemes.

In this paper we present Enzo, an MPI-parallel 3D Eulerian adaptive mesh refinement code. Though it was originally designed to study cosmological structure formation, the code is extensible and can be used for a wide range of astrophysical phenomena. For more information on the performance and optimization of the Enzo code, see the contribution by James Bordner in this volume. The Enzo web page, which contains documentation and the source code, can be found at `http://cosmos.ucsd.edu/enzo/`.

2 Methodology

2.1 Application and AMR Implementation

Enzo is developed and maintained by the Laboratory for Computational Astrophysics at the University of California in San Diego. The code is written in a mixture of C++ and Fortran 77. High-level functions and data structures are implemented in C++ and computationally intensive lower-level functions are implemented in Fortran. Enzo is parallelized using the MPI message-passing library[6] and uses the HDF5[7] data format to write out data and restart files in a platform-independent format. The code is quite portable and has been ported to numerous parallel shared and distributed memory systems, including the IBM SPs and p690 systems, SGI Origin 2000s and numerous Linux Beowulf-style clusters.

The code allows hydrodynamic and N-body simulations in 1, 2 and 3 dimensions using the structured adaptive mesh refinement of Berger & Colella[2], and allows arbitrary integer ratios of parent and child grid resolution and mesh refinement based on a variety of criteria, including baryon and dark matter overdensity or slope, the existence of shocks, Jeans length, and cell cooling time. The code can also have fixed static nested subgrids, allowing higher initial resolution in a subvolume of the simulation. Refinement can occur anywhere within the simulation volume or in a user-specified subvolume.

The AMR grid patches are the primary data structure in Enzo. Each individual patch is treated as an individual object, and can contain both field variables and particle data. Individual patches are organized into a dynamic distributed AMR mesh

[6] `http://www-unix.mcs.anl.gov/mpi/`
[7] `http://hdf.ncsa.uiuc.edu/HDF5/`

hierarchy using arrays of linked lists to pointers to grid objects. The code uses a simple dynamic load-balancing scheme to distribute the workload within each level of the AMR hierarchy evenly across all processors.

Although each processor stores the entire distributed AMR hierarchy, not all processors contain all grid data. A grid is a *real grid* on a particular processor if its data is allocated to that processor, and a *ghost grid* if its data is allocated on a different processor. Each grid is a real grid on exactly one processor, and a ghost grid on all others. When communication is necessary, MPI is used to transfer the mesh or particle data between processors. The tree structure of a small illustrative 2D AMR hierachy – six total grids in a three level hierarchy distributed across two processors – is shown on the left in Figure 1.

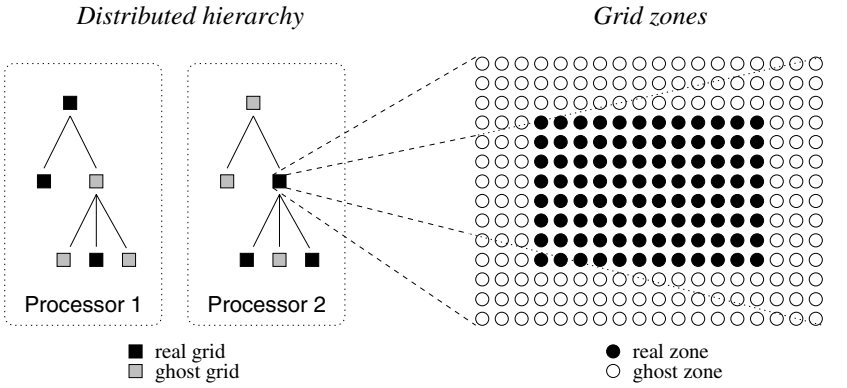


Fig. 1. Real and ghost grids in a hierarchy; real and ghost zones in a grid.

Each data field on a real grid is an array of zones with dimensionality equal to that of the simulation (typically 3D in cosmological structure formation). Zones are partitioned into a core block of *real zones* and a surrounding layer of *ghost zones*. Real zones are used to store the data field values, and ghost zones are used to temporarily store neighboring grid values when required for updating real zones. The ghost zone layer is three zones deep in order to accomodate the computational stencil in the hydrodynamics solver (Section 2.3), as indicated in the right panel in Figure 1. These ghost zones can lead to significant computational and storage overhead, especially for the smaller grid patches that are typically found in the deeper levels of an AMR grid hierarchy.

For more information on Enzo implementation and data structures, see references [3], [4], [5] and [6].

2.2 N-body Dynamics

The dynamics of large-scale structures are dominated by "dark matter," which accounts for $\sim 85\%$ of the matter in the universe but can only influence baryons via

gravitational interaction. There are many other astrophysical situations where gravitational physics is important as well, such as galaxy collisions, where the stars in the two galaxies tend to interact in a collisionless way.

There are multiple ways that one can go about calculating the gravitational potential (which is an elliptical equation in the Newtonian limit) in a structured AMR framework. One way would be to model the dark matter (or other collisionless particle-like objects, such as stars) as a second fluid in addition to the baryon fluid and solve the collisionless Boltzmann equation, which follows the evolution of the fluid density in both physical space and velocity space (referred to collectively as "phase space". This is computationally prohibitive due to the large dimensionality of the problem and because the interesting portion of the solution to the equation does not tend to occupy a small volume of the computational domain, which makes this approach unappealing in the context of an AMR code.

Enzo uses a totally different approach to collisionless systems, namely, the N-body method. This method follows trajectories of a representative sample of individual particles and is much more efficient than a direct solution of the Boltzmann equation in most astrophysical situations. The particle trajectories are controlled by a simple set of coupled equations (for simplicity, we omit cosmological terms):

$$\frac{d\vec{x_p}}{dt} = \vec{v_p} \tag{1}$$

$$\frac{d\vec{v}_p}{dt} = -\nabla\phi \tag{2}$$

Where $\vec{x_p}$ and $\vec{v_p}$ are the particle position and velocity vectors, respectively, and the term on the right-hand side of the second equation is the gravitational force term. The solution to this can be found by solving the elliptic Poisson's equation:

$$\nabla^2\phi = 4\pi G\rho \tag{3}$$

where ρ is the density of both the collisional fluid (baryon gas) and the collisionless fluid (particles).

These equations are finite-differenced and for simplicity are solved with the same timestep as the equations of hydrodynamics. The dark matter particles are sampled onto the grids using the triangular-shaped cloud (TSC) interpolation technique to form a spatially discretized density field (analogous to the baryon densities used to calculate the equations of hydrodynamics) and the elliptical equation is solved using FFTs on the triply periodic root grid and multigrid relaxation on the subgrids. Once the forces have been computed on the mesh, they are interpolated to the particle positions where they are used to update their velocities.

2.3 Hydrodynamics

The primary hydrodynamic method used in Enzo is based on the piecewise parabolic method (PPM) of Woodward & Colella [7] which has been significantly modified for

the study of cosmology. The modifications and several tests are described in Bryan et al. [8], but we provide a short description here.

PPM is a higher-order-accurate version of Godunov's method with third-order-accurate piecewise parabolic monotolic interpolation and a nonlinear Riemann solver for shock capturing. It does an excellent job capturing strong shocks and outflows. Multidimensional schemes are built up by directional splitting, and produce a method that is formally second-order-accurate in space and time and explicitly conserves energy, momentum and mass flux. The conservation laws for fluid mass, momentum and energy density are written in comoving coordinates for a Friedman-Robertson-Walker spacetime. Both the conservation laws and Riemann solver are modified to include gravity , as calculated in Section 2.2.

There are many situations in astrophysics, such as the bulk hypersonic motion of gas, where the kinetic energy of a fluid can dominate its internal energy by many orders of magnitude. In these situations, limitations on machine precision can cause significant inaccuracy in the calculation of pressures and temperatures in the baryon gas. In order to address this issues, Enzo solves both the internal gas energy equation and the total energy equation everywhere on each grid, at all times. This *dual energy formalism* ensures that the method yields the correct entropy jump at strong shocks and also yields accurate pressures and temperatures in cosmological hypersonic flows.

As a check on our primary hydrodynamic method, we also include an implementation of the hydro algorithm used in the Zeus astrophysical code. [9, 10] This staggered grid, finite difference method uses artificial viscosity as a shock-capturing technique and is formally first-order-accurate when using variable timesteps (as is common in structure formation simulations), and is not the preferred method in the Enzo code.

2.4 Additional Physics Packages

Several physics packages are implemented in addition to dark matter and adiabatic gas dynamics. The cooling and heating of gas is extremely important in astrophysical situations. To this extent, two radiative cooling models and several uniform ultraviolet background models have been implemented in an easily extensible framework.

The simpler of the two radiative cooling models assumes that all species in the baryonic gas are in equilibrium and calculates cooling rates directly from a cooling curve assuming $Z = 0.3\ Z_{\odot}$. The second routine, developed by Abel, Zhang, Anninos & Norman [11, 12], assumes that the gas has primordial abundances (ie, a gas which is composed of hydrogen and helium, and unpolluted by metals), and solves a reaction network of 28 equations which includes collisional and radiative processes for 9 seperate species (H,H^+,He, He^+, He^{++},H^-,H_2^+,H_2, and e^-. In order to increase the speed of the calculation, this method takes the reactions with the shortest time scales (those involving H^- and H_2^+) and decouples them from the rest of the reaction network and imposes equilibrium concentrations, which is highly accurate for cosmological processes. See Anninos et al. [12] and Abel et al. [11] for more information.

The vast majority of the volume of the present-day universe is occupied by low-density gas which has been ionized by ultraviolet radiation from quasars, stars and other sources. This low density gas, collectively referred to as the "Lyman-α Forest" because it is primarily observed as a dense collection of absorption lines in spectra from distant quasars (highly luminous extragalactic objects), is useful because it can be used to determine several cosmological parameters and also as a tool for studying the formation and evolution of structure in the universe (see [13] for more information). The spectrum of the ultraviolet radiation background plays an important part in determining the ionization properties of the Lyman-α forest, so it is very important to model this correctly. To this end, we have implemented several models for uniform ultraviolet background radiation based upon the models of Haardt & Madau [14].

One of the most important processes when studying the formation and evolution of galaxies (and to a lesser extent, groups and clusters of galaxies and the gas surrounding them) is the formation and feedback of stars. We use a heuristic prescription similar to that of Cen & Ostriker [15] to convert gas which is rapidly cooling and increasing in density into star "particles" which represent an ensemble of stars. These particles then evolve collisionlessly while returning metals and thermal energy back into the gas in which they formed via hot, metal-enriched winds.

As mentioned in Section 1, Enzo can be downloaded from the web at `http://cosmos.ucsd.edu/enzo/`. Vigorous code development is taking place, and we are in the process of adding ideal magnetohydrodynamics and a flux-limited radiation diffusion scheme to our AMR code, which will significantly enhance the capabilities of the code as a general-purpose astrophysical tool.

3 Adaptive Mesh Characteristics

The adaptive nature of grid cells in the AMR simulations results in a wide range of baryon mass scales being resolved. Figure 2 shows the distribution of cells as a function of overdensity for a range of Enzo simulations in a simulation volume which is $3\ h^{-1}$ megaparsecs on a side. These simulations use either a 64^3 or 128^3 root grid and either 5 or 6 levels of refinement (such that $L_{box}/e = 4096$, where L_{box} is the box size and e is the smallest spatial scale that can be resolved). All grids are refined by a factor of 2.0, and grids are refined when dark matter density (baryon density) exceeds a factor of 4.0 (2.0) times the mean density of cells at that level. In addition, a simulation is performed where the overdensity threshold is doubled. Initial conditions are generated using power spectra and methods common to cosmological simulations. Examination of Figure 2 shows that the entire density range in the simulations is covered by large numbers of cells. In particular, cells at low densities are well-resolved in these simulations, which is in stark contrast to simulations performed using Lagrangian methods, which are typically undersampled at low density. Raising the overdensity threshold for refinement decreases the total number of cells but their relative distribution as a function of overdensity is unchanged. In all simulations the total number of cells at the end of the run has increase by a factor of $\sim 8 - 10$ from the number of cells in the root grid.

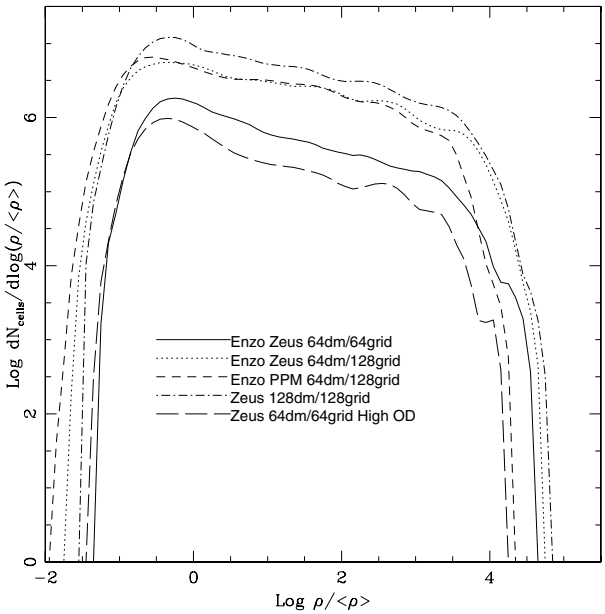


Fig. 2. Number of cells as a function of baryon overdensity (normalized by bin size) for a representative suite of Enzo cosmology simulations. The simulations are labelled such that the first number corresponds to the number of dark matter particles and the second number shows the root grid size, ie, 64dm/128grid means that the simulation has 64^3 dark matter particles and a 128^3 root grid. Simulations with two different hydrodynamic methods are used. PPM: Piecewise Parabolic Method. [8] Zeus: The hydro method used in the Zeus astrophysical code. [9]

Figure 3 shows the distribution of number of cells at the end of a simulation as a function of the mass of baryons in that cell. Arrows indicate the mean cell mass contained on the root grid at the onset of the simulation for simulations covering the same spatial volume as the simulations described above with a $64^3, 128^3$ or 256^3 root grid (labelled N64, N128 and N256, respectively). Over the course of the simulation the mean mass resolution, as indicated by the peak of the distribution, increases by almost an order of magnitude relative to the initial mass resolution, though the distribution of cell masses is quite large. Figure 3 shows that the mean cell mass as a function of overdensity (at the end of the simulation run) stays fairly constant, which lower mean cell masses in underdense regions and higher mean cell masses in highly overdense regions (presumably due to the limitation on the number of levels of adaptive mesh refinement allowed). The mean cell mass over the entire density range is between $\sim 5-10$ times better than the starting mass resolution for all simulations. Runs with lower overdensity criteria for refinement have somewhat better mass resolution overall.

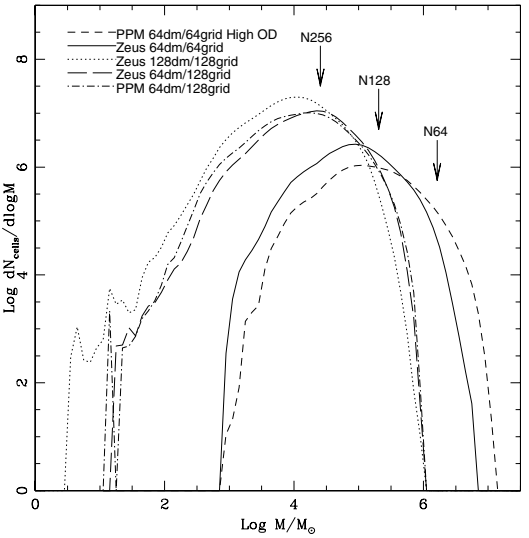


Fig. 3. Number of cells as a function of baryonic mass (normalized to bin size) for several Enzo simulations with $L_{box} = 3h^{-1}Mpc$. The arrows correspond to the mean mass resolution of the root grid for simulation volumes with the same volume but root grids with $64^3, 128^3$ or 256^3 cells (labelled N64, N128 and N256, respectively). See Figure 2 for a description of the line labels.

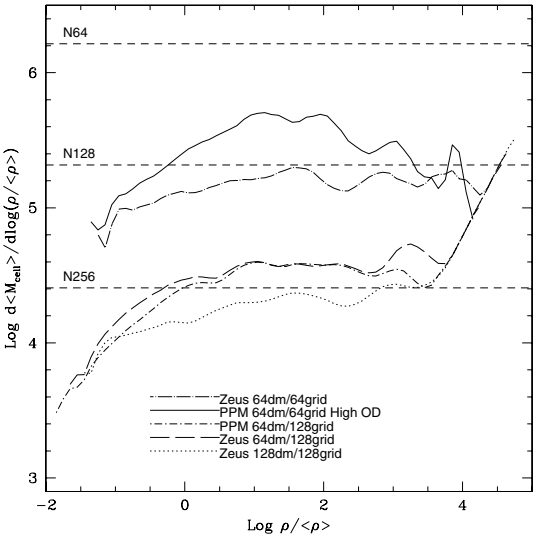


Fig. 4. Mean baryonic mass in cells as a function of baryon overdensity (normalized to bin size) in Enzo simulations. These are the same simulations (with the same labels) as in Figure 2. Horizontal lines correspond to the initial mean mass resolution of simulations with the same volume but $64^3, 128^3$ and 256^3 root grid cells (labelled N64, N128 and N256, respectively).

4 Summary

In this paper we have presented Enzo a cosmology code which combines collisionless N-body particle dynamics with a hydrodynamics package based on the piecewise parabolic method, all within a block-based adaptive mesh refinement algorithm. Several other physics packages are implemented, including multiple models for gas cooling and ionization, a uniform ultraviolet background model for gas heating, and a prescription for star formation and feedback.

Enzo is being released to the public as a community astrophysical simulation code. This code is being modified and documented to be as widely useful as possible, and can be found at `http://cosmos.ucsd.edu/enzo/`. Active development is taking place, centering on the addition of magnetohydrodynamics and a diffusive radiative transfer algorithm.

Further information concerning the performance of the Enzo code (including a package of performance monitoring and visualization tools) is described in a contribution by James Bordner in this volume, and on the Enzo website.

References

1. J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy & Astrophysics*, 30:543, 1992.
2. M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comp. Phys.*, 82:64–84, 1989.
3. G. L. Bryan and M. L. Norman. A hybrid amr application for cosmology and astrophysics. In N. Chrisochoides, editor, *Workshop on Structured Adaptive Mesh Refinement Grid Methods*, page 165. IMA Volumes in Mathematics No. 117, 2000.
4. G. L. Bryan and M. L. Norman. In D. A. Clarke and M. Fall, editors, *Computational Astrophysics; 12th Kingston Meeting on Theoretical Astrophysics, proceedings of meeting held in Halifax; Nova Scotia; Canada October 17-19; 1996*. ASP Conference Series # 123, 1997.
5. G. L. Bryan. Fluids in the universe: Adaptive mesh in cosmology. *Computing in Science and Engineering*, 1:2:46, 1999.
6. M. L. Norman and G. L. Bryan. Cosmological adaptive mesh refinement. In Kohji Tomisaka Shoken M. Miyama and Tomoyuki Hanawa, editors, *Numerical Astrophysics : Proceedings of the International Conference on Numerical Astrophysics 1998 (NAP98), held at the National Olympic Memorial Youth Center, Tokyo, Japan, March 10-13, 1998.*, page 19. Kluwer Academic, 1999.
7. P. R. Woodward and P. Colella. A piecewise parabolic method for gas dynamical simulations. *J. Comp. Physics*, 54:174, 1984.
8. G. L. Bryan, M. L. Norman, J. M. Stone, R. Cen, and J. P. Ostriker. A piecewise parabolic method for cosmological hydrodynamics. *Comp. Phys. Comm.*, 89:149–168, 1995.
9. J. M. Stone and M. L. Norman. Zeus-2d: A radiation magnetohydrodynamics code for astrophysical flows in two space dimensions. i - the hydrodynamic algorithms and tests. *ApJ*, 80:753, 1992.
10. J. M. Stone and M. L. Norman. Zeus-2d: A radiation magnetohydrodynamics code for astrophysical flows in two space dimensions. ii. the magnetohydrodynamic algorithms and tests. *ApJ*, 80:791, 1992.

11. T. Abel, P. Anninos, Y. Zhang, and M. L. Norman. Modeling primordial gas in numerical cosmology. *New Astronomy*, 2:181–207, August 1997.
12. P. Anninos, Y. Zhang, T. Abel, and M. L. Norman. Cosmological hydrodynamics with multi-species chemistry and nonequilibrium ionization and cooling. *New Astronomy*, 2:209–224, August 1997.
13. M. Rauch. The Lyman Alpha Forest in the Spectra of QSOs. *Annual Review of Astronomy and Astrophysics*, 36:267–316, 1998.
14. F. Haardt and P. Madau. Radiative Transfer in a Clumpy Universe. II. The Ultraviolet Extragalactic Background. *ApJ*, 461:20–+, April 1996.
15. R. Cen and J. P. Ostriker. Galaxy formation and physical bias. *ApJL*, 399:L113–L116, November 1992.

Toward Optimizing Enzo, an AMR Cosmology Application

James Bordner

University of California, San Diego

Summary. Enzo is a parallel hybrid SAMR / N-body code designed to simulate cosmological structure formation. This paper describes our approach to gathering and visualizing performance information from Enzo, which will be used to direct our subsequent modeling and optimization effort. Understanding the performance of AMR applications on distributed memory architectures is challenging, owing in part to the dynamic multilevel data structures and variety of communication patterns involved. To facilitate the task of measuring, modeling, and optimizing Enzo's performance, we are developing the Enzo Performance Monitoring System (EPMS). We review some existing performance tools, describe the EPMS, and show some preliminary performance data obtained using the EPMS.

1 Introduction

Enzo [1] is a 3D MPI-parallel structured AMR [2] / N-body cosmology application developed at the Laboratory for Computational Astrophysics, University of California, San Diego. The current version of Enzo can simulate a wide variety of astrophysical processes, including hydrodynamics, self-gravity, the evolution of up to twelve chemical species of H and He, radiative cooling, uniform ultraviolet background radiation, and star formation with feedback. Development is underway to further augment Enzo with magnetohydrodynamics and implicit radiation diffusion modules. A branch of the Enzo code is being prepared for public release, and is currently downloadable by friendly-users at `http://cosmos.ucsd.edu/enzo/`. A more detailed description of Enzo can be found in the paper by O'Shea et al. in this volume.

A necessary step in optimizing an application is to understand its existing performance. Because of the complexity of Enzo's software implementation, data structures, and communication requirements, as well as that of modern parallel hardware architectures, we are designing and implementing the *Enzo Performance Monitoring System* (EPMS). The EPMS, currently under development, is a collection of tools for obtaining, analyzing, and visualizing a wide variety of performance data. When completed, it will be used to help measure, model, and optimize Enzo's serial, parallel, and scaling performance.

First we review some existing parallel performance tools in §2, then we describe the design and implementation of the EPMS in §3, and finally we provide some example performance results in §3.3.

2 Some Existing Performance Tools

There are many software tools already available for probing the serial and parallel performance of applications. In this section we briefly review four of them: prof/gprof, Jumpshot, PAPI, and svPablo. A more detailed review of performance tools can be found in [3].

Perhaps the most well-known and widely-used performance tools are the Unix tool *prof* and its variants, including the GNU profiler *gprof*. Using them is easy: first an application is compiled with the appropriate compiler flag (typically -p or -pg), then the resulting profile-enabled application is run. Running the application generates a binary file of profile data, which the prof or gprof tool uses to generate a readable text summary of the application's performance. This summary includes how much time is spent in each function, and how many times each function is called. This basic timing information can be valuable when optimizing smaller serial programs; however, these tools do not directly support parallel applications, and they do not provide any further information that could help diagnose why certain functions are particularly costly. Furthermore, timing accuracy can suffer if the profile time intervals are too long, and application execution time can suffer if profile time intervals are too short. Nevertheless, the tools are very easy to use, and can quickly provide useful basic performance information.

Jumpshot [4] is a performance visualization tool developed at the Laboratory for Advanced Numerical Software, Argonne National Laboratory. Jumpshot reads event traces generated by the MPI[1] profiling interface MPE. It displays time-lines of MPI calls, and can indicate matching sends and receives. Jumpshot's scope is restricted to MPI communication performance, and is not designed to match MPI calls to their location in the source code, let alone differentiate between the same MPI functions called from different modules. Nevertheless, Jumpshot is relatively easy to use, can be useful for visualizing communication patterns in an application, and for quickly spotting possible communication-related performance problems.

The Performance Application Programming Interface, or *PAPI*[2], is a software library developed at the Innovative Computing Laboratory, University of Tennessee, Knoxville. PAPI is designed to provide a portable interface to hardware performance counters. Typical performance counters available on modern processors include floating-point operation counts, cache miss or hit counts, memory load and store counts, etc. PAPI thus provides a powerful low-level interface to a wide range of serial performance metrics, and is an important component of the Enzo Performance Monitoring System.

[1] http://www.mcs.anl.gov/mpi/

[2] http://icl.cs.utk.edu/projects/papi/

Lastly, *svPablo* [5] is a larger-scale performance visualization package developed by the Pablo Research Group, University of Illinois at Urbana-Champaign. It is portable; can be used to interactively instrument C, Fortran 77, or Fortran 90 code using a GUI interface; can correlate performance metrics (including PAPI events if PAPI is available) back to source code lines in a graphical source browser; and can recognize calls to MPI, MPI I/O, Unix I/O, and HDF I/O. Although svPablo does not currently support C++ applications such as Enzo, the developers intend to support C++ in the near future. Of the performance tools reviewed here, it comes closest to providing the wide range of functionality we envisioned for the Enzo optimization project.

3 The Enzo Performance Monitoring System

The Enzo Performance Monitoring System (EPMS) is a software system designed for obtaining and visualizing a wide range of performance-related metrics, including those based on PAPI or user-defined counters. Care is being taken in its design and implementation to produce a software product that is both easy to use and applicable to other applications, while not compromising on functionality.

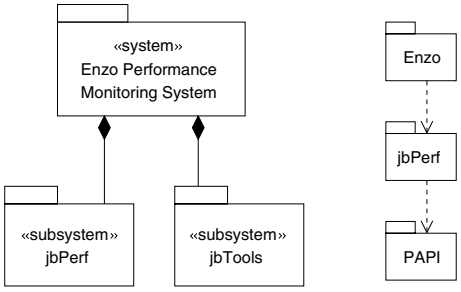


Fig. 1. Enzo Performance Monitoring System.

The EPMS is composed of two subsystems, jbPerf and jbTools, as shown in the diagram on the left in Figure 1. jbPerf is a performance API called by Enzo for collecting performance data, then writing the data in trace files; jbTools is a set of command-line utilities for post-processing trace files generated by jbPerf. As indicated on the right in Figure 1, jbPerf may in turn call PAPI if it is available.

Before describing jbPerf and jbTools in more detail, we first discuss in §3.1 the range of performance issues the EPMS is being designed to support. These targeted performance issues, both software- and hardware-related, will help provide context for understanding the design decisions made during the development of jbPerf and jbTools. jbPerf will be described in §3.2, and jbTools in §3.3.

3.1 Targeted Performance Issues

While the overall time to solution is perhaps the most important performance metric, determining it a priori for a given software application, hardware architecture, and problem configuration requires detailed knowledge of the software's performance characteristics and the hardware architecture. In this section we review some of the main performance issues, from both hardware and software perspectives.

Concurrent Hardware Performance Issues

Three major potential sources of inefficiency when running a parallel application on an SMP cluster architecture are *load balancing*, *communication*, and *storage access*. These are illustrated in Figure 2.

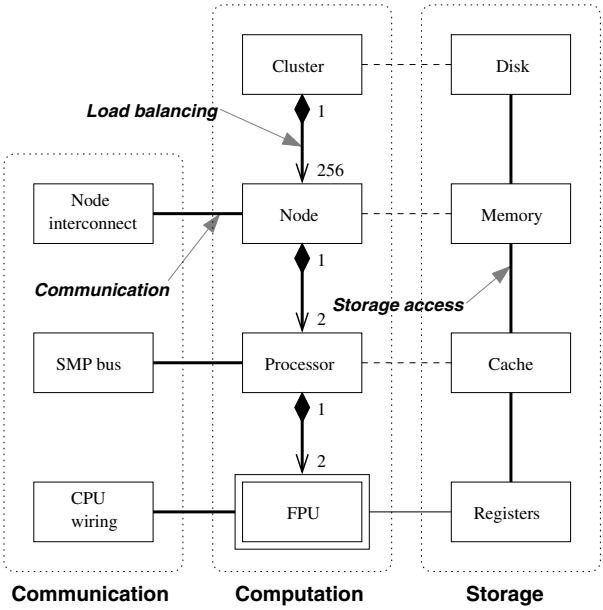


Fig. 2. Potential sources of performance inefficiency.

The center column in Figure 2 represents the compositional layering of computational components in an SMP cluster, from the cluster level at the top, down to the floating-point unit level at the bottom. For concreteness, specific multiplicities are shown for the NSF TeraGrid[3] cluster at NCSA: each of that cluster's 256 nodes contains two processors, and each processor contains two floating-point units. The right column represents the storage subsystem, with storage components roughly

[3] http://www.teragrid.org/

matched with the corresponding computational components. The left column represents the communication subsystem, again with its components roughly matched with the corresponding computational components. Within each subsystem is a potential for loss of performance: computational load balancing inefficiency, communication overhead, and memory storage access times.

Load balancing is an issue at multiple architectural levels, but is most important at the node level, since that level has the highest degree of parallelism. Enzo currently uses a dynamic load balancing scheme that relocates real grids from heavily loaded processors to lightly loaded processors. More sophisticated dynamic load balancing schemes have been developed by Z. Lan [6] [7], which will likely be introduced into a later version of the public code branch of Enzo. Also, a static load balancing scheme is being developed for use with Enzo by Liebrock et al., based on Liebrock's Ph.D. thesis work [8].

Communication overhead is inevitable in most parallel applications, and can easily dominate computation. As with load balancing, communication overhead is most significant at the node level. Specific communication patterns can lead to reduced efficiencies on some communication systems due to "hot spots" in node interconnect topologies; these may be difficult or impossible to predict a priori based only on the interconnect's nominal latency and bandwidth. Approaches to lowering communication overhead can include reordering MPI sends and receives to decrease latency, using nonblocking MPI calls to overlap communication and computation, enhancing the load balancing scheme to improve grid locality, redesigning algorithms to reduce the amount of communication needed, or using different algorithms altogether.

Accessing memory and disk can reduce performance due to both the latency and bandwidth limitations of the higher-level storage components. As with communication, memory access efficiencies can be difficult to estimate a priori due to hardware details of the cache hierarchy. Approaches to optimizing storage access speeds can include prefetching, reordering grid computations into smaller blocks to increase cache reuse, reordering data to improve spatial and temporal locality, reformulating algorithms to use cache-efficient high-level BLAS routines, and redesigning algorithms to have higher ratios of floating-point operations to memory references.

These three potential sources of inefficiency—load balancing, communication overhead, and storage access—are the main sources of hardware-related performance inefficiencies targeted by the EPMS.

AMR Application Performance Issues

Three performance issues arising from an AMR application perspective that we are concerned with are *code components*, AMR *hierarchy levels*, and *simulation timesteps*. While perhaps not the only axes of interest, we nevertheless consider them necessary for a thorough understanding of an AMR application's overall performance.

Different physics components and data structure manipulation functions in Enzo can have very different performance characteristics. For example, the computational, memory, and communication requirements for the hydrodynamics solver are different from those of the gravity solver, which in turn are different from those of the mesh

hierarchy rebuilding component, etc. To better understand the performance and scaling properties of Enzo as a whole, it helps to understand the performance and scaling properties of its individual components.

Since Enzo is an AMR code, in particular one that takes multiple time steps on deeper AMR levels, performance characteristics can vary depending on the AMR level. Understanding how performance characteristics change between different AMR levels can be helpful in predicting performance for an Enzo run with a known static distribution of grids across multiple levels of refinement.

Finally, since the AMR hierarchy is dynamic, to predict the performance of an entire Enzo simulation, we need to understand how performance properties change as Enzo's mesh hierarchy evolves. We therefore wish to collect performance data for each time-step.

3.2 Collecting Performance Data Using jbPerf

jbPerf is the API component of the EPMS. It consists of a single C++ class that was designed to collect a wide variety of performance information from a running application. This performance data can include wall-clock (real) time, CPU (virtual) time, hardware counters accessed through the optional PAPI library, and user-defined and controlled counters, which can be used for collecting performance data related to communication and disk I/O.

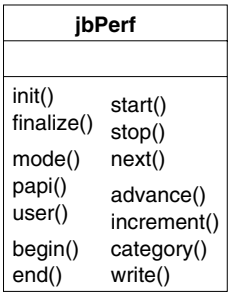


Fig. 3. The jbPerf class.

The core capability of jbPerf is to instrument regions of code using concise jb::perf.start(*region*) and jb::perf.stop(*region*) calls. These calls can be nested, and both inclusive and exclusive performance data are collected for each region. These functions are used to collect cumulative performance data for individual code components in Enzo.

For regions with dynamically changing performance characteristics, the function jb::perf.next(*region*) can be inserted immediately after the call to stop(). Each time next() is called, a new set of counters is created and initialized for the associated region. The related function jb::perf.advance() is also provided, which effectively calls next() globally for all known regions. In Enzo, the advance() function is called after

each time-step in a simulation. Most event tracing performance libraries implicitly call next() after each stop(); by making the call explicit and optional, the lengths of performance traces can be reduced substantially.

Most code regions instrumented in Enzo are in the main time stepping loop, and are called for all levels. To partition performance data among separate levels, the jb::perf.category(*category*) function was added to jbPerf. This function essentially modifies all subsequent code region names by appending the *category* string. In Enzo, category() is used to create a separate set of regions for each level, without requiring modifications to the *region* arguments of start() and stop().

papi(), user(), and increment() are used to control the PAPI and user counters in jbPerf. The jb::perf.papi(*papi-counter*) function can be called at the beginning of a program to activate a PAPI event. For example, papi("fp-ins") will cause subsequent start() and stop() calls to count floating-point operations for regions, papi("lst-ins") will count memory load and store instructions, etc. user() is used to create a user-defined counter, as in jb::perf.user("hdf-write-bytes"), and increment() is used to increment a user counter, as in jb::perf.increment("mpi-send-bytes",num_bytes). Thus, by providing user counters, applications can monitor performance metrics that may be application-specific or otherwise unavailable, at the cost of having to manually increment the counters.

Lastly, the function write() is used to dump all data collected so far to disk files. One file is created for each processor, region, and category combination. Files contain sequences of counts (both inclusive and exclusive) for each active PAPI and user event, plus other basic performance information such as time stamps and call counts.

3.3 Processing and Visualizing Performance Data Using jbTools

Whereas jbPerf is used to collect performance data as Enzo is run, jbTools will be used to post-process and visualize the collected data.

Although jbTools is still under active development, it will eventually contain command-line utilities for generating plots of Enzo performance and scaling data. Currently implemented utilities include jb-extract for selecting a counter for a region, jb-add, jb-mul, etc. for performing basic arithmetic operations on sets of counter data, jb-mflop for computing mflop rates, and jb-load for computing an estimate of load balancing efficiencies.

Figure 4 illustrates the types of plots we would like jbTools to generate semi-automatically. Although the plots themselves were not produced directly by jbTools, they were nevertheless generated from jbPerf output files using existing lower-level jbTools utilities for processing the data. Actual plots were created manually using the GNU plotting program gnuplot. The test problem used an initial root grid of size $N = 128^3$, and was run on eight processors of SDSC's[4] TeraGrid Intel Itanium 2 Linux cluster.

Plotted on the left in Figure 4 is the relative time per time-step spent in the parallel FFT solver, which is used to compute the gravitational potential on the coarsest-level

[4] http://www.sdsc.edu/

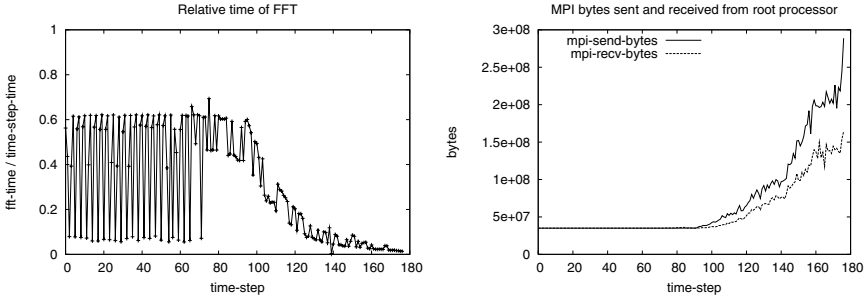


Fig. 4. Sample Enzo performance data collected using jbPerf.

grids. Several observations are immediately apparent, including the reduction of the relative times for later time-steps, the relatively large percentage of time spent in the FFT during the earlier time-steps, and the bimodal distribution of relative times during the earlier time-steps. While the cause of the first observation is known[5], the second and third observations require further investigation. Once jbTools is fully functional, it is expected to be a powerful tool for quickly investigating just such performance-related questions.

Plotted on the right in Figure 4 are the numbers of bytes sent and received during each time-step, relative to the root processor. As the AMR hierarchy begins to grow, we see an increase in communication traffic as expected. Again, once jbTools is fully functional, it should help facilitate the task of investigating observed performance behavior, such as why there are more sends than receives, or whether the floating-point operation rates are reduced significantly by the increased communication.

4 Conclusions

Although the EPMS is being designed specifically for Enzo, care is being taken to keep it independent of Enzo, and as easy to use as possible. The design of jbPerf has more-or-less converged to a steady-state, and the source code is expected to be released soon at `http://cosmos.ucsd.edu/jbPerf/`, after some further code refactoring. The jbTools subsystem will continue to be developed, and will also be released publicly. After we begin to obtain more substantial performance and scaling information about Enzo, results will be posted at `http://cosmos.ucsd.edu/enzo-perf/`.

[5]This is due to the generation of more and more grids in the AMR hierarchy as the simulation progresses, making the relative cost of the coarse-level based FFT less and less.

5 Acknowledgements

Optimizing Enzo for the TeraGrid is funded by a grant from the National Science Foundation through the National Computational Science Alliance's[6] "Community Codes Expedition."

References

1. G. Bryan and M.L. Norman, A Hybrid AMR Application for Cosmology and Astrophysics, in Workshop on Structured Adaptive Mesh Refinement Grid Methods, N. Chrisochoides (ed), March 1997
2. Berger and P. Collela, Local Adaptive mesh refinement for Shock Hydrodynamics, J. Comp. Phys., 82:64–84, 1989
3. Daniel A. Reed and Ruth A. Aydt, Tools for Performance Tuning and Debugging, in Sourcebook of Parallel Computing, J. Dongarra (ed), I. Foster, G. Fox (ed), W. Gropp, K. Kennedy, L. Torczon and A. White (ed), Morgan Kaufman Publishers, 2003.
4. O. Zaki, E. Lusk, W. Gropp, and D. Swider, Toward Scalable Performance Visualization with Jumpshot, Int'l J. of High Performance Computing Applications, 1999.
5. Luiz DeRose and Daniel A. Reed, SvPablo: A Multi-Language Architecture-Independent Performance Analysis System, Proceedings of the International Conference on Parallel Processing (ICPP'99), Fukushima, Japan, September 1999.
6. Z. Lan, V. Taylor, and G. Bryan, A Novel Dynamic Load Balancing Scheme for Parallel Systems, Journal of Parallel and Distributed Computing, Vol 62/12, pp.1763-1781, 2002.
7. Z. Lan, V. Taylor, and G. Bryan, Dynamic Load Balancing of SAMR applications on Distributed Systems, Journal of Scientic Programming, Vol 10(4), pp. 319-328, 2002.
8. Lorie M. Liebrock, Using Problem Topology in Parallelization, Rice University Technical Report CRPC-TR94477-S, September 1994.

[6] http://www.ncsa.uiuc.edu/Projects/Alliance/.

Construction and Application of an AMR Algorithm for Distributed Memory Computers

Ralf Deiterding

California Institute of Technology,1200 East California Blvd., Mail-Code 158-79, Pasadena, CA 91125, ralf@cacr.caltech.edu

While the parallelization of blockstructured adaptive mesh refinement techniques is relatively straight-forward on shared memory architectures, appropriate distribution strategies for the emerging generation of distributed memory machines are a topic of on-going research. In this paper, a locality-preserving domain decomposition is proposed that partitions the entire AMR hierarchy from the base level on. It is shown that the approach reduces the communication costs and simplifies the implementation. Emphasis is put on the effective parallelization of the flux correction procedure at coarse-fine boundaries, which is indispensable for conservative finite volume schemes. An easily reproducible standard benchmark and a highly resolved parallel AMR simulation of a diffracting hydrogen-oxygen detonation demonstrate the proposed strategy in practice.

1 Introduction

The adaptive mesh refinement (AMR) method after Berger and Collela [BC88] is widely used for adaptive simulations on logically rectangular finite volume meshes. Instead of replacing single cells by finer ones, the AMR method constructs a hierarchy of properly nested refinement grids. The striking efficiency of this algorithm, in particular for instationary supersonic gas dynamical problems, was demonstrated by Berger and her collaborators in [BBS94].

Up to now, various reliable implementations of the AMR method for single processor computers have been developed [BL98, CW93]. Even implementations for parallel computers with shared memory architecture have reached a stable state [BBS94]. Parallelism is an inherent feature of the AMR algorithm and in a shared memory environment simply the numerical solution on the whole sequence of grids has to be advanced in parallel to achieve a sufficient load-balancing. The question for an efficient parallelization strategy becomes more delicate for distributed memory machines, because the costs of communication can not be neglected anymore. Due to the technical difficulties in implementing hierarchical adaptive methods in a

distributed memory environment only few parallelization efforts are documented, cf. [RBL00, PB97, KB95].

This paper describes a rigorous domain decomposition approach that partitions the entire hierarchy from the base level on. By employing ghost or halo cell regions, which are synchronized whenever the algorithm applies boundary conditions, an overlap between subgrids is constructed that allows most operations of the AMR algorithm to be carried out strictly local. After a brief characterization of the employed finite volume methods in Sec. 2, we review the sequential AMR algorithm in Sec. 3. In Sec. 4, we specify the domain decomposition and discuss the necessary extensions of the previously described subroutines in parallel. Section 5 presents parallel AMR simulations for the Euler equations of gas dynamics obtained with our public domain code AMROC [DA03] on typical Linux-Beowulf-clusters.

2 Finite Volume Schemes

The Berger-Collela AMR method is a dynamic mesh adaptation approach, which is tailored especially for the adaptive numerical solution of hyperbolic conservation laws

$$\partial_t \vec{q}(\vec{x},t) + \nabla \cdot \vec{f}(\vec{q}(\vec{x},t)) = \vec{0}\,,\ \vec{x} = (x_1,\ldots,x_d)^T \in \mathbb{R}^d\,,\ t \in \mathbb{R}_0^+ \tag{1}$$

on logically rectangular finite volume (FV) meshes. For simplicity, we restrict our attention to the two-dimensional case and assume an equidistant FV discretization of the computational domain $G_0 \subset \mathbb{R}^2$ with mesh widths Δx_1, Δx_2 and a constant time step Δt. The discrete mesh points are defined by $(x_1^i, x_2^j) := \left(\left(i+\frac{1}{2}\right)\Delta x_1, s\left(j+\frac{1}{2}\right)\Delta x_2\right)$, $i,j \in \mathbb{Z}$ and $t_\kappa := \kappa\Delta t$, $\kappa \in \mathbb{N}_0$. In each point (x_1^i, x_2^j, t^κ) we define a discrete value $\vec{Q}_{ij}^{\kappa}$ as an approximation to the vector of state $\vec{q}(\vec{x},t)$ averaged over the control volume $[x_1^{i-1/2}, x_1^{i+1/2}) \times [x_2^{j-1/2}, x_2^{j+1/2})$. These values are updated by a time-explicit *conservative* $(2s+1)^2$-point FV scheme of the form

$$\mathcal{H}^{(\Delta t)}:\ \vec{Q}_{ij}^{\kappa+1} = \vec{Q}_{ij}^{\kappa} - \frac{\Delta t}{\Delta x_1}\left(\vec{F}^1_{i+\frac{1}{2},j} - \vec{F}^1_{i-\frac{1}{2},j}\right) - \frac{\Delta t}{\Delta x_2}\left(\vec{F}^2_{i,j+\frac{1}{2}} - \vec{F}^2_{i,j-\frac{1}{2}}\right) \tag{2}$$

with *numerical fluxes* given by

$$\vec{F}^1_{i+\frac{1}{2},j} = \vec{F}_1(\vec{Q}^{\kappa}_{i-s+1,j-s},\ldots,\vec{Q}^{\kappa}_{i+s,j+s}), \vec{F}^2_{i,j+\frac{1}{2}} = \vec{F}_2(\vec{Q}^{\kappa}_{i-s,j-s+1},\ldots,\vec{Q}^{\kappa}_{i+s,j+s}).$$

For vanishing boundary fluxes, scheme (2) satisfies the important discrete conservation property $\sum_{i,j}\vec{Q}_{jk}^{\kappa+1} = \sum_{i,j}\vec{Q}_{jk}^{\kappa}$, which is essential for the correctness of the approximation, if Eq. (1) admits discontinuous solutions, as it is the case e.g. for Euler equations. The numerical fluxes in (2) are often evaluated by solving a Riemann problem between neighboring cells approximately. In this case, a typical stability condition for $\mathcal{H}^{(\Delta t)}$ could be

$$C_{CFL} := \max_{j,k}\left(S_{j+\frac{1}{2},k}\frac{\Delta t}{\Delta x_1}, S_{j,k+\frac{1}{2}}\frac{\Delta t}{\Delta x_2}\right) \le 1\,, \tag{3}$$

where $S_{j+\frac{1}{2},k}$, $S_{j,k+\frac{1}{2}}$ denote the maximal signal speeds in both space directions according to the approximative solution of the Riemann problems at the cell interfaces.

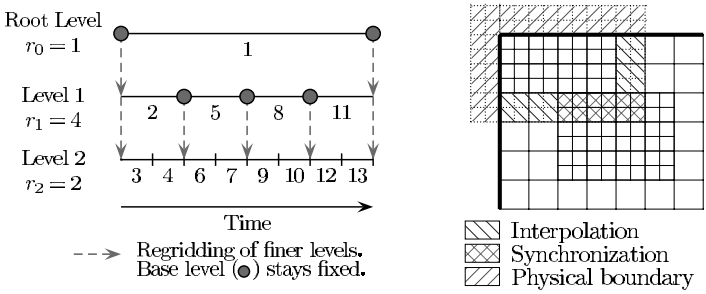


Fig. 1. Left: Recursive integration order. Right: Sources of ghost cell values.

3 Blockstructured Adaptive Mesh Refinement

A significant advantage of the blockstructured idea over other mesh refinement strategies is that the update operator $\mathcal{H}^{(\cdot)}$ only needs to be implemented on a single *uniform* Cartesian grid G, where s layers of auxiliary cells (ghost or halo cells) around G should be employed to define discrete boundary conditions. Cells flagged by various error indicators are clustered dynamically at run-time into non-overlapping rectangular subgrids $G_{l,m}$ that define the domain of an entire level $l = 0, \ldots, l_{\max}$ by

$$G_l := \bigcup_{m=1}^{M_l} G_{l,m} \, .$$

Refinement grids are derived recursively from coarser ones and a hierarchy of successively embedded levels is thereby constructed. All mesh widths on level l are r_l-times finer than on level $l-1$, i.e. $\Delta t_l := \Delta t_{l-1}/r_l$ and $\Delta x_{n,l} := \Delta x_{n,l-1}/r_l$ with $r_l \in \mathbb{N}, r_l \geq 2$ for $l > 0$ and $r_0 = 1$, and a time-explicit FV scheme (in principle) remains stable under a condition like (3) on all levels of the hierarchy. The recursive integration order visualized in the left sketch of Fig. 1 is an important difference to unstructured adaptation strategies and is one of the main reasons for the high efficiency of the approach.

The numerical scheme is applied on level l by calling the single-grid routines $\mathcal{H}^{(\cdot)}$ in a loop over all subgrids $G_{l,m}$. The execution of the numerical loop in `UpdateLevel()` in Alg. 1 requires the previous setting of the ghost cell values. Three types of boundary conditions have to be considered in the sequential case, see right sketch of Fig. 1. Cells outside of the root domain G_0 are used to implement physical boundary conditions. Ghost cells in G_l have a unique interior cell analogue and are set by copying the data value from the grid, where the interior cell is contained (synchronization). On the root level no further boundary conditions need to be considered, but for $l > 0$ also internal boundaries can occur. They are set by a conservative time-space interpolation from two previously calculated time steps of level $l-1$.

Beside a general data tree that stores the topology of the hierarchy, the AMR method requires at most two regular arrays assigned to each subgrid, which contain the discrete vector of state $\vec{Q}$ for the actual and updated time step. In the following,

we denote by $\vec{Q}^l(t)$ and $\vec{Q}^l(t+\Delta t_l)$ the unions of these arrays on level l. The regularity of the input data for the numerical routines allows high performance on vector and super-scalar processors and cache optimizations. Small data arrays are effectively avoided by leaving coarse level data structures untouched, when higher level grids are created. Values of cells covered by finer subgrids are overwritten by averaged fine grid values subsequently.

3.1 Conservative Flux Correction

Replacing coarse cell values by averaged fine grid values modifies the numerical stencil on the coarse grid. In general the important property of conservation is lost. A flux correction is necessary that introduces the involved fine grid fluxes into Eq. (2). In two and three space dimensions hanging nodes additionally have to be considered. As an example we consider cell (j,k) in Fig. 2 on level l. After the numerical update

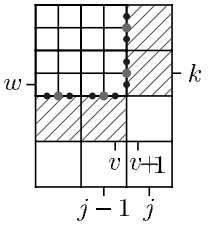


Fig. 2. Location of numerical fluxes required for flux correction. Cells to correct are shaded.

on level l a correction term associated to the boundary of level $l+1$ is initialized by $\delta\vec{F}^{1,l+1}_{j-\frac{1}{2},k} := -\vec{F}^{1,l}_{j-\frac{1}{2},k}$. During the r_{l+1} update steps of level $l+1$ all necessary fine level fluxes are accumulated, i.e.

$$\delta\vec{F}^{1,l+1}_{j-\frac{1}{2},k} := \delta\vec{F}^{1,l+1}_{j-\frac{1}{2},k} + \frac{1}{r_{l+1}^2}\sum_{\nu=0}^{r_{l+1}-1}\vec{F}^{1,l+1}_{v+\frac{1}{2},w+\nu}(t+\mu\Delta t_{l+1}) \tag{4}$$

with $\mu = 0,\ldots,r_{l+1}-1$. After the integration of the fine level, the correction is applied by calculating

$$\vec{Q}^{\kappa+1}_{jk} := \vec{Q}^{\kappa+1}_{jk} + \frac{\Delta t_l}{\Delta x_{1,l}}\,\delta\vec{F}^{1,l+1}_{j-\frac{1}{2},k}\,.$$

The edge- or face-centered flux correction terms $\delta\vec{F}^{n,l+1}$ have to be stored along the boundaries, where a level $l>0$ abuts the next coarser level. To avoid the usage of the numerical fluxes $\vec{F}^n$ on the entire level, the grid-wise application of the numerical scheme and the computation of the correction terms are effectively combined in `UpdateLevel()` in the loop over all all grids on level l.

```
AdvanceLevel(l)
 Repeat r_l times
   Set ghost cells of Q^l(t)
   If time to regrid
     Regrid(l)
   UpdateLevel(l)
   If level l+1 exists
     Set ghost cells of Q^l(t+Δt_l)
     AdvanceLevel(l+1)
     Average Q^{l+1}(t+Δt_l) onto
       Q^l(t+Δt_l)
     Correct Q^l(t+Δt_l) with δF^{n,l+1}
   t := t+Δt_l
```

Alg. 1. Recursive AMR algorithm.

```
Regrid(l)
 For ι = l_c Downto l Do
   Flag N^ι according to Q^ι(t)
   If level ι+1 exists?
     Flag N^ι below Ğ_{ι+2}
   Flag buffer zone on N^ι
   Generate Ğ_{ι+1} from N^ι
 Ğ_l := G_l
 For ι = l To l_c Do
   CĞ_ι := G_0\Ğ_ι,  Ğ_{ι+1} := Ğ_{ι+1}\CĞ_ι^1
 Recompose(l)
```

Alg. 2. Regridding procedure.

3.2 Recursive Grid Generation

The basic recursive AMR algorithm is formulated in Alg. 1. Except the regridding procedure, all operations have already been explained. New refinement grids on all higher levels are created by calling `Regrid()` from level l. Level l by itself is not modified. To consider the nesting of the level domains already in the grid generation, Alg. 2 starts at the highest refineable level l_c, where $0 \le l_c < l_{\max}$. The refinement flags are stored in grid-based integer arrays N^ι. A clustering algorithm [BBS94] is necessary to create a new refinement $\breve{G}_{\iota+1}$ on basis of N^ι until the ratio between flagged and all cells in every new grid $\breve{G}_{\iota+1,m}$ is above a prescribed threshold $0 < \eta_{tol} < 1$.

Before the new grids $\breve{G}_{\iota+1}$ can be used to replace $G_{\iota+1}$, the proper nesting of the new refinement grids has to be enforced over the modified hierarchy. In Alg. 2 we evaluate the invalid region for level $\iota+1$ by calculating the complement $C\breve{G}_\iota := G_0 \backslash \breve{G}_\iota$ of the next coarser level domain $\breve{G}_\iota$ in G_0 and by enlarging $C\breve{G}_\iota$ by one additional layer of cells on level ι. We denote this enlarged region by $C\breve{G}_\iota^1$. The operation $\breve{G}_{\iota+1} := \breve{G}_{\iota+1} \backslash C\breve{G}_\iota^1$ then eliminates all invalid regions from the new level domain $\breve{G}_{\iota+1}$.

The reinitialization of the hierarchy is done in the subroutine `Recompose(l)`, which is formulated in Alg. 3a. In particular, grid-based auxiliary data $\vec{\breve{Q}}(\breve{G}_\iota, t)$ is necessary to reorganize the vector of state. Cells in newly refined regions $\breve{G}_\iota \backslash G_\iota$ are initialized by interpolation, values of cells in $\breve{G}_\iota \cap G_\iota$ are copied. As interpolation requires the previous synchronized reorganization of $\vec{Q}^{\iota-1}(t)$, the recomposition algorithm traverses the hierarchy upwards.

4 Parallelization by Domain Decomposition

We follow a rigorous domain decomposition approach and partition the AMR hierarchy from the root level on. We assume a parallel machine with P identical nodes

and split the root domain G_0 into P non-overlapping portions G_0^p, $p = 1,\ldots,P$ by

$$G_0 = \bigcup_{p=1}^{P} G_0^p \quad \text{with} \quad G_0^p \cap G_0^q = \varnothing \ \text{ for } p \neq q \,.$$

The key idea now is that all higher level domains are required to follow the decomposition of the root level, i.e.

$$G_l^p := G_l \cap G_0^p \,. \tag{5}$$

Condition (5) can cause the splitting of a subgrid $G_{l,m}$ into multiple subgrids on different processors. Under requirement (5) we estimate the work on an arbitrary subdomain $\Omega \subset G_0$ by

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\nu=0}^{l} r_\nu \right] \,. \tag{6}$$

Herein, $\mathcal{N}_l(\cdot)$ denotes the total number of FV cells on level l in the given domain. The product in (6) is used to consider the time step refinement. A nearly equal distribution of the work necessitates

$$\mathcal{L}^p := \frac{P \cdot \mathcal{W}(G_0^p)}{\mathcal{W}(G_0)} \approx 1 \quad \text{for all } p = 1,\ldots,P \,. \tag{7}$$

A considerable advantage of the proposed decomposition strategy is the reduction of the communication costs. Together with the use of ghost cells our approach allows an almost local execution of Alg. 1. In particular, the inter-level operations interpolation and averaging remain strictly local. The only parallel operations that have to be considered additionally are the parallel ghost cell synchronization and the application of flux correction terms across processor borders. Especially `UpdateLevel()` need not be modified. Apparently, the new vector of state on each subgrid $G_{l,m}^p$ and the fluxes can be computed strictly local, but also the evaluation of the correction terms does not require communication.

4.1 Local Calculation of Flux Corrections

To illustrate this, we assume a parallel border in Fig. 2 at $j - \frac{1}{2}$. Let cell (j,k) be contained in G_l^q and let cell (v,w) be contained in G_{l+1}^p. Then the necessary correction term $\delta\vec{F}_{j-1/2,k}^{1,l+1}$ resides on node p, because it is assigned to the fine level. Its initialization requires the coarse grid flux $\vec{F}_{j-1/2,k}^{1,l}$. This flux is available on node p, because the basic AMR strategy ensures that below (v,w) an interior coarse cell $(j-1,k)$ exists having $\vec{F}_{j-1/2,k}^{1,l}$ as flux into a ghost cell (j,k). On the other hand, $\vec{F}_{j-1/2,k}^{1,l}$ is also computed on node q, where (j,k) is interior and $(j-1,k)$ is a ghost cell. As the ghost cells have been synchronized before the numerical update, the same boundary flux is calculated on both nodes, cf. Fig. 3. The fine grid fluxes $\vec{F}_{v+1/2,w+\nu}^{1,l+1}$ are only available on p, because no abutting interior fine grid cell exists on q. As the correction term $\delta\vec{F}_{j-1/2,k}^{1,l+1}$ is also stored on p the summation in (4) remains local. The only operation

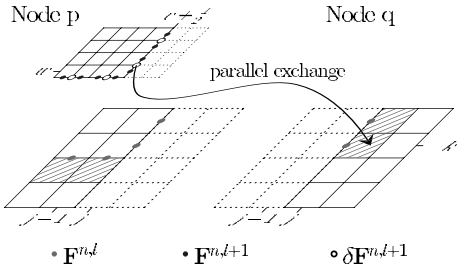


Fig. 3. Flux correction in parallel.

of the flux correction that necessarily requires communication is the final application of the correction terms as mentioned in the previous section. But the communication costs are minor, because the corrections are only necessary along lower-dimensional domains.

4.2 Parallel Grid Generation

Analogous to Alg. 1 the regridding in Alg. 2 is hardly affected by the parallelization. The flagging of cells on each level can be done locally. If a refinement criterion requires auxiliary time steps (i.e. error estimation by Richardson extrapolation, cf. [BC88]), additional synchronizations will be necessary, but this does not affect Alg. 2. The only operation in Alg. 2 that needs special attention is the clustering.

The clustering algorithm could be executed strictly locally on $N(G_\iota^p)$ or it could be executed on the data of the entire level $N(G_\iota)$. Usually, the results will be identical for $\eta_{tol} = 1$ only. To avoid the expensive global concatenation of all data sets $N(G_\iota^p)$ to $N(G_\iota)$, we execute the clustering algorithm strictly locally and communicate just the results $\breve{G}_{\iota+1}^p$ to obtain the global list $\breve{G}_{\iota+1} = \bigcup_p \breve{G}_{\iota+1}^p$. To consider a buffer zone of b cells before local clustering, the grid-based integer arrays N^ι are extended by b ghost cells. A parallel synchronization of these ghost cells before creating the buffer zone locally ensures the appropriate flagging of interior cells.

The main changes in the regridding procedure are in `Recompose(`l`)`. Instead of Alg. 3a we apply Alg. 3b. Due to our distribution strategy we now have to consider a complete reorganization of the entire hierarchy even for a regridding at a higher level. In particular, the whole relevant data of levels with $\iota \leq l$ have to be copied. Like the synchronization operation, these copy operations are partially local and parallel. For levels with $\iota < l$ the relevant data is $\vec{Q}^\iota(t)$, $\vec{Q}^\iota(t+\Delta t_\iota)$ and $\delta\vec{F}^{n,\iota}$, for level l we have to copy $\vec{Q}^l(t)$ and $\delta\vec{F}^{n,l}$. The initialization of a level with $\iota > l$ is in principle identical to Alg. 3a. Interpolation is a strictly local operation, provided that the next coarser level has already been reorganized. The copy operation is a combination of local and parallel copy.

Alg. 3b is significantly more complex than Alg. 3a, because it considers the general case of a complete parallel redistribution of the AMR hierarchy even at higher level time steps. However, in practice it usually suffices to allow this operation only

```
Recompose(l) - sequential

For ι = l+1 To l_c+1 Do
   Interpolate Q⃗^{ι-1}(t) on Q⃗̃^ι(t)
   Copy Q⃗^ι(t) on Q⃗̃^ι(t)
   Set ghost cells of Q⃗̃^ι(t)
   Q⃗^ι(t) := Q⃗̃^ι(t)
   G_ι := Ğ_ι
```

Alg. 3a. Sequential recomposition.

```
Recompose(l) - parallel
  Derive G_0^p from
        {G_0,...,G_l, Ğ_{l+1},...,Ğ_{l_c+1}}
  For ι = 0 To l_c+1 Do
    If ι > l
      Ğ_ι^p := Ğ_ι ∩ G_0^p
      Interpolate Q⃗^{ι-1}(t) on Q⃗̃^ι(t)
    else
      Ğ_ι^p := G_ι ∩ G_0^p
      If ι > 0
        Copy δF⃗^{n,ι} onto δF⃗̃^{n,ι}
        δF⃗^{n,ι} := δF⃗̃^{n,ι}
    If ι ≥ l then ν_ι = 0 else ν_ι = 1
    For ν = 0 To ν_ι Do
      Copy Q⃗^ι(t+νΔt_ι) on Q⃗̃^ι(t+νΔt_ι)
      Set ghost cells of Q⃗̃^ι(t+νΔt_ι)
      Q⃗^ι(t+νΔt_ι) := Q⃗̃^ι(t+νΔt_ι)
    G_ι^p := Ğ_ι^p ,  G_ι := ⋃_p G_ι^p
```

Alg. 3b. Parallel recomposition.

on the root level. Under this simplification, Alg. 3b reduces mostly to Alg. 3a. The creation of the new load-balanced distributions G_0^p then has to be considered just for the case $l = 0$ and only $\vec{Q}^\iota(t)$ has to be copied over processor borders.

4.3 Partitioning

It is evident, that the overall efficiency of the chosen parallelization strategy depends especially on the first step of Algorithm 3b, the partitioning algorithm. This algorithm has to meet several requirements. It must balance the estimated workload, while the parallel synchronization costs should be small. A slight change of the hierarchy should require only a moderate data redistribution. The algorithm must be fast, because it is carried out on-the-fly.

Distribution strategies based on space-filling curves seem to give an acceptable compromise between these partially competing requirements. A space-filling curve defines a continuous mapping from $[0,1]$ onto $[0,1]^d$ and is well suited to define an ordered sequence on the root level cells of a blockstructured domain. This sequence can easily be split into portions of equal size yielding load-balanced new distributions G_0^p. As space-filling curves are constructed recursively, they are locality-preserving by definition and naturally avoid an excessive redistribution overhead. Further on, the surface is small, which reduces the synchronization costs.

Our present implementation utilizes a partitioning algorithm based on Hilbert's space-filling curve [PB96]. The Figs. 4 and 6 display domain decompositions derived with this algorithm for the work estimation formula (6). Apparently, the extensions

of the domain assigned to each node vary remarkably, but the workloads according to (7) always differ by less than 5%.

5 Computational Results

We use a standard test for Euler equations of a single polytropic gas to evaluate the proposed parallelization strategy within the MPI-based AMROC implementation [DA03]. A homogeneous circular region of high pressure and density expands in an enclosed box. After a few time steps, the initial discontinuity separates into a rapidly expanding shock wave, a following slower contact discontinuity and a collapsing smooth rarefaction wave.

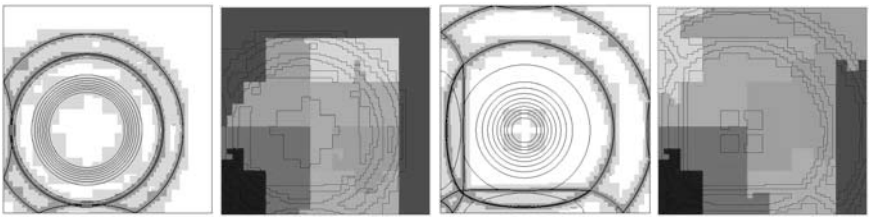
We utilize a base grid of 150×150 cells and apply a two-level refinement with the factors $r_1 = 2$ and $r_2 = 4$. About 200 root level grid time steps with $C_{CFL} \approx 0.8$ to $t_{end} = 0.5$ were computed, where the Clawpack implementation of the Wave Propagation Method [LeV97] with the approximate Riemann solver of Roe was employed as numerical update routine. A repartitioning of the hierarchy was done only at root level time steps, cf. Sec. 4.2. A standard Linux-Beowulf-cluster of Pentium-III-1 GHz CPUs connected with Fast Ethernet (effective bandwidth ≈ 40 MB) was used for the benchmarks. Exemplary results on eight nodes are shown in Fig. 4. While the AMROC computation on one node required 152 min, the execution time decreased to remarkable 13.9 min on 16 nodes. Table 1 shows a breakdown of the computational time for the most important AMR operations. For one node the fractions spent

Table 1. Computational time on P nodes.

Task [%]	P=1	P=2	P=4	P=8	P=16
Update by $\mathcal{H}^{(\cdot)}$	86.6	83.4	76.7	64.1	51.9
Flux correction	1.2	1.6	3.0	7.9	10.7
Boundary setting	3.5	5.7	10.1	15.6	18.3
Recomposition	5.5	6.1	7.4	9.9	14.0
Misc.	4.9	3.2	2.8	2.5	5.1
Time [min]	**151.9**	**79.2**	**43.4**	**23.3**	**13.9**
Efficiency [%]	**100.0**	**95.9**	**87.5**	**81.5**	**68.3**

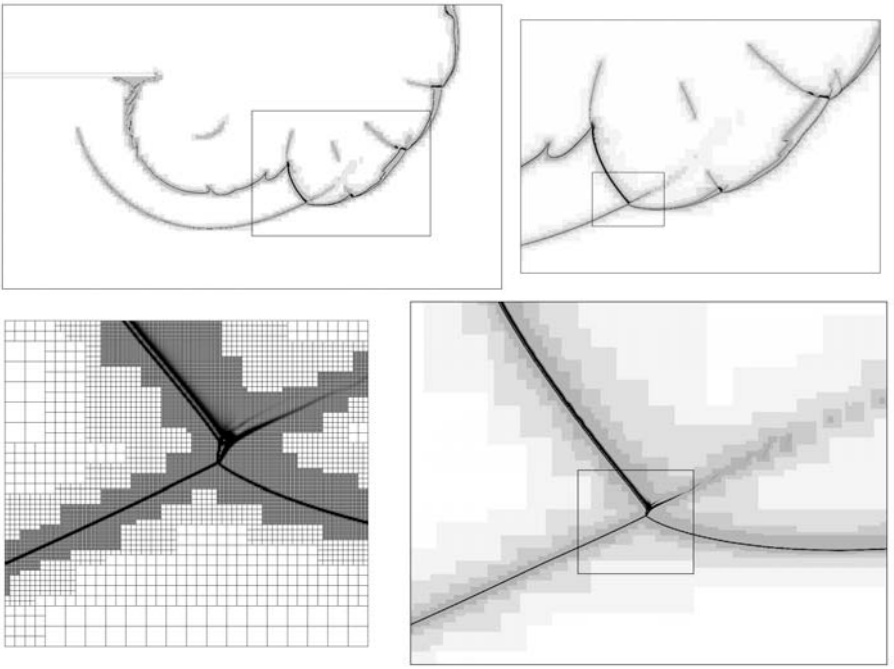
in different parts of the code are in good agreements with the results in [BC88] and at least for a moderate number of computing nodes we achieve an acceptable parallel efficiency.

In order to demonstrate that our parallelization approach is also well suited for cutting-edge AMR simulations, we briefly present exemplary results for a two-dimensional hydrogen-oxygen detonation propagating out of a tube into unconfinement. The simulation reproduces the critical width for square tubes and is in perfect qualitative agreement with experimental results. The computation was run effectively in less than 4 days real time on a Linux-Beowulf-cluster of 48 CPUs and



After 38 time steps After 79 time steps

Fig. 4. Circular Riemann problem in an enclosed box. Isolines of density on two refinement levels (indicated by gray scales) and distribution to eight nodes (indicated by different colors).



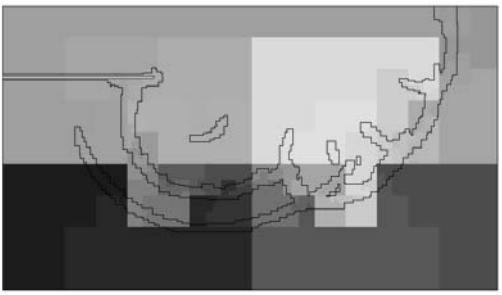


Fig. 5. (Upper four graphics.) Planar detonation diffraction. Density distribution on four refinement levels 240μs after the detonation has left the tube. Multiple zooms are necessary to display the finite volume cells.

Fig. 6. Planar detonation diffraction. Distribution of computational domain to 48 nodes.

spent ≈ 2000 h CPU time in the update operator $\mathcal{H}^{(\cdot)}$, which was a special approximative Riemann solver for multi-component Euler equations with general equation of state. The reaction terms according to a detailed non-equilibrium reaction mechanism were incorporated numerically into $\mathcal{H}^{(\cdot)}$ with a fractional step method and required the additional solution of a *stiff* initial value problem in each FV cell. See [Dei03] for details.

As detonation simulations require an extraordinarily high local resolution to capture the influence of the chemical kinetics correctly, the computation benefits remarkably from dynamic mesh adaptation. The graphics in Fig. 5 display the solution on the refinement levels $240\,\mu$s after the detonation has left the tube (730 root level time steps with $C_{CFL} \approx 0.8$, one half of the domain was simulated) and the enormous efficiency of the refinement is apparent. The base grid used 508×288 cells and four levels of refinement with $r_{1,2,3} = 2$, $r_4 = 4$, which corresponds to ≈ 150 M cells, but at the time step displayed the simulation uses less than 3.0 M cells on all levels.

6 Conclusions

We have described a locality-preserving parallelization strategy for the blockstructured AMR algorithm after Berger and Colella, which is tailored especially for distributed memory machines. The approach is based on domain decomposition and reduces the communication costs. In particular, the important flux correction procedure, which can become quite complicated in a distributed memory environment, can be implemented with ease. Benchmark calculations with our MPI-based implementation AMROC show promising parallel speed-up and we were able to obtain exceptional detonation simulations with the framework on standard Linux-Beowulf-clusters, cf. [Dei03].

References

BBS94. J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-dimensonal adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15(1):127–138, 1994.

BC88. M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1988.

BL98. M. Berger and R. LeVeque. Adaptive mesh refinement using wave-prop. algorithms for hyperbolic systems. *SIAM J. Num. Anal.*, 35(6):2298–2316, 1998.

CW93. W. Crutchfield and M. L. Welcome. Object-oriented implementation of adaptive mesh refinement algorithms. *J. Scientific Prog.*, 2:145–156, 1993.

Dei03. R. Deiterding. *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Techn. Univ. Cottbus, Sep 2003.

DA03. R. Deiterding. AMROC - Blockstructured Adaptive Mesh Refinement in Object-oriented C++. Available at http://amroc.sourceforge.net, Oct 2003.

KB95. S. R. Kohn and S. B. Baden. A parallel software infrastructure for structured adaptive mesh methods. In *Proc. of the Conf. on Supercomputing '95*, Dec 1995.

LeV97. R. J. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *J. Comput. Phys.*, 131(2):327–353, 1997.

PB96. M. Parashar and J. C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proc. 29th Hawaii Int. Conf. on System Sciences*, Jan 1996.

PB97. M. Parashar and J. C. Browne. System engineering for high performance computing software: The HDDA/DAGH infrastructure for implementation of parallel structured AMR. In *Structured Adaptive Mesh Refinement Grid Methods*, IMA Volumes in Mathematics and its Applications. Springer, 1997.

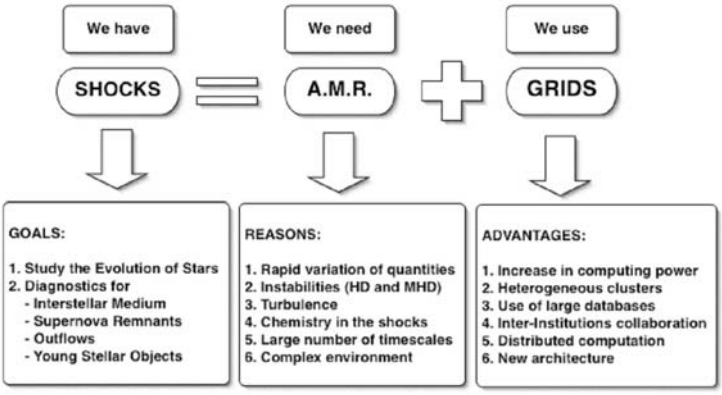
RBL00. C. A. Rendleman, V. E. Beckner, M. Lijewski, W. Crutchfield, and J. B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3, 2000.

Adaptive Mesh Refinement in a Grid Computing Environment

G. C. Murphy, T. Lery, and L. O'C. Drury

Dublin Institute for Advanced Studies, 5 Merrion Square, Dublin 2 , Ireland.
{gmurphy,lery,ld}@cp.dias.ie

The major part of a star's lifetime, from birth to death, is punctuated by strong shocks. Such discontinuities can be efficiently treated using Adaptive Mesh Refinement. Moreover, by using theoretical properties of shocks, it is possible to begin the mesh refining before a gradient appears. We present here our new project of implementation of shock prediction within AMR techniques with the aim of using Grid technology as part of the CosmoGrid. This Irish project consists of a network of the major research computing facilities in Ireland. Such a Grid will allow us to use AMR techniques applied to astrophysical shocks in a new type of computing architecture.



1 Astrophysical Shocks

1.1 Physical Modelling

Astrophysical systems, like supernova remnants, jets and outflows, are mainly observed thanks to their strong shocks. A typical example is a Type Ia supernova, which

occurs when a white dwarf undergoes thermonuclear detonation at the end of its life. This results in a spherical piston-type shock being driven out from the centre. As a first approximation, the detonation can be modelled using the Euler Equations for an ideal compressible gas.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{U} = 0 \tag{1}$$

$$\frac{\partial (\rho \mathbf{U})}{\partial t} + \nabla \cdot (\rho \mathbf{U} \otimes \mathbf{U} + pI) = 0 \tag{2}$$

$$\frac{\partial (\frac{1}{2}\rho \mathbf{U}^2 + \varepsilon)}{\partial t} + \nabla \cdot (\mathbf{U}(\frac{1}{2}\rho \mathbf{U}^2 + p + \varepsilon)) = 0 \tag{3}$$

The equations are expressed in the primitive variables ρ (density),**U** (velocity) and p (pressure). An example solution to the Euler Equations for a shock tube or Riemann problem is illustrated in Figure 1. A more complex physical problem is sketched in figure 2. This shows a box model of diffusive shock acceleration, a mechanism proposed responsible for producing nuclear cosmic rays. The situation becomes even more complex during star formation where magnetic field and chemistry play an important role in the dynamics. [Drur99].

1.2 Simulation Results

A sample simulation of magneto-hydrodynamic jet propagation is shown in Figure 3. The simulation includes MHD, cooling and chemistry. In A, the jet enters a uniform environment, while it goes through a density step in B and C, from high to low density in B, and the opposite direction in C. [Lery02]

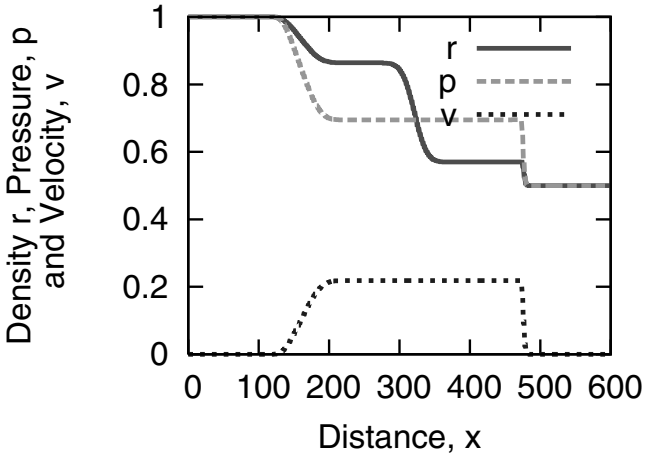


Fig. 1. Example of Solution to Riemann Problem: Density, Pressure and Velocity curves for 1-D Simulation

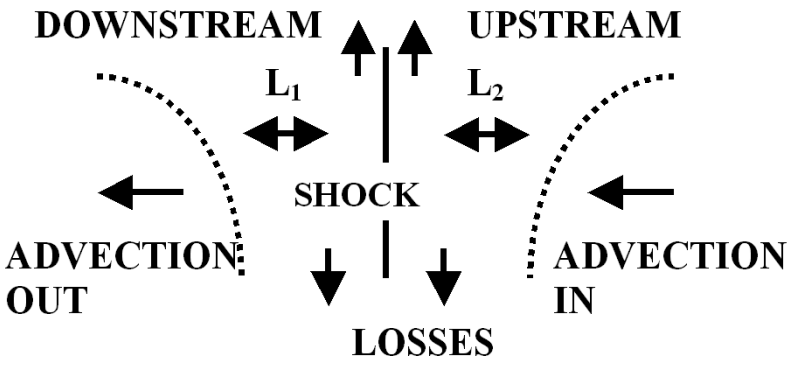


Fig. 2. Sketch for box model of diffusive shock acceleration. Source: L. Drury et al. 1999

1.3 Adaptive Mesh Refinement Techniques

Steep gradients such as those shown in Figure 1 may be treated using the techniques of Adaptive Mesh Refinement. Using known properties of shocks, it may be possible to predict the location of the shock independently, and thus begin the refining before a gradient becomes apparent. This may allow us to take advantage of the heterogeneous nature of the Grid.

2 Grid Refinement

2.1 Sharing Resources

The Grid paradigm is about enabling communities (virtual organizations) to share resources as they pursue common goals. It allows to aggregate computing power and to make use of distributed resources in order to get a larger virtual computer. Single PCs can be added to existing large Beowulf clusters together with any other fast machine, like Crays. The Grid software allows us to choose automatically the best set of machines to run a computation at various locations, physically separated.

2.2 Domain Mapping

In order to properly distribute the workload between different processors, the underlying data structure can be broken into pieces and each piece assigned to a different processor designated by the Grid software. The borders of the partitioned data structure must be exchanged between adjacent nodes. It is therefore desirable to divide the domain in such a way as to minimize the surface area of the partitions [Berg96].

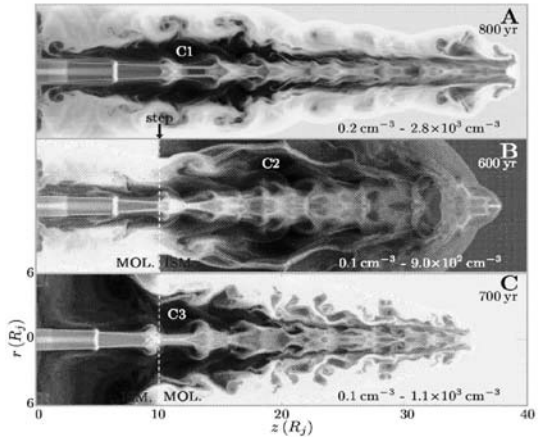


Fig. 3. Simulations of a pulsating jet coming from a star in its infancy. Case A: the jet enters a uniform interstellar medium. Case B: the jet exits the molecular cloud. Case C: the jet enters a molecular cloud. Note the numerous magnetic and hydrodynamic instabilities as well as the cavities (C1, C2, and C3) Source: T. Lery

3 CosmoGrid

CosmoGrid is the first Irish countrywide Grid project. It includes all the major universities and research institutes in Ireland, for a total of 9 institutions. It will consist of 5 gateways linking the Beowulf clusters and the existing fast computers. The object of the program is to increase sharing of high performance computing resources and encourage cooperation in astrophysics, geophysics and atmospheric physics. It allows the participants to harness and make available their combined computing power to perform individual computations in an efficient manner.

4 Conclusions & Current Status

As Adaptive Mesh Refinement can be computationally intense, it is an ideal candidate for the Grid environment. We intend to implement a version of Adaptive Mesh Refinement which takes advantage of the flexibility offered by grid technology in order to study astrophysical shocks.

- CosmoGrid gateways are up and running and new clusters are added to the infrastructure.
- We are currently selecting an AMR code for the implementation of our shock models with the aim to use the Grid technology to effectively study the stucture of shocks and their properties.

5 Acknowledgements

This work was carried out as part of the CosmoGrid project, funded under the Programme for Research in Third Level Institutions (PRTLI) administered by the Irish Higher Education Authority under the National Development Plan and with partial support from the European Regional Development Fund.

References

Berg96. Berger M.J., Aftosmis, M.J., and Melton, J.E.,: Accuracy, adaptive methods and complex geometry. Proc. 1st AFOSR Conf. on Dynam. Mot. in CFD. Rutgers, NJ 1996.

Zeld66. Zel'dovich, Ya. B., Raizer, Yu. P.,: Physics of Shock Waves and High Temperature Hydrodynamic Phenomena. Academic Press, 1966, 374

Drur99. Drury L.O'C., Duffy P., Eichler D., Mastichiadis A.: On box models of shock acceleration and electron synchrotron spectra. Astron. Astrophys. 1999, 347, 370

Lery02. Lery, T., Henriksen R.N., Fiege, J.D., Ray, T.P., Frank, A., Bacciotti, F.: A Global jet/circulation model for young stars 2002, Astron. Astrophys., in press.

Performance of Vector/Parallel Orientated Hydrodynamic Code

Shigeki Miyaji, Ayato Noro, Tomoya Ogawa, Mitue Den, Kazuyuki Yamashita, Hiroyoshi Amo, and Kazushi Furuta

Graduate School of Science and Technology, Chiba University, Japan
Simulator Group, Communications Research Laboratory, Japan
The Center for Educational Research, University of Yamanashi, Japan
NEC, Japan

Summary. We have developed a hydrodynamic 3 dimensional AMR code which is suitable for vectorization and parallelization based on Fully Threaded Tree method. For the case of 3D FTT-AMR simulation, the number of higher resolution cells is eight times of that of mother low resolution cells. Therefore if the region where the highest resolution is required has certain volume, it is better to use single time step for every levels of resolution because we can avoid artificial flow at the surface of resolution interface. This level-independent time step also makes possible to simplify the evaluation of refinement indicator. As a result, we can overcome the demerit of consuming CPU time by computing at level-independent time step less than CFL limit, by boosting vector acceleration ratio.

1 Introduction

Many astrophysical phenomena originate at small region and propagate to other large region. Because astrophysical parameters vary wide range, we need Adaptive Mesh Refinement (AMR) scheme to simulate astrophysical phenomena. For example, star forms from interstellar gas by collapsing $< 10^{-6}$. Massive star dies by a supernova explosion which initiates at the central core and becomes visible when the exploding shock wave reached to the surface by expanding $> 10^{6}$. Certainly such wide range of space resolution needs AMR scheme. Moreover, because timescales of physical processes also varies many orders of magnitude from the central dense region to the outer vacant region. Therefore we need very stable and highly vectorized and/or parallelized AMR code in order to simulate astrophysical phenomena. Here, we present our implementation of vector-parallel orientated AMR code for astrophysical hydrodynamic simulation.

Among some AMR methods, we employed Fully Threaded Tree (FTT) method (Khokhlov [1]) in our simulation code. FTT method gives an easy operation of refining and coarsening cells by using list-vectors. Refining and coarsening of the oct are automatically done according to the value of refinement indicator.

Because we focussed on 3D simulation with vector and parallel options, our FTT code (see e.g., Ogawa [2]) is different from Khokhlov's original FTT method at following points: Using the level-independent time step and distributing the value of the refinement indicator of each oct over surrounding octs by a simple point-spread-function. Such simplification is preferable for keeping longer vector length so that we could obtain higher vector acceleration ratio and therefore shorter CPU time by utilizing level independent single time step.

In section 2, we describe our implementation in detail. Section 3 is devoted for the measurements of vectorization and parallelization efficiency of our code. Summary and discussion is given in section 4.

2 Our FTT-AMR scheme

Our implementation [2] of vector-parallel orientated AMR code utilizes single level independent time step and non-local refinement indicator. We adopted Roe-MUSCL scheme [4] for the part of flux evaluation in which the 3rd order accuracy in space is achieved. Though our code has 2nd order accuracy in time by employing operator splitting method [4], we use 1st order in this paper.

For FTT method of AMR code, Khokhlov [1] designed the time step $\Delta t(l)$ of level l cells, which have 2^{-l} side length, to be

$$\Delta t(l) = \Delta t(l_{\max}) 2^{l_{\max}-l}, \tag{1}$$

where subscript max means the maximum level (the highest resolution). When the hydrodynamical variables are advanced by this time step at level l, the CFL condition should be satisfied minimally at every level l.

However, when there exists a hydrodynamical flow at the boundary between cells of different levels, inevitable noise takes place at the boundary and may cause numerical difficulty. This kind of numerical difficulty is a common problem for the case that different resolution meshes are placed adjacently as like nested grids scheme. When a flow from larger resolution mesh comes to finer resolution mesh, if the time step of finer resolution mesh is shorter than that of larger resolution mesh, overshooting of the flow would be enhanced, because quantities of the finer resolution mesh are time advanced twice for 2^{-l} cell structure regardless the back effects of the flow. Usually, one would introduce a damping factor on the flow in order to avoid the growth of such overshooting or would add ghost meshes to calculate the flow.

In our case, we adapt single level-independent time step determined by the CFL condition for the highest resolution level $l_{\max}$ cells as

$$\Delta t(l) = \Delta t(l_{\max}). \tag{2}$$

This means that time steps used for lower level calculations are shorter than their CFL allowance limit which means that we are wasting CPU time. However for the case of 3D simulation, each oct has at least 2^3 cells so that the number of higher resolution octs are at least eight times larger than that of lower resolution parent

octs of the same volume. When we calculate physical variables of different level octs with different time steps as like Khokhlov's case, we should calculate them with shorter vector length (the total number of each level octs instead of the total number of octs). Therefore, when we calculate physical variables of each resolution octs separately with vector option, we are wasting CPU time even though we have reduced the number of operations themselves.

Therefore, it is worth to calculate physical variavles of all octs at every time step with longer vector length and reduce the number of executions of do-loops. Then we could obtain better vector acceleration ratio and resultant reducing of CPU time and could avoid unphysical flow between different levels interface.

The refinement indicator ξ at every oct of our implementation is calculated as follows: At first, calculate local value by standard definition of refinement indicator. Then, sum up indicators of surrounding cell with a simple point-spread-function and revise the refinement indicator as a non-local one.

In this simulation, the local refinement indicator is calculated under the guideline of AMR2003 benchmark session for hyperbolic benchmark (single physics; Sedov point- like explosion): Refinement criterion is that density and pressure jump should be smaller than 0.1.

We introduce following gradient function $G(q)$,

$$G(q) = \max\left(\frac{\max(|q_+|, |q_-|)}{\min(|q_+|, |q_-|)}\right) - 1, \tag{3}$$

where q is physical variables and the suffixes "+" and "−" denote a pair of adjoining cells and the "+" cell is the one having larger coordinate value. For the case of hydrodynamic calculation of Sedov point-like explosion which is given as a bench mark problem in the benchmark session of AMR2003, only mass density ρ and pressure P for q in $G(q)$ should be the refinement criterion, we can write down the formula to determine ξ as

$$\begin{cases} \xi = 1 \\ \quad \text{when} \\ \max\left(G(\rho), G(P)\right) > 0.1. \end{cases} \tag{4}$$

As like Sedov point-like explosion, we often face the strong shock wave in the astrophysical phenomena. Because stellar envelope has steep density and pressure gradients small overpressure grows into a strong shock when it propagates into outer envelope (this is so called "pressure growing"). When this kind of strong shock approached, standard AMR refinement recipe often could not catch up this sudden change. Therefore we have to refine meshes well before the arrival of the shock wave. Such preparation is also better because the shock wave may enhance numerical noise generated at the time of refinement, which causes numerical instability.

We require not only that the indicator should be smooth in the simulation volume in order to avoid refining cells having occasional noise but also that the indicator should have information of nearby cells in order to prepare refinement well before

the arrival of the shock wave. For this purpose, we revise the definition of ξ as not to a local indicator but to an indicator of larger volume, i.e., we revise the value of the target oct by summing ξ of the surrounding octs.

We define $\xi_{\rm rev}$ as

$$\xi_{\rm rev} = \sum_{\text{nearby surrounding cells}} w\xi, \tag{5}$$

where w is a weight function. In this simulation we set $w = 1$. This simple summation is better than the sophisticated Khokhlov's method which estimates diffusion front. The reason is that we need not estimate diffusion thickness because we used single level-independent timestep and CFL condition at the highest cells ensures that the shock and contact discontinuity fronts shift at most only one cell by one time step. Then, we can code the program for vectorization and/or parallelization much easier.

By using upper and lower thresholds, namely $\xi_{\rm refine}$ and $\xi_{\rm coarsen}$, when the value of $\xi_{\rm rev}$ falls out of threshold limits, the operation of refining or coarsening is carried out. We set $\xi_{\rm refine} = 2.9$ and $\xi_{\rm coarsen} = 2.1$ through out in this simulation.

3 Performance evaluation of our implementation

Performance evaluation of our code is done according to the guideline of AMR2003 benchmark session. Because our code is vector-parallel orientated, we do not use provided SGI machine as like other benchmark session presenters but we use NEC SX-6, a vector-parallel machine, as a counter part.

According to the AMR2003 guideline, we set parameters as the box size $[0,1]$, the density at the center $\rho_c = 8$, the pressure at the center $P_c = 2 \times 10^{-5}$, the explosion energy $E_{tot} = 0.25$, and the specific heat $\gamma = 1.4$. Initial explosion takes place at the central 4 cells at the center of our simulation box $(0.5, 0.5, 0.5)$. Because we set simulation box size as $[0,1]$ instead of $[-1,1]$ in the guideline, other variables are normalized according to the box size. We simulated 4 different effective resolution cases of 128^3, 256^3, 512^3, and 1024^3, namely cases L5-7, L5-8, L5-9, and L5-10, respectively.

We used one node (8PEs) SX-6 at Communications Research Laboratory for this performance evaluation of our implementation. We measured each case with 4 different models according to compiler options; vector-parallel (VP), scalar-parallel (SP), vector-nonparallel (VN), and scalar-nonparallel (SN).

First, we show how much CPU-times of each part of our code are spent for each model. In tables 1, 2, 3, and 4 we showed the result of first 1000 steps for the cases of levels L5-7, L5-8, and L5-9 models and first 4000 steps for the case of L5-10 models. The part of Flux calculates flux of physical parameters and the part of AMR changes mesh sizes which implements refining and coarsening the mesh at each time step. The part of Other contains input and output. Total shows the total CPU-time we spent.

Table 1. CPU-times spent at each part for the case of 128^3 resolution (L5-7).

Part	VP	SP	VN	SN
Flux	180.4 s	537.8 s	295.9 s	3214.7 s
AMR	4.3 s	14.1 s	4.9 s	48.8 s
Other	3.3 s	6.3 s	2.0 s	19.1 s
Total	188.0 s	558.2 s	302.8 s	3282.6 s

Table 2. CPU-times spent at each part for the case of 256^3 resolution (L5-8).

Part	VP	SP	VN	SN
Flux	158.4 s	553.0 s	328.5 s	3141.0 s
AMR	5.7 s	17.3 s	6.3 s	42.1 s
Other	2.7 s	6.1 s	2.2 s	18.6 s
Total	166.8 s	576.4 s	337.0 s	3201.7 s

Table 3. CPU-times spent at each part for the case of 512^3 resolution (L5-9).

Part	VP	SP	VN	SN
Flux	193.0 s	604.2 s	308.3 s	3257.2 s
AMR	7.3 s	20.8 s	7.2 s	57.6 s
Other	2.8 s	6.3 s	3.4 s	19.2 s
Total	203.1 s	631.3 s	318.9 s	3334.0 s

Table 4. CPU-times spent at each part for the case of 1024^3 resolution (L5-10).

Part	VP	SP	VN	SN
Flux	1107 s	5421 s	2933 s	32538 s
AMR	47 s	235 s	15 s	461 s
Other	11 s	49 s	11 s	196 s
Total	1165 s	5705 s	3004 s	33195 s

The final shapes of Sedov point-like explosion at $t = 0.05$ are shown in figs. 1, 2, 3, and 4. Density, pressure, and momentum are plotted against the distance from the center for all cells. Theoretical solution is shown by thin solid curves.

Tables 5, 6, 7, and 8 shows the number of octs we used at 1000 step and at the final step when $t = 0.05$. As one can find easily, the most plentiful part of the total octs is the highest level even for the case of the lowest resolution case at the last half time steps (see table 5). For the case of higher resolution cases, the portion where the number of the highest resolution dominates the total number of octs start at much earlier time step (see table 8). Such oct structure ensures our argument of ignoring CPU-time consumption by useing the single time step for every level.

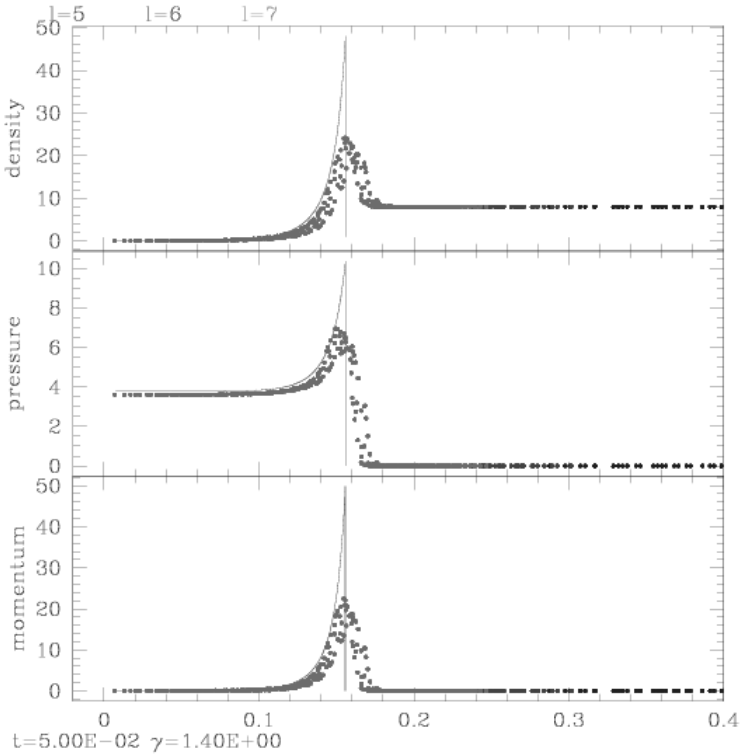


Fig. 1. Sedov point-like explosion at t=0.05 with 128^3 resolution. All cells are plotted according to the distance from the center.

Table 5. Oct structure at 1000 step ($t = 0.0236$) and at the final (1747) step ($t = 0.05$) for the case of 128^3 resolution (L5-7)

Oct level	1000 step	final step
L5	4096	4096
L6	1664	2880
L7	6912	12544
Total	12672	19520

The total CPU time $\tau_{\rm CPU}$ for these cases with VP option, the total vetorization rate VEC, vector acceleration ratios with parallel option $V_P = \tau_{\rm CPU}({\rm SP})/\tau_{\rm CPU}({\rm VP})$ and without parallel option $V_N = \tau_{\rm CPU}({\rm SN})/\tau_{\rm CPU}({\rm SP})$ at the 1000 step (4000 step for L5-10 case), and parallel acceleration ratios with vector option $P_V = \tau_{\rm CPU}({\rm VN})/\tau_{\rm CPU}({\rm VP})$ and without vector option $P_N = \tau_{\rm CPU}({\rm SN})/\tau_{\rm CPU}({\rm SP})$ at the 1000 step (4000 step for L5-10 case) are given in Table 9.

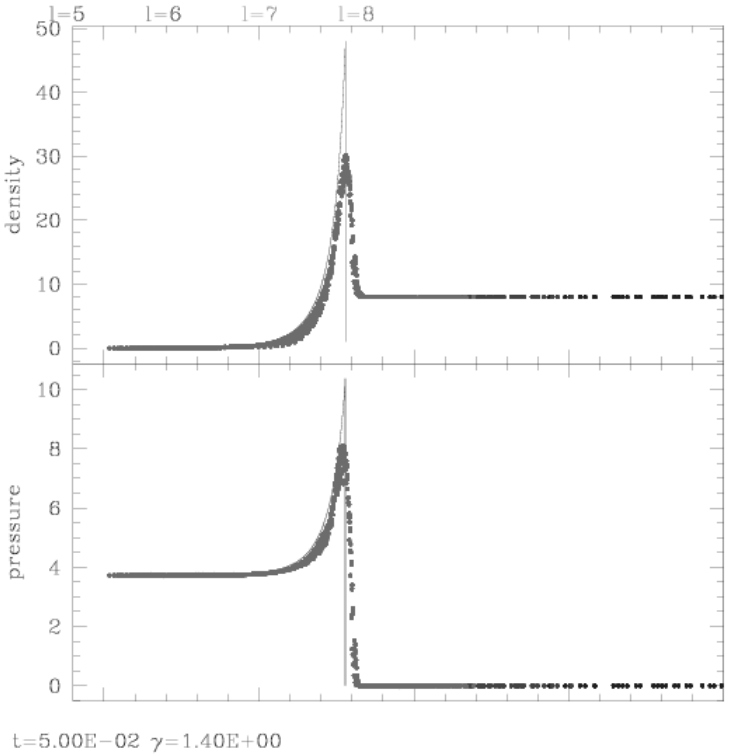


Fig. 2. Sedov point-like explosion at t=0.05 with 256^3 resolution. All cells are plotted according to the distance from the center.

Table 6. Oct structure at 1000 step $(t = 0.0042)$ and at the final (6666) step $(t = 0.05)$ for the case of 256^3 resolution (L5-8)

Oct level	1000 step	final step
L5	4096	4096
L6	704	2880
L7	1664	12160
L8	6912	62464
Total	13376	81600

Our code shows high vectorization rate (more than 99 %) and good parallel ratio P_N ($5.281 - 5.881$ for 8 PEs). Though vectorization and parallelization compensate each other because vector length is shorter when we use parallelization, we still have allowable efficiency in L5-10 case.

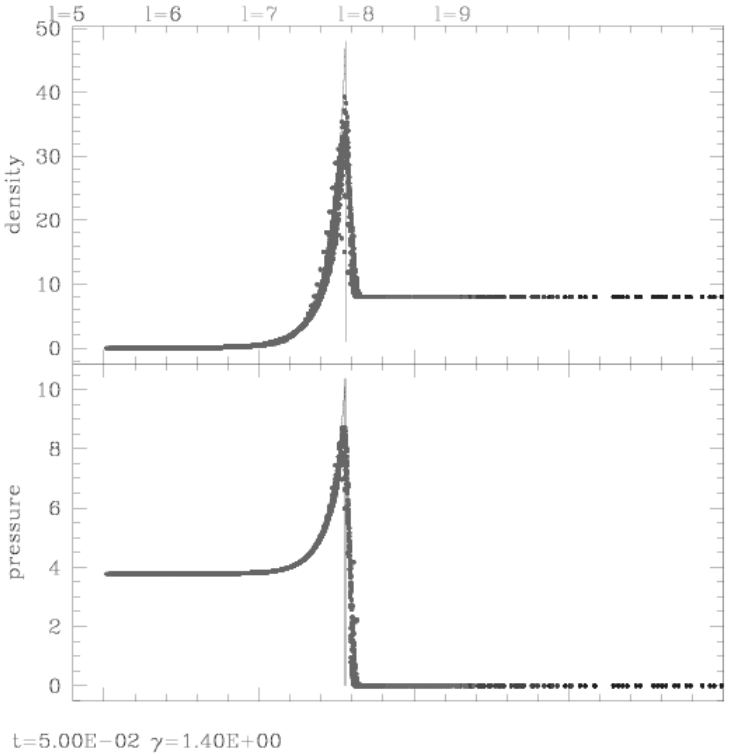


Fig. 3. Sedov point-like explosion at t=0.05 with 512^3 resolution. All cells are plotted according to the distance from the center.

Table 7. Oct structure at 1000 step $(t = 0.00074)$ and at the final (18930) step $(t = 0.05)$ for the case of 512^3 resolution (L5-9)

Oct level	1000 step	final step
L5	4096	4096
L6	256	2880
L7	704	10816
L8	1664	59392
L9	6912	373312
Total	13632	450496

4 Summary and Discussion

Our code employed simpler method than Khokhlov's [1] sophisticated method, but we could show that our approach works well for this simulation. Especially, although

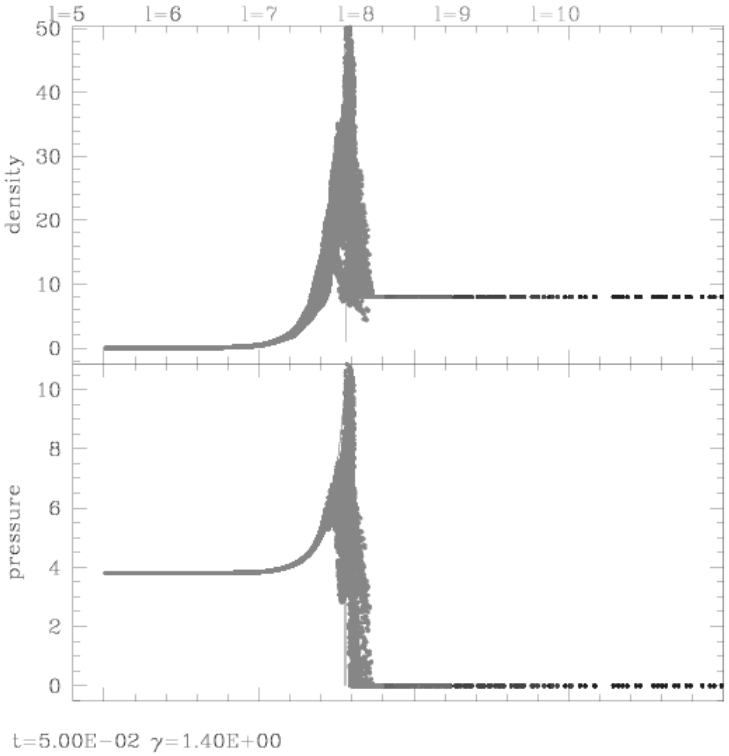


Fig. 4. Sedov point-like explosion at t=0.05 with 1024^3 resolution. All cells are plotted according to the distance from the center.

we applied plain weight function ($w = 1$), we have not experienced any numerical difficulty. When the shock approaches from diogonal direction, the weight of vertices

Table 8. Oct structure at 1000 step $(t = 0.00080)$ and at the final (41938) step $(t = 0.05)$ for the case of 1024^3 resolution (L5-10)

Oct level	1000 step	final step
L5	4096	4096
L6	256	2880
L7	704	10816
L8	2096	59328
L9	7488	373168
L10	37056	2629128
Total	51696	3079416

Table 9. Total CPU time and vectorization rates are listed. V_P and V_N are listed in the upper part and P_V and P_N are listed in the lower part of columns.

Cases	CPU time	*VEC*	V_P / P_V	V_N / P_N
L5-7	261 s	99.33 %	2.969	10.84
			1.611	5.881
L5-8	2640 s	99.50 %	3.456	9.50
			2.020	5.555
L5-9	15827 s	99.69 %	3.108	10.45
			1.570	5.281
L5-10	166491 s	99.80 %	4.897	11.05
			2.579	5.819

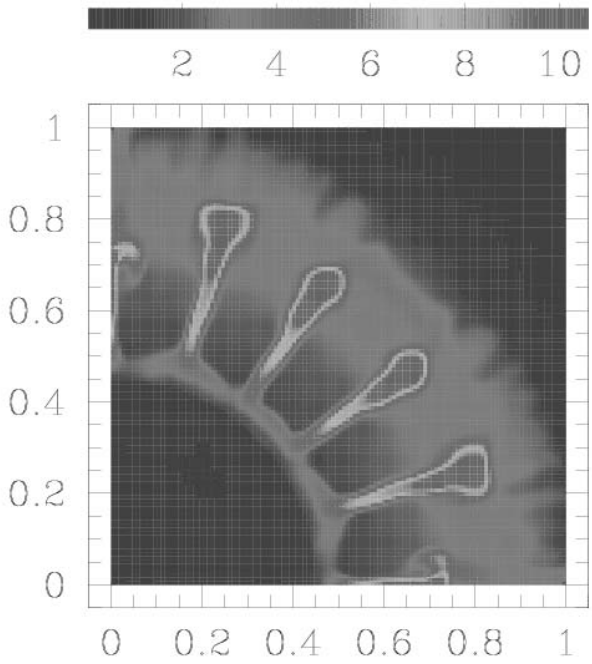


Fig. 5. Density contour of the RT instability in SN1987A after 4600 seconds from the explosion at the center on the equator [3].

should be weighted higher because only one vertex represents the shock. However, in this simulation, though the profile of the front is spherical, plain weight function well predicts its approach and causes no trouble even we use the plain weight function in calculating non-local refinement indicator.

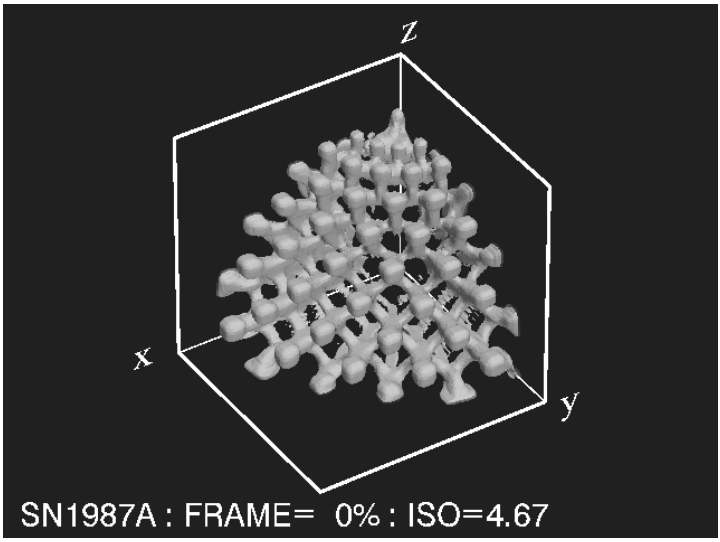


Fig. 6. 3D view of the RT instability in SN1987A after 4600 seconds [3].

Vector and parallel acceleration ratios are fair when both options are set, but actual simulation always needs the highest machine efficiency, i.e., the total number of octs would be large so that we could expect longer vectorization length even when we set parallel option. One of such example is a parameter search for 3D Rayleigh-Taylor (and/or Richchtmeyer-Meshkov) instability in the supernova envelope. In this case, because the RT instability initiates at the very inner region and grow into outer large envelope and its non-linear growth makes the structure of density/composition discontinuity very complicated, we need high resolution AMR code to simulate the growth of RT instability. We have shown that our code is efficient for such simulation [3] (figs. 5 and 6). Noro et al. [3] also evaluated the performance of our code with such RT simulation with pseudo-vector parallel machine of Hitachi SR8000. Though their result showed good parallel efficiency but vector efficiency alone is poor (though the total acceleration ratio is 5.736 for 8 PEs but $V_N = 1.024$) by that pseudo-vector machine because it is actually a parallel machine.

We could conclude that our vector-parallel orientated implementation of the FTT-AMR scheme is efficient in simulations of astrophysical phenomena. Detailed example of our astrophysical simulation would be published elsewhere.

The earth simulator [5] is a massively parallel vector-parallel machine and other similar machines would be available soon. So, our implemetation of vector-parallel orientated FTT-AMR scheme would have popularity when one want to use these machines.

Acknowledgments This computation is carried out by NEC SX-6 at Communications Research Laboratory. This work is supported in part by a Grant in Aid for Scientific Research No. 15037202 from MECSST, Japan.

References

1. KHOKHLOV, A. M., 1998, J. Comp. Phys., **143**, 519.
2. OGAWA, T., 2003, PhD Thesis., Chiba University.
3. NORO, A., OHTA, T, OGAWA, T, YAMASHITA, K., MIYAJI, S., AND DEN, M., 2002, ISHPC2002, Lec. Note Comp. Sci., **2327** 207.
4. HIRSCH, C., 1992, "Numerical Computation of Internal and External Flows (Vol.2: Computational Methods for Inviscid and Viscous Flows)", *A Wiley-Interscience Publication*.
5. URL:www.es.jamstec.go.jp/esc/eng/

On the efficiency of AMR in NIRVANA3

U. Ziegler

Astrophysikalisches Institut Potsdam, D-14482 Potsdam, Germany uziegler@aip.de

Summary. The efficiency of the grid-adaptive magnetohydrodynamics code NIRVANA3 is studied. For that two three-dimensional benchmark problems are proposed: a hydrodynamical implosion problem possessing spherical symmetry and a shock-cloud collision problem in a magnetic medium. An efficiency parameter is defined which contains both the obtained speedup factor and an error estimate.

1 Introduction

I investigate the efficiency of a solution-adaptive scheme for the equations of ideal magnetohydrodynamics (MHD) in three space dimensions given in conservation form

$$\partial_t \rho + \nabla\cdot(\rho\mathbf{v}) = 0, \tag{1}$$

$$\partial_t(\rho\mathbf{v}) + \nabla\cdot\left[\rho\mathbf{v}\mathbf{v} + \left(p + \frac{1}{2\mu}|\mathbf{B}|^2\right) I - \frac{1}{\mu}\mathbf{B}\mathbf{B}\right] = 0, \tag{2}$$

$$\partial_t e + \nabla\cdot\left[\left(e + p + \frac{1}{2\mu}|\mathbf{B}|^2\right)\mathbf{v} - \frac{1}{\mu}(\mathbf{v}\cdot\mathbf{B})\mathbf{B}\right] = 0, \tag{3}$$

$$\partial_t \mathbf{B} + \nabla\times\mathbf{E} = 0. \tag{4}$$

ρ is the mass density, e is the (total) energy density, $\mathbf{v}$ is the velocity, $\mathbf{E} = -\mathbf{v}\times\mathbf{B}$ is the electric field, μ is the magnetic permeability and γ is the ratio of specific heats. The equations are supplemented by the divergence-free constraint for the magnetic field,

$$\nabla\cdot\mathbf{B} = 0,$$

and the thermal pressure is computed from the ideal gas equation of state

$$p = (\gamma-1)\left(e - \frac{1}{2}\rho|\mathbf{v}|^2 - \frac{1}{2\mu}|\mathbf{B}|^2\right). \tag{5}$$

The equations are solved numerically with a hybrid approach: Faraday's law Eq. (4) is treated with constraint transport (CT) techniques ie. a special discretization

utilizing a staggered grid (see eg. Evans & Hawley [6]). Hydrodynamics is updated with the second-order Godunov-type central scheme of Kurganov et al. [8]. The combined method is conservative and preserves the zero divergence condition for **B** in discretized form up to machine precision. The resulting MHD scheme does neither rely on Riemann solvers nor on any characteristic decomposition typical for upwind schemes. It is easy to implement, computationally fast and, yet, compelling in accuracy to second-order approximate Riemann solver schemes (see Ziegler [12]). Note that no additional effort is needed to enforce solenoidality of **B** as opposed to schemes based on a non-staggered approach. In particular, neither extra source terms proportional to $\nabla\cdot\mathbf{B}$ have to be introduced in order to propagate $\nabla\cdot\mathbf{B}$-errors away as in Powell's [9] approach nor is it necessary to solve a Poisson equation as in the projection method proposed by Brackhill & Barnes [1].

The MHD scheme is embedded in a grid refinement tool which is based on the principles of block-structured adaptive mesh refinement (AMR) elaborated by Berger & Collela [3] and used by many others (see eg. Walder & Folini [11], Berger & LeVeque [4], Bell et al. [5], Friedel et al. [7], Balsara & Spicer [2], Steiner et al. [10]). Although found on the same ideas, the AMR implementation here differs in many respects from this original approach. I pay attention to these differences in Sect. 2 which thoroughly deals with the AMR design. Worth mentioning at most, block-structured AMR has been adapted to the central-constraint transport ansatz with complications arising from the 2-step Runge-Kutta time integrator and the use of CT ie. the staggered grid. Moreover, the presented AMR approach attempts to combine flexibility in grid adaptation by using very small blocks of fixed size for refinement with speed by applying an intelligent patch clustering algorithm to speedup integration. All such developments has dropped into the software package NIRVANA3 – a versatile MHD code for grid-adaptive simulations in astrophysics and related fields of research.

2 AMR design

2.1 Refinement procedure

Controlled by specific criteria refinements on a fixed base grid are realized by hierarchically nested blocks (or patches) of size 4^3 cells (in 3D) with increasingly finer mesh spacing. Blocks of the same resolution build a refinement level l with $l = 1$ for the first refinement level and $l = L$ for the maximal allowed refinement level specified by the user. Adjacent levels have a refinement ratio of 2. Technically, the set of blocks build an oct-tree data structure. A block stands as a logical unit which not only holds the MHD variables and positions but also contains information about its environment in terms of pointers to its parent block or base grid, respectively, its grid neighbors and its nested child blocks. An example patch distribution is shown in Fig. 1.

The criteria used to control mesh refinement is twofold:

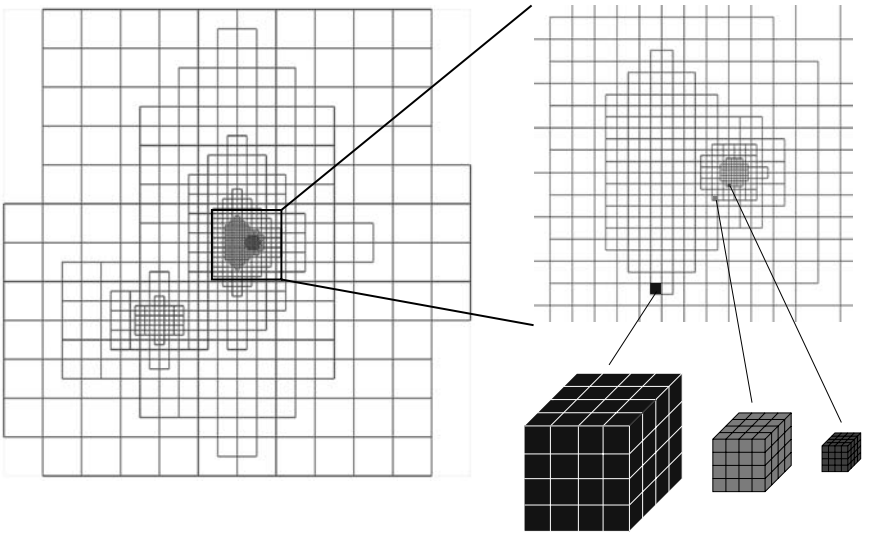


Fig. 1. An example patch distribution with 7 levels of refinement (the base grid is not shown). Each rectangle (cube in 3D) represents a patch of dimension $4 \times 4(\times 4)$ cells

i) on the base grid $(l = 0)$ normalized gradients in all variables are evaluated. If such gradient of *any* variable in a local neighborhood exceeds a prescribed threshold a block is generated and added to the data structure. Variables of the block are initialized by using conservative reconstruction procedures. If, on the other hand, gradients for *all* variables fall short of the thresholds an existing block is destroyed and removed from the data structure.
ii) refinement on blocks are triggered by a simple error indicator: during the update procedure of the grid hierarchy it becomes necessary to synchronize adjacent levels l and $l - 1$ in a restriction step in order to retain local conservation properties (see 2.3). Before such restriction is performed, the deviation from conservation between these two levels is used as indicator for grid refinement. This is reminiscent of a Richardson-type error estimation.

2.2 Patch clustering and time integration

Although mesh refinement is done in terms of individual blocks, the MHD solver does not directly operate on such blocks. Instead, blocks are temporarily mapped onto larger rectangular grid units called superblocks (see Fig. 2). The equations are then solved on such superblocks. The purpose of patch clustering is to increase efficiency: small blocks serve for a flexible adaptation ie. regions which actually need no refinement but yet are refined are sparse whereas patch clustering significantly reduces the otherwise overwhelming overhead due to the large number of interconnected blocks. For a given refinement level, blocks making up a coherent region are clustered in a way such that the corresponding superblock has maximum number

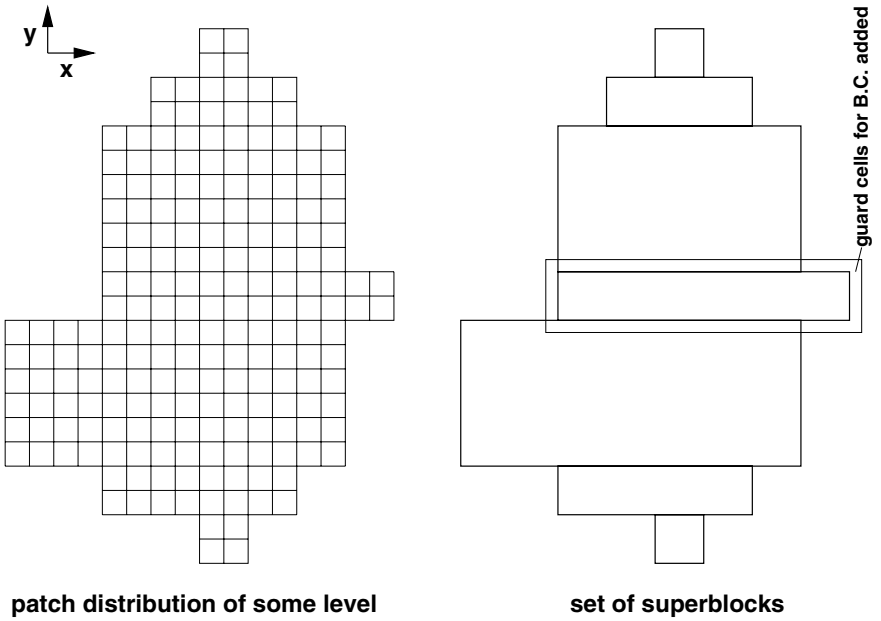


Fig. 2. The principle of patch clustering. In this example 7 superblocks are created

of cells in x-direction. Guard cells to take up boundary conditions are automatically added. Since a refinement level usually consists of several coherent regions with arbitrary bounding a whole set of superblocks is created. Depending on the degree of fragmentation of the grid structure, each refinement level may produce only a few superblocks or as much as hundreds or even thousands. Patch clustering is computationally inexpensive compared to the integration step. It just requires to collect the corresponding patches, to copy variables from these patches to the superblock and to assign boundary conditions.

The sequence of grids is time-advanced in a certain order. The finest level is integrated first. The rule then is that a level $l > 0$ is integrated twice before the coarser level $l - 1$ is integrated with a time-step being the sum of the finer time-steps. Of course, all individual time-steps obey the CFL condition. The last step in the procedure consists of updating the base grid. Whenever two adjacent levels match in time during this cycle synchronization of their solutions take place on consistency reasons and to maintain conservation properties of the core scheme. Synchronization is discussed next.

2.3 Synchronization processes

Several mechanism in the AMR method ensure that the solution remains consistent on the adaptive grid hierarchy during the course of evolution:

i) **restriction**. Since the fine grid solution is considered more accurate than the underlying coarse grid solution the latter is replaced by the fine grid solution by a conservative copy in those regions covered by the fine grid. In case of the magnetic field this means that the magnetic flux at fine cell faces is mapped onto coarse cell faces coinciding with those fine cell faces.
ii) **flux/electric field fixup**. The restriction procedure requires a fixup of the hydrodynamical fluxes and of the electric field at refinement interfaces. More precisely, the numerical flux at a coarse cell face has to be replaced by the temporal sum of the numerical fluxes of those fine cell faces which cover the coarse cell face. This restores conservation. Similarly, the electric field at a coarse cell edge must be replaced by the temporal sum of the electric fields of those fine cell edges which cover the coarse cell edge. This restores solenoidality of the magnetic field. However, the situation is complicated by the use of a *2-step* Runge-Kutta method. As a consequence, the fixup steps are carried out for both the predictor step and corrector step separately.
iii) **synchronization of internal boundary conditions**. The use of the 2-step Runge-Kutta time integrator or, more general, any multistage ODE solver requires additional synchronization. That is, after the predictor step boundary conditions along common interfaces of superblocks in direct contact need to be synchronized in order to avoid a time mismatch. For a single-step time integrator this would be redundant.

3 AMR efficiency

3.1 Definitions

Numerical errors are measured against a single-grid, high-resolution reference solution in a modified L_1-norm. I define the error in some quantity u as

$$\varepsilon_u = \frac{\sum_l |u_l - u_l^{\mathrm{ref}}|}{\sum_l |u_l^{\mathrm{ref}}|}$$

where the sum runs over all *effective* grid cells l of the grid hierarchy. Effective grid cells are all those not covered by finer cells. On a single grid without AMR l just stands for the grid indices $\{ijk\}$.

$$u_l^{\mathrm{ref}} = \frac{1}{\delta V_l} \sum_{\{ijk\} \in I^l} u_{ijk}^{\mathrm{ref}} \cdot \delta V_{ijk}^{\mathrm{ref}}$$

must be understood as an effective reference value for cell l (with volume element δV_l) obtained by volume-averaging reference values u_{ijk}^{ref} (with volume elements $\delta V_{ijk}^{\mathrm{ref}}$) in an environment I^l that contains all reference cells $\{ijk\}$ space-filling cell l ie. $\delta V_l = \sum_{\{ijk\} \in I^l} \delta V_{ijk}^{\mathrm{ref}}$.

The speedup factor S of an AMR application is defined as the inverse ratio of the CPU times of the AMR simulation and a comparison simulation with a global fine grid,

$$S = \frac{T_{comp}}{T_{AMR}}.$$

The speedup factor is often used to characterize the efficiency of an AMR implementation. As long as the numerical error in the AMR simulation is comparable to the error in the comparison simulation S is a sufficient number to do so. If this is not the case, however, because essential features of the solution are not resolved by the AMR algorithm the speedup S alone is no longer a proper measure to judge whether an AMR implementation is efficient or not. Instead, I propose to use the number

$$EFF = S \cdot \frac{\varepsilon_{comp}}{\varepsilon_{AMR}}$$

which involves errors for the comparison simulation and AMR simulation, respectively. Whereas S can be determined easily computing the errors requires the reference solution being performed with even higher resolution than the comparison simulation.

For the two 3D benchmark problems below (Sect. 3.2, 3.3) such high-resolution reference simulations could not be done because computer resources were not available. Yet, to get an estimate of the ratio $\varepsilon_{comp}/\varepsilon_{AMR}$ the following (inconsistent) procedure has been applied. First, convergence rates from lower resolved single-grid simulations are calculated using the comparison simulation as reference solution. With the obtained convergence rates one estimates a pseudo-ε_{comp} by extrapolation. Second, using the comparison simulation again as reference solution one can compute a pseudo-ε_{AMR}. If such pseudo-errors turn out of comparable size one can have the legitimated hope that $\varepsilon_{AMR} \approx \varepsilon_{comp}$ holds also for the true errors.

3.2 Implosion problem

The implosion problem considered here as a benchmark is a converging shock problem possessing spherical symmetry. The gas is confined to a cubic box $(x,y,z) = [-1,1]^3$ with reflecting boundary conditions on all six boundaries. The gas is initially at rest and has a constant density $\rho_i = 0.125$ and constant pressure $p_i = 0.1$ inside an interior sphere of radius $R = 0.5$ centered around the coordinate origin. Outside this sphere, $\rho_o = 1$, $p_o = 1$. A shock forms running to the center, rebounds and, thereafter, propagates radially outwards accompanied by a rarefaction wave and contact discontinuity.

Fig. 3 illustrates the result for an AMR simulation with a 50^3 base grid and two further levels of refinement at a time after the shock rebound. The effective resolution is thus 200^3. The density, pressure, velocity and patch distribution is shown in the $z = 0$ coordinate plane. The inner region $r \leq 0.5$ contains the shock and contact discontinuity which are fully covered by level 2 patches whereas the rarefaction is resolved by level 1 patches. Fig. 4 (left) shows a scatter plot of the density for both

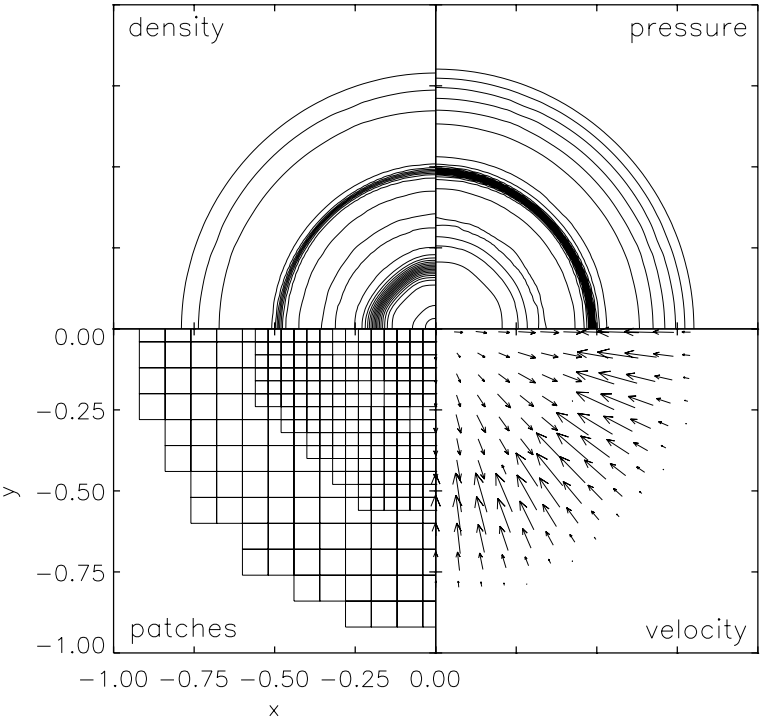


Fig. 3. Contour lines of density $(0.125 < \rho < 1.74)$ and pressure $(0.778 < p < 1.33)$, velocity field $(v_{\max} = 0.249)$ and patch distribution in the $z = 0$ coordinate plane for the AMR simulation. Only a quadrant of the spherically symmetric implosion problem is shown at a time

the AMR simulation and the 200^3 single-grid simulation. The curves are somewhat shifted relative to each other to make a comparison easier. Without shift the curves would coincide which gives a first (positive) impression on the accuracy of the AMR result. Fig. 4 (right) presents the time history of the kinetic energy for all simulations done. Again, there is a remarkable agreement between the AMR solution and the 200^3 comparison result. Errors, convergence rates and timings are listed in Table 1. The resolution study indicates an experimental order of convergence of roughly 1.5. This is surprisingly good taking note of the presence of discontinuities in the flow. Errors have been computed for the primitive variables ρ, p, v_x, v_y, v_z and are largest in the velocity. Pseudo-errors (see Sect. 3.1) for the 200^3 solution are comparable to those for AMR so that one can indeed argue, supported by Fig. 4, $\varepsilon_{\text{comp}} \approx \varepsilon_{\text{AMR}}$. From Table 1 the speedup factor $S = 9.1$ and, hence, an AMR efficiency of $\text{EFF} \approx 9$ results for the implosion problem. The AMR overhead containing all routines which were redundant in a single-grid simulation is 21.8% of the total runtime. A detailed

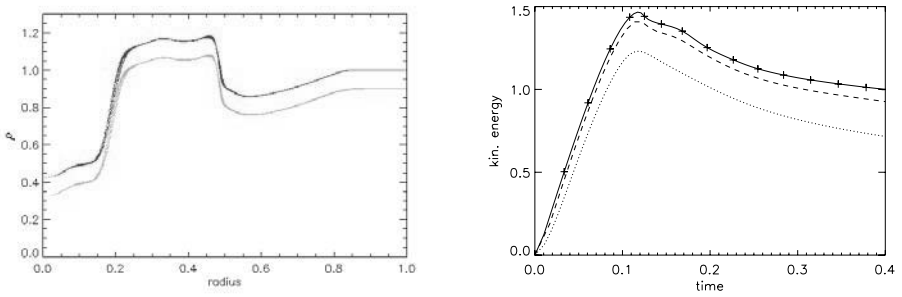


Fig. 4. Left: scatter plot of the density (ρ versus radius). Dark dots correspond to the reference simulation whereas light dots correspond to the AMR simulation which is shifted by $\Delta\rho = 0.1$ relative to the reference solution. Right: time history of the kinetic energy for an AMR simulation (plus signs), 200^3 simulation (solid), 100^3 simulation (dashed) and 50^3 simulation (dotted)

Table 1. Implosion problem – errors, convergence rates, timings

run	ε_ρ		ε_p		$\varepsilon_{v_x(,v_y,v_z)}$		TIME [s]
50^3	$6.5\cdot10^{-3}$	1.56	$8.5\cdot10^{-3}$	1.50	0.185	1.54	48
100^3	$2.2\cdot10^{-3}$		$3.0\cdot10^{-3}$		0.064		1587
A200	$6\cdot10^{-4}$		10^{-3}		0.029		2111
200^3	$7\cdot10^{-4\star}$		$1.1\cdot10^{-3\star}$		$0.022^\star$		19386

*pseudo-errors extrapolated from the measured convergence rate

Table 2. Implosion problem – AMR overhead

	% of runtime
mesh refinement: memory allocation, refinement criterion	5.5
patch clustering	2.0
mesh initialization: interpolation	6.4
synchronization: flux sync., restriction, boundary sync.	7.9
total	21.8

split of the total overhead in overhead due to the mesh refinement, patch clustering, mesh initialization and synchronization processes is given in Table 2.

3.3 Shock-cloud collision

The second benchmark problem is the collision of a density clump with a strong shock wave in a magnetic environment. The computational domain is a Cartesian box given by $(x,y,z) \in [-1/2,1/2]^3$. There is a discontinuity at $x = 0.1$ with left and right states:

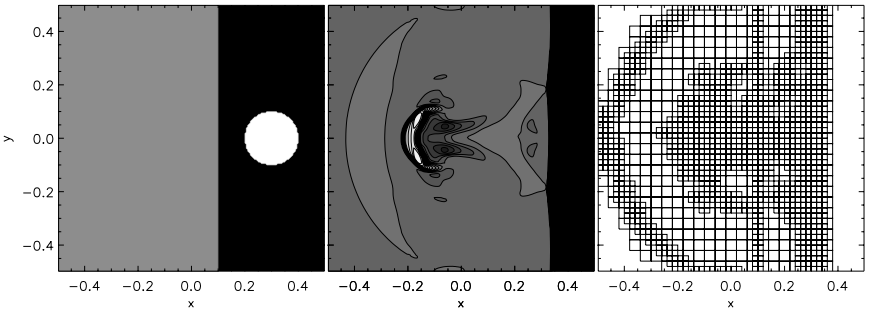


Fig. 5. Contour lines/grey-scale representation of $\log\rho$ at $t = 0$ (left panel, $0 < \log\rho < 1$) and after the collision (middle panel, $0 < \log\rho < 1.38$). Scaling is from minimum (black) to maximum (white). The right panel shows the resulting patch distribution spanning two levels of refinement

$$\begin{pmatrix} \rho \\ p \\ v_x \\ v_y \\ v_z \\ B_x \\ B_y \\ B_z \end{pmatrix}^L = \begin{pmatrix} 3.86859 \\ 167.345 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2.1826182 \\ -2.1826182 \end{pmatrix}, \qquad \begin{pmatrix} \rho \\ p \\ v_x \\ v_y \\ v_z \\ B_x \\ B_y \\ B_z \end{pmatrix}^R = \begin{pmatrix} 1 \\ 1 \\ -11.2536 \\ 0 \\ 0 \\ 0 \\ 0.56418958 \\ 0.56418958 \end{pmatrix}.$$

The flow is highly supersonic on the right side. At $\mathbf{x} = (0.3, 0, 0)$ a spherical density clump with radius 0.15 and density $\rho_{\rm cl} = 10$ is embedded. The clump is in pressure equilibrium with its surrounding medium. The adiabatic index $\gamma = 5/3$. Zero-gradient boundary conditions are used for all variables in all directions except at the upper x-boundary where variables are kept at their initial values. I perform an AMR simulation with a 50^3 base grid and two levels of refinement and single-grid simulations on a N^3 cube grid with $N = 50, 100, 200$. Hence, the effective resolution in the AMR case is identical to the 200^3 simulation. The simulations are stopped at evolution time $t = 0.06$ after which a violent collision between the shock and clump has taken place.

The initial- and resulting density structure is illustrated in Fig. 5. The gas cloud is strongly compressed, heated and significantly deformed. The magnetic field is dragged with the cloud and the resulting Lorentz force acting on the cloud in addition to pressure forces decelerates it. The corresponding growth in magnetic energy is recorded in Fig. 6. Fig. 5 (right panel) also shows the final patch distribution. All essential features occurring in the problem are well resolved, that is, the magnetic shock moving to the right opposite to the supersonic flow, the deformed cloud including its tail structure, the fast magnetosonic wave propagating ahead of the cloud and the stationary Alfvén discontinuity at $x = 0.1$ which is not seen in the density structure. Fig. 7 compares results of the several simulations in a cut along the x-axis.

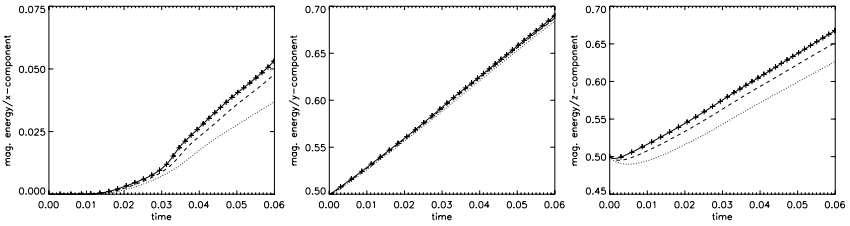


Fig. 6. Time history of the magnetic energy stored in the x-component (left), y-component (middle) and z-component (right). Solid line: 200^3 simulation, dashed line: 100^3 simulation, dotted line: 50^3 simulation, plus signs: AMR simulation

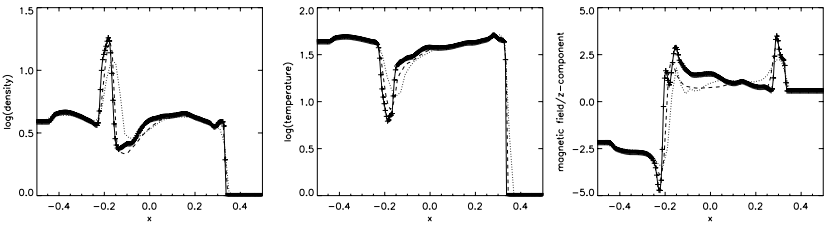


Fig. 7. Cut along the x-axis showing the variation in density (left), temperature (middle) and z-component of the magnetic field (right). Solid line: 200^3 simulation, dashed line: 100^3 simulation, dotted line: 50^3 simulation, plus signs: AMR simulation

Table 3. Shock-cloud collision – errors, convergence rates, timings

run	ε_ρ		ε_p		ε_v		ε_B		TIME [s]
50^3	0.0212	$\rangle 1.07$	0.0221	$\rangle 1.11$	0.2707	$\rangle 1.05$	0.2019	$\rangle 1.00$	194
100^3	0.0126		0.0097		0.1314		0.1032		5747
A200	0.0009		0.0015		0.0172		0.0068		22561
200^3	0.0048*		0.0045*		0.0630*		0.0513*		64485

*pseudo-errors calculated from the measured convergence rate

The logarithm of the density (left panel), logarithm of the temperature (middle panel) and z-component of the magnetic field (right panel) is shown. As in Fig. 6, note the excellent agreement between the AMR solution and 200^3 solution.

The single-grid simulations converge with a rate of ≈ 1 measured in all primitive variables. Using this rate and extrapolating errors one obtains pseudo-errors for the 200^3 case which are larger than the pseudo-errors for AMR taking the 200^3 simulation as reference solution. According to the ideas pointed out in Sect. 3.1 that may be a strong indication that the true errors are very close to each other ie. $\varepsilon_{\mathrm{AMR}} \approx \varepsilon_{\mathrm{comp}}$. From the timings listed in Table 3 one then derives an AMR efficiency of EFF ≈ 3. Finally, details of the AMR overhead for this problem can be found in Table 4.

Table 4. Shock-cloud collision – AMR overhead

	% of runtime
mesh refinement: memory allocation, refinement criterion	3.0
patch clustering	2.0
mesh initialization: interpolation	6.2
synchronization: flux sync., restriction, boundary sync.	6.9
total	18.0

References

1. J.U. Brackbill, D.C. Barnes, J. Comput. Phys. **35**, 426 (1980).
2. D.S. Balsara, D.S. Spicer, J. Comput. Phys. **149**, 270 (1999).
3. M. Berger, P. Collela, J. Comput. Phys. **82**, 64 (1989).
4. M. Berger, R.J. LeVeque, SIAM J. Numer. Anal. **35**, 2298 (1998).
5. J. Bell, M. Berger, J. Saltzmann, M. Welcome, SIAM J. Sci. Comput. **15**, 127 (1994).
6. C.R. Evans, J.F. Hawley, Astrophys. J. **332**, 659 (1988).
7. H. Friedel, R. Grauer, C. Marliani, J. Comput. Phys. **134**, 190 (1997).
8. A. Kurganov, S. Noelle, G. Petrova, SIAM J. Sci. Comput. **23**, 707 (2001).
9. K.G. Powell, P.L. Roe, T.J. Linde, T.I. Gombosi, D.L. De Zeeuw, J. Comput. Phys. **154**, 284 (1999).
10. O. Steiner, M. Knölker, M. Schüssler, Dynamic interaction of convection with magnetic flux sheets: first results of a new MHD code, in: Solar Surface Magnetism, Kluwer, Dordrecht, eds. R.J. Rutten, C.J. Schrijver (1993).
11. R. Walder, D. Folini, A-MAZE: A code package to compute 3D magnetic flows, 3D NLTE radiative transfer, and synthetic spectra, in: Thermal and Ionisation Aspects of Flows from Hot Stars: Observations and Theory, in: ASP Conference Series, Vol. **204**, 281 (2000).
12. U. Ziegler, J. Comput. Phys. subm.

Dynamic Load Balancing of SAMR Applications

Zhiling Lan[1] and Valerie E. Taylor[2]

[1] Illinois Institute of Technology lan@iit.edu
[2] Texas A&M University taylor@cs.tamu.edu

1 Introduction

SAMR is a type of multiscale algorithm that achieves high resolution in localized regions of adaptive simulations. It shows incredible potential as a means of expanding the tractability of a variety of numerical experiments and has been successfully applied to simulate many phenomena that are out of reach with fixed-grid methods. Execution of SAMR applications on parallel and distributed systems involves dynamically distributing the workload among the available processors at runtime. In this paper, we present two novel dynamic load balancing schemes for SAMR applications: one is for parallel systems and the other is for distributed systems.

With any DLB scheme, the major issues to be addressed are the identification of overloaded versus underloaded processors, the amount of data to be redistributed from the overloaded processors to the underloaded processors, and the overhead that the DLB scheme imposes on the application. In investigating DLB schemes, we first analyze the requirements imposed by the applications. In particular, we complete a detailed analysis of ENZO application, a parallel implementation of SAMR in astrophysics and cosmology, and identify the unique characteristics that impose severe challenges on DLB schemes. The results of the detailed analysis of ENZO provide four unique adaptive characteristics relating to DLB requirements: (1) coarse granularity, (2) high magnitude of imbalance, (3) different patterns of imbalance, and (4) high frequency of adaptations. In addition, ENZO employs an implementation that maintains some global information.

Due to the fact that SAMR applications have the above unique adaptive characteristics, most of the existing DLB schemes [Cyb89, EBP00, LK87, OB97, SKK97, SS94, Wal94, WLR93] are not efficient for them. Therefore, we propose a novel dynamic load balancing scheme for SAMR applications on parallel systems (denoted as *parallel DLB*). It interleaves a grid-splitting technique with direct grid movements, for which the objective is to efficiently redistribute workload among all the processors so as to reduce the parallel execution time.

The emergence of high-performance distributed systems provides an economical platform to traditional parallel systems. By using distributed systems, researchers

are able to execute applications that require vast computing power (e.g., beyond that available at any single site). A distributed system is defined as a computing system consisting of at least two autonomous computers linked by computer networks[FK99], thus most of the available DLB schemes designed for homogeneous parallel systems are inadequate for distributed systems. For example, some schemes assume the multiprocessor system to be homogeneous, (e.g. all the processors have the same performance and the underlying networks are dedicated and have the same performance). Some schemes consider the system to be heterogeneous in a limited way (e.g. the processors may have different performance).

In order to efficiently utilize the computing resources provided by distributed systems, an underlying DLB scheme must consider the heterogeneous and dynamic features of distributed systems. In this paper, a dynamic load balancing scheme is proposed for SAMR application on distributed systems (denoted as *distributed DLB*). It takes into consideration: (1) heterogeneity of processors, (2) heterogeneity of networks, (3) shared nature of networks, and (4) adaptive characteristics of the application running on the distributed system. Basically, *distributed DLB* divides load balancing process into two phases: *global balancing phase* and *local balancing phase*. Heuristic methods are proposed to evaluate computational gain and redistribution cost for the *global balancing phase*.

2 Overview of SAMR and ENZO Code

This section gives an overview of the SAMR method, developed by M. Berger et al., and the ENZO code, a parallel implementation of this method for astrophysical and cosmological applications. Additional details about ENZO and the SAMR method can be found in [BC89, Bry99, BAN01].

2.1 Layout of Grid Hierarchy

SAMR represents the grid hierarchy as a tree of grids at any instant in time. The number of levels, the number of grids, and the locations of the grids change with each adaptation. Initially, a uniform mesh covers the entire computational domain. During the computation, finer grids are added in regions that require higher resolution. This process repeats recursively with each adaptation resulting in a tree of grids like that shown in Figure 1. The top graph in this figure shows the overall structure after several adaptations. The remainder of the figure shows the grid hierarchy for the overall structure with the dotted regions corresponding to those that require further refinement. In this grid hierarchy, there are four levels of grids from level 0 to level 3. Throughout execution of a SAMR application, the grid hierarchy changes with each adaptation.

For simplification, SAMR imposes some restrictions on the new subgrids. A subgrid must be uniform, rectangular, aligned with its parent grid, and completely contained within its parent. All parent cells are either completely refined or completely unrefined. Lastly, the refinement factor must be an integer [Bry99].

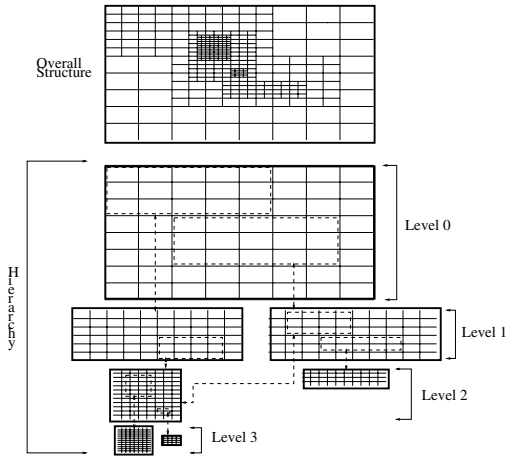


Fig. 1. SAMR Grid Hierarchy

The SAMR integration algorithm goes through the various adaptation levels advancing each level by an appropriate time step, then recursively advancing to the next finer level at a smaller time step until it reaches the same physical time as that of the current level.

2.2 ENZO: A Parallel Implementation of SAMR

ENZO [Bry99] is one of the successful parallel implementations of SAMR, which is primarily intended for use in astrophysics and cosmology. It entails solving the coupled equations of gas dynamics, collisionless dark matter dynamics, self-gravity, and cosmic expansion in three dimensions and at high spatial resolution. The code is written in C++ with Fortran routines for computationally intensive sections and MPI functions for message passing among processors. ENZO was developed as a community code and is currently in use at over six sites.

The ENZO implementation manages the grid hierarchy globally; that is, each processor stores the grid information of all other processors. In order to save space and reduce communication time, the notation of "real" grid and "fake" grid is used for sharing grid information among processors. Each subgrid in the grid hierarchy resides on one processor and this processor holds the "real" subgrid. All other processors have replicates of this "real" subgrid, called "fake" grids. Usually, the "fake" grid contains the information such as dimensional size of the "real" grid and the processor where the "real" grid resides. The data associated with a "fake" grid is small (usually a few hundred bytes), while the amount of data associated with a "real" grid is large (ranging from several hundred kilobytes to dozens of megabytes).

3 Parallel DLB Scheme

After taking into consideration the adaptive characteristics of the SAMR application, we developed a novel DLB scheme which interleaves a grid-splitting option with direct data movement(denoted as *parallel DLB*).

3.1 Description

In this scheme, each load balancing step consists of one or more iterations of two phases: *moving-grid phase* and *splitting-grid phase*. The *moving-grid phase* redistributes grids directly from overloaded processors to underloaded processors by the guidance of the global information; and the *splitting-grid phase* splits a grid into two smaller grids along the longest dimension. Figure 2 gives the pseudocode of our scheme, and the details are given below.

DLB Algorithm

```
Done = 0; LastMax=LastMin=0;
while (MaxLoad > threshhold * AvgLoad  && Done == 0 ) {
    for (j=0;j<NumberOfGrids;j++)    {                          //moving-grid phase
        for (i=0;i<NumberOfGrids;i++) {
            if (grid(i) resides on MaxProc && grid(i) > AvgLoad/threshold-MinLoad
                                          && grid(i) < AvgLoad*threshold -MinLoad) {
                move grid(i) from MaxProc to MinProc;
                update load information of MaxProc and MinProc;
                find new MaxProc and MinProc;
                break;
            }
        }
        if (i == NumberOfGrids)
            break;
    }

    if (MaxLoad > threshold * AvgLoad ) {                       //splitting-grid phase
        if (MaxProc == LastMax && MinProc == LastMin)
            Done = 1;
        LastMax = MaxProc; LastMin = MinProc;
        find the largest grid MaxGrid on MaxProc;
        split MaxGrid into two and one of them redistributed to MinProc;
        update MaxProc, MinProc, MaxLoad, MinLoad, and AvgLoad;
    } else {
        Done = 1;
    }
}
```

Fig. 2. Pseudo-code of Proposed Parallel DLB Scheme

- *moving-grid phase:* After each adaptation, *parallel DLB* is triggered by checking whether $MaxLoad/AvgLoad > threshold$. The *MaxProc* moves its grid directly to *MinProc* under the condition that the redistribution of this grid will make the workload of *MinProc* reach *AvgLoad*. Here, *AvgLoad* denotes the required load for which all the processors would have an equal load. Thus, if there is a suitable sized grid, one direct grid movement is enough to balance an underloaded processor by utilizing the global information. This phase continues until either the

load balancing ratio is satisfied or no grid residing on the *MaxProc* is suitable to be moved.

- *splitting-grid phase:* If no more direct grid movements can be employed and imbalance still exists, the *splitting-grid phase* will be invoked. First, the *MaxProc* finds the largest grid it owns (denoted as *MaxGrid*). If the size of *MaxGrid* is no more than $(AvgLoad - MinLoad)$ which is the amount of load needed by *MinProc*, the grid will be moved directly to *MinProc* from *MaxProc*; otherwise, *MaxProc* splits this grid along the longest dimension into two smaller grids. One of the two split grids, whose size is about $(AvgLoad - MinLoad)$, is redistributed to *MinProc*. After such a splitting step, *MinProc* reaches the average load. Note that splitting does not mean splitting into equal pieces. Instead, the splitting is done exactly to fill the "hole" on the *MinProc*.

If the imbalance still exists, another attempt of interleaving *moving-grid phase* and *splitting-grid phase* will continue. Note that both the *moving-grid phase* and *splitting-grid phase* execute in parallel. Eventually, either the load is balanced, which is our goal, or there are not enough grids to be redistributed among all the processors.

Parallel DLB makes sure that each grid movement makes an underloaded procrssor reach, but not exceed, the average load. Further, the use of global load information to move and split the grids eliminates the variability in time to reach the equal balance and avoids chances of *thrashing*. In other words, the situation that multiple overloaded processors send their workload to an underloaded processor and make it overloaded will not occur by using the proposed DLB.

In order to minimize the overhead of the scheme, *nonblocking communication* is explored in this scheme. In the mode of nonblocking communication, a nonblocking post-send initiates a send operation and returns before the message is copied out of the send buffer. A separate complete-send call is needed to complete the communication. The nonblocking receive is proceeded similarly. In this manner, the transfer of data may proceed concurrently with computations done at both the sender and the receiver sides.

3.2 Performance Results

The proposed scheme was implemented in the ENZO code and examined by executing it on the SGI Origin2000 machines at NCSA. Our experiments show that by using *parallel DLB*, the parallel execution time of ENZO can be reduced by up to 57% and the quality of load balancing (defined as $\frac{MaxLoad}{AvgLoad}$) can be improved by a factor of six, as compared to the original DLB scheme used in ENZO. More details about the experiments can be found in [LTB02].

A parameter called *threshold* is used in *parallel DLB* (see Figure 2), which determines whether a load-balancing process should be invoked after each refinement. Intuitively, the ideal value should be 1.0, which means all the processors are evenly and equally balanced. However, the closer this threshold is to 1.0, the more load-balancing actions are entailed, so the more overhead may be introduced. Further, for SAMR applications, the basic entity is a "grid" which has a minimal size requirement. Thus the ideal situation in which the load is perfectly balanced may not

be obtained. The *threshold* is used to adjust the quality of load balancing, whose value influences the efficiency of the overall DLB scheme. Our sensitivity analysis in [LTB02] indicates that setting it to 1.25 results in the best performance in terms of execution time and quality of load balancing.

4 Distributed DLB Scheme

After taking into consideration the major issues related to distributed computing, we propose a dynamic load balancing scheme for SMAR applications on distributed systems (denoted as *distributed DLB*).

4.1 Description

In this scheme, a distributed system is divided into multiple groups. Here, a "group" is defined as a set of processors that have the same performance and share a dedicated intra-connected network; in other word, a group is a homogeneous system connected with dedicated system-area networks [CT00]. A group can be a shared-memory parallel computer, a distributed-memory parallel computer, or a cluster of workstations. Communications within a group are referred as local communication, and those between different groups are remote communications.

Figure 3 illustrates the overall structure of *distributed DLB*. To address the heterogeneity of networks, a two-level approach is explored for load balancing, that is, the load balancing process is divided into two phases: *global balancing phase* and *local balancing phase*. Each phase consists of three steps: *imbalance detection*, *evaluation process*, and *redistribution process*. The fundamental objective is to minimize remote communication as well as balance the workload among processors. During the *global balancing phase*, the proposed scheme interacts with the whole system which includes all the processors and the networks connected them and the entire computational domain. However, during the *local balancing phase*, each process only focuses on a portion of the computational domain (as shown by the dotted arrow lines) and interacts with a single machine.

Distributed DLB address the heterogeneity of processors by generating a relative performance weight for each processor. When distributing workload among processors, the load is balanced proportional to these weights. One of the key issues for *global balancing phase* is to decide when such an action should be performed and whether it is advantageous to do so. Heuristic methods have been proposed to evaluate the computational gain and the redistribution cost for global redistributions. The scheme addresses the dynamic features of networks by adaptively choosing an appropriate action based on the current observation of the traffic on the networks[LTB01].

4.2 Performance Results

The proposed *distributed DLB* was implemented in the ENZO code and tested on three different distributed environments including a LAN-connect system, a WAN-

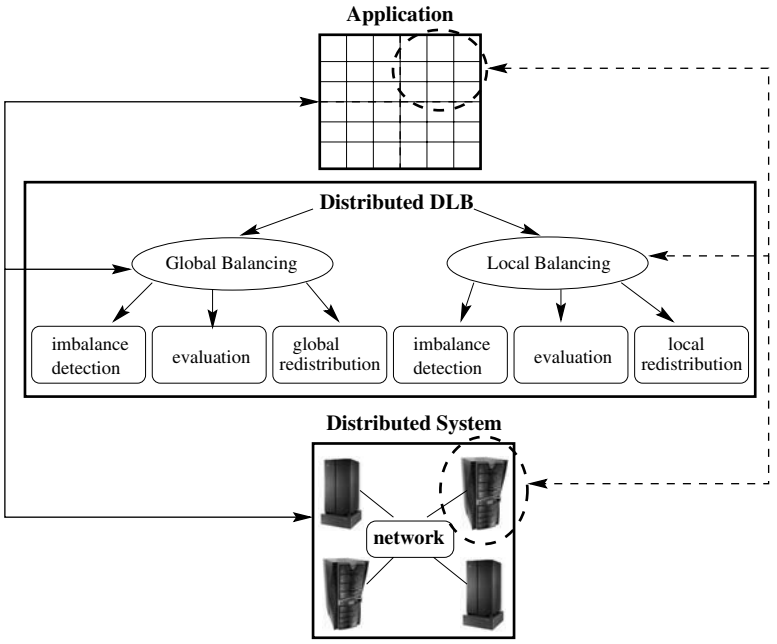


Fig. 3. Overall Structure of *distributed DLB*

connected system across two cities, and a WAN-connected system across two countries. Our experiments show that by using *distributed DLB*, the total execution time can be reduced by 9%-56% and the average improvement is more than 26%, as compared with using *parallel DLB* which does not consider the heterogeneous and dynamic features of distributed systems. Further, the quality of load balancing can be either maintained or improved as well, especially when heterogeneous processors are used.

5 Summary and Future Work

In this paper, we presented two novel dynamic load balancing schemes for SAMR applications: one is for parallel systems denoted as *parallel DLB* and the other is for distributed systems denoted as *distributed DLB*. *Parallel DLB* scheme divides the load balancing process into two steps: *moving-grid phase* and *splitting-grid phase*. *Distributed DLB* scheme takes into consideration: (1) heterogeneity of processors, (2) heterogeneity of networks, (3) shared nature of networks, and (4) adaptive characteristics of the application running on the distributed system. Experiments show that both DLB schemes can significantly improve the performance of SAMR applications on parallel and distributed systems, in particular, the ENZO code.

Currently, we are working on exploring large-scale cosmology application on both the nation-wide TeraGrid[tg] and the Illinois-based DOT (Distributed Optical

Testbed)[TFM+04]. Further, we also work on generalizing the proposed scheme, with the goal of developing a general and extensible dynamic load balancing tool to be used with a variety of large-scale adaptive applications on distributed environments.

References

BAN01. G. Bryan, T. Abel, and M. Norman. Achieving extreme resolution in numerical cosmology using adaptive mesh refinement: Resolving primordial star formation. In *Proc. of SC2001*, Denver, CO, 2001.

BC89. M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, May 1989.

Bry99. G. Bryan. Fluid in the universe: Adaptive mesh refinement in cosmology. *Computing in Science and Engineering*, 1(2):46–53, March/April 1999.

CT00. J. Chen and V. Taylor. Part: Mesh partitioning for efficient use of distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 12:111–123, 2000.

Cyb89. G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 7:279–301, October 1989.

EBP00. R. Elsasser, B.Monien, and R. Preis. Diffusive load balancing schemes for heterogeneous networks. In *Proc. of SPAA*, 2000.

FK99. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, California, 1999.

LK87. F. Lin and R. Keller. The gradient model load balancing methods. *IEEE Transactions on Software Engineering*, 13(1):8–12, January 1987.

LTB01. Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of samr applications on distributed systems. In *Proc. of SC2001*, Denver, CO, 2001.

LTB02. Z. Lan, V. Taylor, and G. Bryan. A novel dynamic load balancing scheme for parallel systems. *Journal of Parallel and Distributed Computing*, page 1763 1781, 2002.

OB97. L. Oliker and R. Biswas. Plum: Parallel load balancing for adaptive refined meshes. *Journal of Parallel and Distributed Computing*, 47(2):109–124, 1997.

SKK97. K. Schloegel, G. Karypis, and V. Kumar. Multilevel diffusion schemes for repartitioning of adaptive meshes. *Journal of Parallel and Distributed Computing*, 47(2):109–124, 1997.

SS94. A. Sohn and H. Simon. Jove: A dynamic load balancing framework for adaptive computations on an sp-2 distributed multiprocessor. In *NJIT CIS Technical Report*, New Jersey, 1994.

TFM+04. V. Taylor(PI), I. Foster, J. Mambretti, P. Dinda, X. Sun, and A. Choudhary. *DOT: Distributed Optical Testbed to Facilitate the Development of Techniques for Efficient Execution of Distributed Applications*. NSF EIA-0224377, 2002 - 2004.

tg. *The NSF TeraGrid*. http://www.teragrid.org.

Wal94. C. Walshaw. Jostle: Partitioning of unstructured meshes for massively parallel machines. In *Proc. Parallel CFD'94*, 1994.

WLR93. M. Willebeek-LeMair and A. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, 4(9):979–993, September 1993.

Part IV

Applications

The Impact of AMR in Numerical Astrophysics and Cosmology

Michael L. Norman

Laboratory for Computational Astrophysics at the Center for Astrophysics and Space Sciences, University of California at San Diego, La Jolla, CA 92093, USA

Abstract. I survey the use and impact of adaptive mesh refinement (AMR) simulations in numerical astrophysics and cosmology. Two basic techniques are in use to extend the dynamic range of Eulerian grid simulations in multi-dimensions: cell refinement, and patch refinement, otherwise known as block-structured adaptive mesh refinement (SAMR). In this survey, no attempt is made to assess the relative merits of these two approaches. Rather, the discussion focuses on how AMR is being used and how AMR is making a scientific impact in a diverse set of fields from space physics to the cosmology of the early universe. The increased adoption of AMR techniques in the past decade is driven in part by the public availability of AMR codes and frameworks. I provide a partial list of resources for those interested in learning more about AMR simulations.

1 Introduction

Since its introduction roughly 20 years ago [1], adaptive mesh refinement (AMR) has emerged as an important class of numerical techniques for improving the accuracy and dynamic range of grid-based calculations for fluid dynamics problems. Such problems, especially compressible flow, develop steep gradients (shock waves and contact discontinuities) which, in the absence of mesh refinement, become sources of error for the global solution (e.g., [2]). Through appropriate local mesh refinement, AMR can be thought of as a numerical technique for optimizing the quality of a numerical solution for a given computational cost (e.g., [3]).

The last ten years have seen the application of AMR methods to problems in space physics [4, 5, 6], astrophysics [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17], and cosmology [18, 19, 20, 21, 22, 23, 24, 25, 26]. Here, a variety of physical processes may operate singly or together in astrophysical fluids to expand the range of important length- and time-scales. Such processes include gravity and gravitational instability, reaction kinetics, magnetic reconnection, radiation transfer, ionization fronts, etc. AMR has also been applied to the solution of gravitational N-body problems

[27, 20], and to hybrid particle-fluid simulations in cosmology [18, 28]. In such applications, the physics is intrinsically multi-scale, and AMR can be thought of as a numerical technique for extending the dynamic range of resolved physics, regardless of the computational cost. It is these sorts of applications which are reviewed here, and where AMR can make a scientific and not just an economic impact.

AMR is also having a positive impact on the methodology of computational physics itself. I am referring to the validation of computational codes through resolution studies. With AMR, it is now practical on today's supercomputers to perform resolution studies over a sufficient range of scales to obtain convergent results on properties of interest which may be compared with laboratory experiments ([17]).

Table 1 shows the impressive diversity of topics AMR has been applied to. Particularly interesting is the range of physical processes AMR has been adapted to model. This is listed in the second column of Table 1. As of today, AMR has been successfully applied to ideal gas dynamics (Newtonian and special relativistic), reactive gas dynamics, MHD (ideal and resistive), self-gravitating gas dynamics and MHD, N-body dynamics, and hybrid fluid/N-body systems. As we heard at this conference, methods are under development for radiative transfer (Howell), radiation hydrodynamics (Weaver), and solid mechanics (Falle). It is clear AMR is a method of wide applicability, and one that is growing in its impact. This growth is fueled in part by the public availability of AMR codes and frameworks (see Section 3).

Table 1. Classes of AMR applications.

Topic	Physics	Select References
Code validation	HD, reactive HD	[17]
Solar and space physics	MHD	[5]
Supernovae and nucleosynthesis	reactive HD	[29, 30, 31]
Interstellar medium	HD, MHD	[9, 15, 32, 33]
Star formation	grav HD, grav RHD	[13, 11]
Astrophysical jets	HD, rel HD	[8, 34, 35]
N-body dynamics	particles, grav	[20, 21]
Hydrodynamic cosmology	hybrid	[18, 26]

In this paper I survey the use and impact of adaptive mesh refinement simulations in numerical astrophysics and cosmology. Two basic techniques are in use to extend the dynamic range of Eulerian grid simulations in multi-dimensions: cell refinement (CR), and patch refinement (PR), otherwise known as block-structured adaptive mesh refinement (SAMR). Details of these two approaches are given elsewhere in this volume. In this review, no attempt is made to assess the relative merits of these two approaches. Rather, the discussion focuses on how AMR is being used and how AMR is making a scientific impact in a diverse set of fields from space physics to the cosmology of the early universe. At the end, I provide a partial list of software resources for those interested in learning more about AMR simulations. [**NB.** The following is meant to be representative rather than complete. I apologize to authors in advance if their work is not mentioned.]

2 Applications of AMR

2.1 Solar and Space Physics

Coronal mass ejections (CMEs) are transient solar events in which mass and magnetic field are ejected from the solar surface. These dynamic events originate in closed magnetic field regions of the corona. They produce large-scale reconfigurations of the coronal magnetic field and generate solar wind disturbances which manifest as geomagnetic storms here on Earth. Groth et al. [5, 6] have applied AMR to 3D simulations of solar coronal outflows and ejections. The central question which motivates their research is what is the mechanism and timescale for coronal mass ejections (CMEs)? The simulations solve the equations of ideal MHD in 3D Cartesian geometry, supplemented with the Sun's gravitational field and a coronal heating term. The equations are solved using the upwind, cell-centered, finite volume scheme of Powell [36, 37], which is a Godunov-type scheme for ideal MHD. Numerical fluxes are computed using the approximate Riemann solvers of both Roe [38] and Harten et al. [39], adapted to the MHD eigensystem. The MHD solver is married to the parallel AMR framework of de Zeeuw et al. [4]. In this approach, the base grid is decomposed into blocks of constant size, each of which is assigned to a separate processor. If any cell within a block are flagged for refinement, the entire block is refined by a factor of two in each dimension, resulting in eight sub-blocks of the size of the parent block.

The CME problem is initialized with a model of the quiescent solar wind which involves a high speed wind and open magnetic field lines at high solar latitudes, and a lower wind speed and closed magnetic field lines in the equatorial region. The CME event is triggered with the introduction into the computational domain of a density pulse in the closed field region. The mass loading inflates the closed field region until it bursts open. Uisng AMR, they are able to follow the CME outburst to $\frac{1}{2}$ AU. According to the authors, the benefits of AMR in this demonstration calculation are cost/memory savings, and the ability to refine a region of interest (the magnetic X-point) which moves through the computational volume. Although they were unable to answer the ultimate question, the authors argue that AMR combined with more realistic solar magnetic field configurations and initiation mechanisms will lead to improved understanding of CMEs.

2.2 Supernovae and Nucleosynthesis

An important and growing class of applications for AMR simulations is calculating the mechanisms and chemical yields of supernovae models in three dimensions. In recent years, the importance of convective/turbulent motions in both Type Ia (thermonuclear) and Type II (core collapse) supernovae has been recognized [40, 41]. Fully 3D simulations are thus required, and AMR serves two fundamental roles. The first is reducing memory and cpu requirements for expensive 3D simulations, and the other is resolving and tracking dynamic interfaces where energy generation and nucleosynthesis takes place.

Type Ia supernovae are believed to result from the explosive burning of carbon and oxygen in a white dwarf (WD) accreting matter from a binary companion. As the WD's mass approaches the Chandrasekhar limit, a small mass increase causes a substantial contraction of the star. The compression raises the temperature and thereby thermonuclear reaction rates, liberating energy which raises the temperature yet further. Because the WD is degenerate, a thermonuclear runaway ensues until thermal pressure becomes comparable to the degenerate electron pressure. At this point, the WD begins to expand, but not so fast that the thermonuclear reactions are quenched. According to models [42, 31], a nuclear burning front propagates outward from the center of the expanding WD liberating of order 10^{51} ergs of energy—sufficient to unbind the star.

Gamezo et al. [31] have developed a 3D AMR code to simulate the physics described above, and have addressed several key questions: (1) How does a degenerate C/O white dwarf explode and with what energy? (2) What is the nature and structure of the burning front? And (3) what fraction of the WD is burned? They solve Euler's equations of gas dynamics for four reactive species coupled to a nuclear reaction network. The energy equation includes nuclear energy generation and neutrino losses, as well as electron thermal conduction. A key additional incredient to the physical model is a flame capturing technique which adds one additional PDE to be solved for a reaction progress variable. The system is evolved for one octant of the WD star on a 3D Cartesian grid using a cell-refinement AMR technique called FTT (Fully Threaded Tree) developed by Khokhlov [14]. The benefit of cell-refinement over SAMR in this application is that the region of interest is a surface—the flame front—which is more efficiently captured with a cell-refinement technique [14].

Gamezo et al. find that the nuclear flame front is a deflagration front, meaning it advances subsonically into the unburned gas. The flame front is Rayleigh-Taylor (hereafter, RT) unstable and becomes highly convoluted, resembling the head of a cauliflower. AMR is used to track this highly distorted surface as it advances into the unburned star. They find that a healthy explosion results, but that a substantial fraction of the WD remains unburned at disassembly. This is contrary to observations, which are found to be more consistent with detonating WD models (e.g., [43, 44]). Gamezo et al. speculate that the turbulent flame may trigger a detonation at some point, which would complete the burning and remove this discrepancy. The physics of deflagration to detonation transition (DDT) depends on resolving the formation of "hot spots" behind the turbulent flame front on scales comparable to the flame front thickness [45]. This is tiny compared to the radius of the WD, and thus not captured by global simulations despite the use of AMR. At present DDT must be studied using simulations in small volumes ("subscale simulations".)

An example of this is the work of Timmes et al. [30], who used the FLASH code [46] to study the cellular structure of carbon detonation waves in degenerate WDs. Their goal was to assess the effect of numerical resolution on the size and shape of the detonation cells and the shock wave interactions that create them. FLASH is a block-structured AMR code combining the piecewise parabolic method for gas dynamics, nuclear reactions, and the PARAMESH AMR library [46]. The simulation was done in a 2D column of gas with conditions representative of the WD. The detonation was

initiated by sending a shock wave into the column with strength consistent with a steady 1D detonation wave ("CJ wave"). AMR was used to refine by up to a factor of 8 in cell size the reaction zone behind the shock. AMR thus provided the ability to have high resolution just in the detonation wave and track it as it propagated many times its width.

Timmes et al. were able to converge on the size and shape of the detonation cells with rather modest resolution (about 20 cells per burning length scale), but found that the size and distribution of pockets of unburned fuel was very sensitive to resolution. As long as this scale is small compared to the WD radius, observations would average over these compositional differences. However, the burning length scale and hence detonation cell size becomes comparable to the WD radius as density drops toward the edge of the star [47]. If the spectra of Type Ia supernovae reflect large scale abundance variations due to incomplete combustion, the results of Timmes et al would help define the minimum resolution requires to converge on the inhomogeneities.

In a related work, Zingale et al. [48] have applied FLASH to the dynamics of helium detonation on the surface of a neutron star in 2D. In this application, as in Gamezo et al., AMR provides the ability to resolve and track the nuclear burning front in an expanding envelope of gas.

AMR is also making an impact in the understanding of the iron core collapse supernova explosion mechanism and subsequent explosive nucleosynthesis. Supernovae classified by observers as Type Ib and II are believed to be powered by the copious flux of neutrinos emitted as the iron core of a massive star implodes to form a neutron star or black hole [49]. Earlier 2D simulations showed that neutrino heating sets up convective motions in material accreting onto the proto-neutron star/black hole, breaking spherical symmetry [41, 50]. The consequence of this is that the shell of 56Ni which forms just outside this convective region and is later ejected by the supernovae is highly perturbed [29].

AMR has not yet been applied to the very difficult problem of neutrino transport/heating in a 3D convective flow, although a large collaboration is attempting this [51]. Rather, Kifonidis et al. [29, 52] have used AMR to simulate the evolution of the lumpy nickel shell as it is ejected by the explosion. AMR is used to track the expansion of the shell over a factor of 100 in radius and its fragmentation by the RT instability while maintaining high resolution in the shell. The authors map the results of a 2D axisymmetric core collapse simulation 30 ms after bounce into AMRA, a 2D AMR hydro code developed by Plewa & Müller [53]. The equations of multispecies reactive hydrodynamics are solved on an adaptive spherical polar grid using the PPM algorithm [54] and the Consistent Multifluid Advection (CMA) scheme of Plewa & Müller. The latter minimizes numerical diffusion of nuclear species while preserving local mass conservation. Kifonidis et al. find that the 56Ni and other newly formed iron group elements are distributed throughout the inner half of the helium core by RT instabilities operating at the (Ni+Si)/O and (C+O)/He shell interfaces seeded by perturbations from convective overturn during the early stages of the explosion. Using AMR, they are able to carry the calculation out to 20,000 sec post-bounce, follow the details of RT growth and mixing. The result is that fast-moving clumps of 56Ni

are formed with velocities up to 4000 km/s. This offers a natural explanation for the mixing required in light-curve and spectroscopic modeling of Type Ib explosions, including SN1987a.

2.3 Interstellar Medium

The essential complexity of the interstellar medium (ISM) stems from the fact that it is not in equilibrium, dynamically or thermodynamically. A variety of heating and cooling processes operate to split the ISM into multiple thermal phases, each with their own characteristic densities, temperatures, and evolutionary timescales. Self-gravity concentrates the cold, dense phase into molecular clouds, which birth new stars. The most massive stars create large amplitude disturbances in the local heating rate through supernova shocks and ionization fronts, changing the dynamical and thermodynamical state of the gas. To simulate the ISM is rather akin to simulating the Earth's weather, which is affected by both local and global influences, and exhibits complexity in space and time on a vast range of scales. With AMR simulations, we may eventually be able to build an integrated model of the ISM which captures both its structural and statistical properties. At present, researchers are looking at individual processes in greater detail than ever before that ultimately may become part of an integrated model.

The propagation of interstellar shock waves in the ISM has received considerable attention because of their central role as a source of heat and momentum to the interstellar gas. The phenomenology is rich because the ISM is inhomogeneous, and the shocks themselves are typically radiative. A set-piece calculation is the interaction of a strong planar shock wave striking an isolated interstellar cloud, idealized as homogeneous and spherical. The interaction results in the compression and ultimate shredding of the cloud by a combination of Richtmyer-Meshkov, Rayleigh-Taylor, and Kelvin-Helmholtz instabilities. Since these instabilities all grow on the shock-accelerated cloud-intercloud interface, it is important that the interface be tracked with high resolution.

The first AMR simulations were performed by Klein, Mc Kee & Colella [9]. They wanted to calculate how long a shocked cloud would remain intact before mixing into the ISM. They solved the gas dynamical problem in 2D using block-structured AMR and a second-order Godunov hydrodynamics scheme as described in [3]. They found that for strong incident shocks, the evolution is determined by a single parameter: the ratio of cloud to intercloud densities. Using AMR, they performed a resolution study to determine the minimum resolution needed to capture the most destructive modes of the instabilities. They found that a minimum of 100 cells per cloud radius are needed with their second order-accurate scheme, and that the cloud is totally fragmented.

A closely related problem is the long-term evolution of the dense, metal-rich clumps ejected by supernova explosions (cf. [52]). Because dense clumps are decelerated less than the diffuse interclump ejecta, they catch up to the supernova remnant (SNR) shell and puncture it from behind. The clump first encounters the reverse shock, traverses the high pressure intershock region, and then exits the forward shock

if it survives to encounter the onrushing ISM. Cid-Fernandes et al. [32] used the 2D AMR code AMRA [53] to simulate the evolution of a single clump with properties appropriate to a Ni clump from a core collapse SN. They initialized the freely expanding envelope of a Type II SN in spherical polar coordinates, assuming axisymmetry. They placed a clylindrical plug of denser material just upstream of the reverse shock, and followed its subsequent evolution by solving the equations of ideal gas dynamics including equilibrium radiative cooling. Three levels of mesh refinement tracked the blob, providing local resolution equivalent to a uniform grid of 1536 x 160 grid cells. The benefit of AMR over a uniform grid was a cost savings of 350%. They found that the clump is strongly compressed in the intershock region by virtue of the high pressure and strong radiative cooling in the clump. While lacking the resolution of Klein et al. [9], they concluded the clump would most likely be disrupted into secondary fragments before reaching the dense outer shell. They suggested that x-ray flares would result if the largest of these secondary clumps survived to strike the dense shell.

Two other AMR-enabled simulations of interstellar shock waves illustrate the richness of the phenomena. In the first, Poludnenko, Frank & Blackman [55] simulated the propagation of a planar, adiabatic shock through a clumpy medium. The motivation for the simulations was to understand mass-loading and mixing of stellar and galactic outflows by inhomogeneities in the ambient medium. A parameter survey was carried out to assess the effects of clump mass and spatial distribution on the flow. 2D AMR simulations where performed in planar geometry using the AMRCLAW package of Berger & LeVeque [56], which combines SAMR with a second order-accurate Godunov scheme for ideal gas dynamics. The contribution of AMR was to achieve high resolution in each of many clumps scattered randomly throughout the volume as they are shocked and sheared. Resolution equivalent to a 800 x 1600 uniform grid was achieved in the clumps for a fraction of the cpu and memory cost, which is particularly important when conducting a large parameter survey. They found that a critical longitudinal and transverse separation between clumps exists such that for $d < d_{crit}$ and $L < L_{crit}$, the post-shock flow is strongly interacting, leading to enhanced turbulence and mixing.

In the second, a new type of instability was discovered in radiative shocks by Walder and Folini [57]. Radiative shock waves are found in many types of classical nebulae, like supernova remnants, planetary nebulae, Wolf-Rayet ring nebulae, etc. Typically, a shell of dense material is formed by the interaction of a fast outflow with the circumstellar medium (CSM) or interstellar medium (ISM). This shell will be bounded by two shocks: an outer, forward shock compressing the CSM/ISM, and an inner, reverse shock compressing the ejecta. Typically, the outer shock is radiative, while the inner shock is not because of the relative densities involved. The shocked media are separated by a contact discontinuity which is normally RT stable at late times because the shell decelerates.

However, it is well known that strongly radiative shocks suffer from an overstability such that the shock oscillates about its steady-state position in the rest frame of the contact discontinuity [58]. In multidimensions, Walder and Folini showed that different sections of the radiative shock oscillate independently, creating lateral pres-

sure disturbances within the high pressure shell. These chaotic perturbations sometimes accelerate the CD and thereby excite the RT instability. This leads to fingers and clumps of dense, shocked CSM/ISM efficiently mixing with the shocked ejecta. It is suggested by the authors that the mixing will boost the X-ray emission, contribute to rapid variability in the emission spectra, and may contribute to the the clumpy/filamentary appearance of the nebulae mentioned. The numerical simulations were carried out in 2D using AMRCART, which combines block SAMR with a second order accurate Godunov solver for gas dynamics. Optically thin radiative cooling was included assuming equilibrium ionization. The simulation was carried out in the rest frame of the CD, and hence the role of AMR was not shock tracking, but rather maintaining high resolution near the unstable interface.

2.4 Star Formation

The gravitational collapse of a gas cloud to form a star is a notoriously difficult problem because of the large range of length- and time-scales that need to be resolved. This problem has been solved in one dimension assuming spherical symmetry using moving adaptive meshes and implicit time integration [59, 60]. Important structures that need to be resolved span some 9 decades in radius from inside the hydrostatic protostellar core to the edge of the accreting cloud. Important timescales vary by 6 decades from the sound crossing time in the hydrostatic core to the accretion time. AMR holds forth the promise that simulations covering this range of scales will become possible in 3D, permitting a self-consistent investigation of disk accretion and the dynamical role of magnetic fields. While this is still out of reach, important strides have been made on the early stages of non-spherical cloud collapse by Richard Klein and colleagues [12, 13, 10, 11] . They are investigating the important problem of binary star formation using adaptive mesh techniques. They calculate the gravitational fragmentation of slowly rotating molecular cloud cores in 3D, assuming the gas is isothermal and non-magnetic. The equations of isothermal, self-gravitating gas dynamics is solved using a second order-accurate Godunov scheme on a block SAMR grid. The Poisson equation for the gravitational potential is solved using multigrid relaxation as described in [13].

An important early finding was that the cloud would fragment artificially due to numerical perturbations unless the local Jeans length is resolved by at least 4 cells at all times [12]. Since the Jeans length scales as $\rho^{-1/2}$, if collapse raises the central density by a factor x, then the grid spacing must be decreased by a factor $x^{-1/2}$ in order to avoid artificial fragmentation. The simulations of Truelove et al. [12, 13] used AMR was to follow compressions of $x \sim 10^8$ at which time the isothermal assumption breaks down. Local mesh refinement of up to a factor of 10^4 in resolution was accomplished by recursively adding grids refined by a factor of 4 wherever the Jeans condition $J = \Delta x/\lambda_J < 1/4$ was about to be violated. Simulations with fixed dynamic range X will inevitably violate the Jeans condition when $x > X$. Truelove et al. showed that an unperturbed cloud will fragment artificially shortly after the Jeans condition is violated [13]. They went on to show that a slowly rotating cloud with an m=2 nonaxisymmetric perturbation does not fragment into a binary system as had

previously been reported on the basis of fixed resolution simulations by Burkert & Bodenheimer [61], but rather forms a singular isothermal filament in accord with analytic predictions [62].

At sufficiently high number densities, however, the isothermal assumption breaks down because the cloud becomes optically thick to its own cooling radiation [63]. Boss et al. [64] simulated the non-isothermal evolution of the singular filament using both fixed as well as AMR grid codes. Opacity effects were modeled by adopting a barotropic equation of state with a variable gamma-law. The codes agreed providing the Jeans condition was obeyed in the fixed grid run. They found that the filament fragments into a binary system whose properties depend sensitively on the equation of state. Since the EOS only mocks up radiation trapping in an approximate way, the implication is that radiative transfer needs to be modeled self-consistently in future AMR simulations of cloud fragmentation. A flux-limited radiation diffusion algorithm for AMR grids has recently been introduced by Howell & Greenough [65] which is beginning to be applied to star formation [11].

2.5 Astrophysical Jets

Astrophysical jets are highly collimated, high speed bipolar outflows powered by disk accretion onto compact, gravitating objects. They manifest in a surprising diversity of systems and length scales, ranging from the pc-long optical jets from young stars [66] to the Mpc-long radio jets from active galactic nuclei and quasars [67]. Regardless of their origin, the jets themselves are interesting dynamically and morphologically because they sample both the deep potential wells where they were launched, and the interstellar/intergalactic medium they propagate through. Because of their high degree of collimation, the jets are believed to be hypersonic, and in the case of extragalactic jets, relativistic. A near universal feature of astrophysical jets, whether from stars or galaxies, is the occurrence of emission knots. Emission knots are patches of high emissivity arrayed along the length of the jet. These are generally interpreted as internal shock waves in the jet excited by internal or external perturbations [68].

Beginning in the early 1980s, there have been extensive numerical studies of the structure and dynamics of astrophysical jets with the framework of ideal gas dynamics and MHD. From the standpoint of simulations, the salient difference between protostellar jets and extragalactic jets is that the former are dense enough to be strongly radiative, whereas the latter are effectively adiabatic. One of the first applications of AMR in astrophysics was by Falle & Raga [7, 8], who studied the detailed structure of an emission knot in a radiative protostellar jet. 1D models by Raga et al. [69] showed that variations in the outflow velocity would create forward-reverse double shock pairs which propagate down the jet axis with the mean flow velocity, in good accord with observations. However, these models could not capture the bow shock appearance of the knots caused by the lateral expansion of the shocked gas into the ambient medium. Falle & Raga [8] simulated the multidimensional structure of a single knot in the rest frame of the knot. The contribution of AMR was to resolve the strong cooling region and ionization structure behind the radiative shocks,

which can be a small fraction of the jet radius. It is impractical to do this with uniform grids. The calculations were performed in 2D assuming axisymmetry using a block structured AMR grid. Six levels of grid refinement were used for an effective grid resolution of 1280×640 cells. The physics included gas dynamics, solved using the second-order Godunov scheme of Falle [70], non-equilibrium ionization, and radiative cooling. The simulations showed that the knots could survive as coherent entities for many jet radii, and were morphologically similar to those observed. The ability to resolve the ionization structure with AMR allowed them to make synthetic emission maps in the commonly detected [S II] doublet and thereby diagnose the physical conditions in observed jets.

Radio observations of radio-loud active galactic nuclei (AGN) mapped with VLBI techniques reveal one-sided jets with a stationary core and knots of emission that sometimes move superluminally [67]. This is conventially interpreted as the result of relativistic Doppler boosting and time dilation when observing a relativistic jet at small inclination angles [71]. With the advent of good algorithms for relativistic hydrodynamics [72, 73, 35], it becomes possible to model these sources. The structure, stability, and radio morphology of relativistic jets in compact extragalactic radio sources has been studied using AMR simulations by Hughes and collaborators [34, 74, 75]. Given the complexities of underlying hydro and the relativistic radiative transfer, they simply wanted to know whether such models resemble the data. They carried out 2D axisymmetric simulations using the code of Duncan and Hughes [72], which combines a second-order Godunov solver for the relativistic gas dynamics with the block-structured AMR method of Quirk [76]. The contribution of AMR to this work is to crisply capture the internal shocks without resort to large uniform grids and supercomputers. Assuming a simple relation between synchrotron emissivity and gas pressure, Mioduszuski et al. computed synthetic radio maps for a variety of inclination angles, Lorentz factors, and Mach numbers. They found that the VLBI maps of superluminal sources are reasonably well fit with strongly perturbed "pulsed" relativistic jets seen nearly end-on. They found that temporal changes of the models' radio appearance is not easily related to the underlying hydrodynamic quantities due to differential Doppler boosting.

2.6 Galaxies and Cosmology

Although AMR was invented for accurately integrating the hyperbolic partial differential equations of fluid dynamics, the adaptive mesh used in conjunction with the particle-mesh (PM) and P^3M N-body techniques [77] is extremely powerful for simulations of collisionless, self-gravitating systems of particles as arise in galactic dynamics and cosmological structure formation. AMR N-body codes have been developed by Couchman [27], Jessop et al. [78], Kravtsov et al. [20], Bryan and Norman [18, 19] and Knebe et al. [79]. These codes differ in AMR data structures and how AMR is used to optimize the N-body calculation.

Couchman adaptively introduces one or two levels of block mesh refinement around highly clustered regions of particles in order to reduce the number of particle-particle pairs in a P^3M calculation of dark matter clustering, resorting instead to the

faster PM calculation on the subgrids. The Poisson equation is solved on each level of the grid hierarchy using Fourier techniques with special Greens functions. The algorithm is referred to as AP^3M (Adaptive P^3M).

Jessop et al. combine the classic PM scheme with block structured local mesh refinement to achieve high force resolution in condensed systems. The Poisson equation is solved at all levels of the grid hierarchy using Neumann boundary conditions interpolated from parent grids and an ADI relaxation scheme. The algorithm is referred to as PM^2 (Particle Multiple Mesh).

Kravtsov et al. combine the octree cell refinement approach of Khokhlov [14] with PM to create the ART (Adaptive Refinement Tree) code. The Poisson equation is solved using successive multilevel relaxation.

Bryan & Norman adapted and generalized Couchman's AP^3M to an arbitrarily deep AMR grid hierarchy and married it to a PPM-derived hydro solver (see next section.) Two Poisson solvers are implemented: one based on Fourier techniques, and another using multigrid relaxation techniques (see O'Shea et al., these proceedings).

Knebe et al.'s code is similar to the ART code in that cell refinement and multigrid relaxation is used, but the underlying data structures and timestepping schemes are somewhat different.

Klypin et al. [21] have applied the ART code to the gravitational clustering of cold dark matter in cosmological simulations of structure formation. The key difficulty of all such calculations is to resolve the scales on which galaxies form (1-10 kpc) in cosmological volumes large enough to sample the longest perturbation waves or to get good statistics ($\sim$ 100 Mpc). The required spatial dynamic range is therefore $10^4 - 10^5$ in 3D for multiple centers of interest (galaxies). AMR is one option for achieving such resolution; meshless tree codes are another (cf. [80]). Klypin et al. investigated the long-standing overmerging problem, in which N-body simulations of the formation of clusters of galaxies yield too few galaxy-sized dark matter (DM) halos compared with observations [81, 80]. Rather, early simulations found that the galaxy DM halos merged with one another as they orbited within the cluster potential. Originally, it was thought that overmerging was a consequence of the omission of dissipative baryons from the models (e.g., [82]).

Klypin et al. showed that the overmerging problem is primarily a numerical resolution problem. Namely, that with inadequate force resolution, galaxy DM halos are numerically smeared out. As a consequence, the portion of the DM halo that is beyond its tidal radius gets stripped by cluster tidal forces as well as through close encouters with other galaxy halos. The DM halos essentially evaporate in a few orbits and their cores sink to the center of the cluster by dyamical friction. Klypin et al. combined analytic estimates and ART simulations to determine the resolution requirements to avoid overmerging. The simulations used 128^3 DM particles and a base grid of 256^3 cells in a volume 15 h^{-1} Mpc on a side. Up to 7 levels of 2x cell refinement were permitted, for a maximum dynamic range of 32,000 and spatial resolution of 0.5 h^{-1} kpc. They found that a force resolution of 1-2 h^{-1} kpc and a mass resolution of $\sim 2 \times 10^8 h^{-1} M_\odot$ is sufficient to sample the population of galaxy DM halos in a rich cluster of galaxies.

2.7 Hydrodynamic Cosmology

The marriage of a gravitational N-body code for the cold dark matter in the universe with a hydrodynamics code to model the baryonic component is referred to as a cosmological hydrodynamics code. A natural marriage is a PM N-body code with an Eulerian gas dynamics code, and a number of such codes have been developed [83, 84, 85, 86, 87, 88]. The spatial resolution of these codes is limited to the grid spacing, which limits the spatial dynamic range to 1000 or less on current high-end machines. This makes them useful for simulations of the diffuse intergalactic medium (e.g., [89]), but is far short of the 10^{4-5} dynamic range needed for galaxy large scale structure studies, as discussed above. AMR overcomes this limitation.

AMR hydrodynamic cosmology codes have been developed by Byran & Norman [18, 19], Kravtsov [90], and Teyssier [91]. The Bryan & Norman code Enzo combines a block-SAMR code for ideal gas dynamics using a version of the PPM algorithm adapted to cosmological flows [87], with a PM collisionless matter solver as described above (see also paper by O'Shea et al., these proceedings.) The code has been supplemented with the multispecies primordial gas chemistry model of Anninos et al. [92], photo-ionization heating by an evolving metagalactic UV background, and a parameterized model for star formation and feedback. The Kravtsov code builds upon the ART N-body code described above, and adds the second-order Godunov solver for ideal gas dynamics described in [14]. The RAMSES code [91] developed by Teyssier is similar to the Kravtsov code with minor differences in implementation.

A problem all three groups have attacked is the formation of an X-ray cluster of galaxies, treating the baryons as non-radiative. This is a good approximation since the 10^8 K gas characteristic of many X-ray clusters has a cooling time long compared to the Hubble time. A particular set of initial conditions known as the Santa Barbara cluster has served as a community test problem, and is described in Frenk et al. [93]. The chief difficulty is resolving the X-ray core radius ($\sim$ 100 kpc) in the forming cluster within a simulation volume 64 Mpc on a side. Most of the X-ray luminosity is contained within this region. If one wants to resolve the core radius with 10 cells, say, then a dynamic range of 6,400 is required. Bryan & Norman [18] achieved a dynamic range of 8,192 using a base grid of 128^3 cells and 6 levels of 2x refinement. These results were compiled with those of 11 other codes and presented in [93]. It was found that to compute the X-ray luminosity to within a factor of 2 accuracy, at least this resolution is required. Kravtsov, Klypin & Hoffman [94] simulated the Santa Barbara cluster with the ART code with the same resolution as Bryan & Norman, and found excellent agreement with their results. These simulations, as well as those of Teyssier [91], have shown that the distribution of thermal pressure in the cluster is a much more robust quantity. The significance of this is that Sunyaev-Zeldovich (SZ) effect in clusters of galaxies, which is proportional to the line-of-sight integral of the intracluster gas pressure, is robustly predicted. This makes AMR simulations a powerful tool for guiding upcoming observational surveys of high redshift clusters using the SZ effect (e.g., [95]).

A second fruitful application of AMR cosmological hydrodynamics concerns the formation of the first bound objects and stars in the universe. Within the CDM model of structure formation, dark matter begins clustering on small mass scales after the epoch of matter-radiation equality—about 30,000 years after the Big Bang. The characteristic mass scale for DM halos increases with time such that by redshifts of z=20-30, it becomes comparable to the Jeans mass in the expanding, adiabatically cooling, primordial gas. Abel, Bryan & Norman [22] have used AMR to simulate how baryons collect into the potential well of such a low mass DM halo and the ensuing cooling and contraction of the gas to form a primordial molecular cloud in the halo's center. The simulation used a 64^3 base grid and 12 levels of 2x refinement for a dynamic range of 2.6×10^5. In addition to dark matter, gravity and gas dynamics, the calculation solved a 9-species chemical reaction network to model the gas phase reactions which produce molecular hydrogen—the primary coolant in primordial gas. At this resolution, a primordial molecular cloud of size ~ 5 pc was well resolved in a simulation volume 128 kpc (comoving) on a side at z=19. At the end of the calculation, a single, gravitationally unstable cloud core of mass $\sim 100 M_\odot$ began collapsing. To follow the evolution of the collapsing core to higher densities, Abel, Bryan & Norman [23] used an additional 15 levels, for a dynamic range of 10^{10}. The mesh refinement was driven by the Jeans condition and an analogous condition based on the local cooling time. At this resolution, they were able to refute the prediction by Silk [96] that a chemo-thermal instability would fragment the collapsing core into low mass stars. At the end of the calculation, only a single, collapsing, fully molecular cloud core was found with a size comparable to the Solar System. With mean density of 10^{15} cm^{-3}, this core would trap its cooling radiation and become hydrostatic. Based on the accretion rate at the end of the simulation, it is predicted that the cloud envelope of would accrete in 10^4 years, forming a Population III star with a mass in the range $30 - 300 M_\odot$. A calculation with 34 levels of refinement (dynamic range 10^{12}) by Bryan, Abel & Norman [97] confirms this result.

Table 2. Downloadable AMR Software.

Code	Description	URL
AMRCLAW	SAMR infrastructure and hyperbolic solvers	[98]
AMRCART	SAMR application: 3D MHD	[99]
BEARCLAW	SAMR infrastructure and PDE solvers	[100]
CHOMBO	SAMR infrastructure and PDE solvers	[101]
Enzo	SAMR application: hydrodynamic cosmology	[102]
FLASH	SAMR application: reactive fluid dynamics	[103]
MLAPM	AMR application: cosmological N-body	[104]
NIRVANA	SAMR application: 2D and 3D MHD	[105]
PARAMESH	SAMR infrastructure	[106]
SAMRAI	SAMR infrastructure	[107]

3 AMR Software

Here I tabulate some AMR libraries and application codes that are available for download (Table 2). This list is incomplete, because of the lack of centralized information about such tools, as well as the rapid rate of development in the field.

Acknowledgements: MLN acknowledges partial support from the National Computational Science Alliance via NSF Cooperative Agreement ACI-9619019.

References

1. M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53:484–512, 1984.
2. P. R. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comp. Physics*, 54:115–173, 1984.
3. M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comp. Phys.*, 82:64–84, 1989.
4. D. De Zeeuw. Amr method for mhd. *SIAM J. Sci. Comp.*, 20:22, 1998.
5. C. P. T. Groth, D. L. de Zeeuw, T. I. Gombosi, and K. G. Powell. A Parallel Adaptive 3D MHD Scheme for Modeling Coronal and Solar Wind Plasma Flows. *Space Science Reviews*, 87:193–198, 1999.
6. C. P. T. Groth, D. L. de Zeeuw, T. I. Gombosi, and K. G. Powell. Global three-dimensional MHD simulation of a space weather event: CME formation, interplanetary propagation, and interaction with the magnetosphere. *J. Geophys. Res.*, 105:25053–25078, November 2000.
7. S. A. E. G. Falle and A. C. Raga. The structure of knots in variable stellar jets. I - Symmetric knots. *MNRAS*, 261:573–583, April 1993.
8. S. A. E. G. Falle and A. C. Raga. The structure of knots in variable stellar jets-II. Asymmetric knots. *MNRAS*, 272:785–799, February 1995.
9. R. I. Klein, C. F. McKee, and P. Colella. On the hydrodynamic interaction of shock waves with interstellar clouds. 1: Nonradiative shocks in small clouds. *ApJ*, 420:213–236, January 1994.
10. R. I. Klein, K. Truelove, and C. F. McKee. The Jeans Condition: A New Constraint on Spatial Resolution in Simulations of Self- Gravitational Hydrodynamics. In *ASP Conf. Ser. 123: Computational Astrophysics; 12th Kingston Meeting on Theoretical Astrophysics*, pages 152–+, 1997.
11. R. I. Klein, R. T. Fisher, M. R. Krumholz, and C. F. McKee. Recent Advances in the Collapse and Fragmentation of Turbulent Molecular Cloud Cores. In *Revista Mexicana de Astronomia y Astrofisica Conference Series*, pages 92–96, January 2003.
12. J. K. Truelove, R. I. Klein, C. F. McKee, J. H. Holliman, L. H. Howell, and J. A. Greenough. The Jeans Condition: A New Constraint on Spatial Resolution in Simulations of Isothermal Self-gravitational Hydrodynamics. *ApJLett*, 489:L179+, November 1997.
13. J. K. Truelove, R. I. Klein, C. F. McKee, J. H. Holliman, L. H. Howell, J. A. Greenough, and D. T. Woods. Self-gravitational Hydrodynamics with Three-dimensional Adaptive Mesh Refinement: Methodology and Applications to Molecular Cloud Collapse and Fragmentation. *ApJ*, 495:821–+, March 1998.
14. A. M. Khokhlov. Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *J. Comp. Phys.*, 143:519–543, 1998.

15. A. Kritsuk, T. Plewa, and E. Müller. Convective cores in galactic cooling flows. *MNRAS*, 326:11–22, September 2001.
16. S. A. E. G. Falle, R. F. Coker, J. M. Pittard, J. E. Dyson, and T. W. Hartquist. numerical simulations of tail formation in wind interactions with injected material. *MNRAS*, 329:670–674, January 2002.
17. A. C. Calder, B. Fryxell, T. Plewa, R. Rosner, L. J. Dursi, V. G. Weirs, T. Dupont, H. F. Robey, J. O. Kane, B. A. Remington, R. P. Drake, G. Dimonte, M. Zingale, F. X. Timmes, K. Olson, P. Ricker, P. MacNeice, and H. M. Tufo. On Validating an Astrophysical Simulation Code. *ApJSupp*, 143:201–229, November 2002.
18. G. L. Bryan and M. L. Norman. In D. A. Clarke and M. Fall, editors, *Computational Astrophysics; 12th Kingston Meeting on Theoretical Astrophysics, proceedings of meeting held in Halifax; Nova Scotia; Canada October 17-19; 1996*. ASP Conference Series # 123, 1997.
19. G. L. Bryan and M. L. Norman. A hybrid amr application for cosmology and astrophysics. In N. Chrisochoides, editor, *Workshop on Structured Adaptive Mesh Refinement Grid Methods*, page 165. IMA Volumes in Mathematics No. 117, 2000.
20. A. V. Kravtsov, A. A. Klypin, and A. M. Khokhlov. Adaptive Refinement Tree: A New High-Resolution N-Body Code for Cosmological Simulations. *ApJSupp*, 111:73–+, July 1997.
21. A. Klypin, S. Gottlöber, A. V. Kravtsov, and A. M. Khokhlov. Galaxies in N-Body Simulations: Overcoming the Overmerging Problem. *ApJ*, 516:530–551, May 1999.
22. T. Abel, G. L. Bryan, and M. L. Norman. The Formation and Fragmentation of Primordial Molecular Clouds. *ApJ*, 540:39–44, September 2000.
23. T. Abel, G. L. Bryan, and M. L. Norman. The Formation of the First Star in the Universe. *Science*, 295:93–98, January 2002.
24. C. Loken, M. L. Norman, E. Nelson, J. Burns, G. L. Bryan, and P. Motl. A Universal Temperature Profile for Galaxy Clusters. *ApJ*, 579:571–576, November 2002.
25. K. Tassis, T. Abel, G. L. Bryan, and M. L. Norman. Numerical Simulations of High-Redshift Star Formation in Dwarf Galaxies. *ApJ*, 587:13–24, April 2003.
26. D. Nagai, A. V. Kravtsov, and A. Kosowsky. Effect of Internal Flows on Sunyaev-Zeldovich Measurements of Cluster Peculiar Velocities. *ApJ*, 587:524–532, April 2003.
27. H. M. P. Couchman. Mesh-refined P3M - A fast adaptive N-body algorithm. *ApJL*, 368:L23–L26, February 1991.
28. G. L. Bryan. Fluids in the universe: Adaptive mesh in cosmology. *Computing in Science and Engineering*, 1:2:46, 1999.
29. K. Kifonidis, T. Plewa, H.-T. Janka, and E. Müller. Nucleosynthesis and Clump Formation in a Core-Collapse Supernova. *ApJLett*, 531:L123–L126, March 2000.
30. F. X. Timmes, M. Zingale, K. Olson, B. Fryxell, P. Ricker, A. C. Calder, L. J. Dursi, H. Tufo, P. MacNeice, J. W. Truran, and R. Rosner. On the Cellular Structure of Carbon Detonations. *ApJ*, 543:938–954, November 2000.
31. V. N. Gamezo, A. M. Khokhlov, E. S. Oran, A. Y. Chtchelkanova, and R. O. Rosenberg. Thermonuclear Supernovae: Simulations of the Deflagration Stage and Their Implications. *Science*, 299:77–81, January 2003.
32. R. Cid-Fernandes, T. Plewa, M. Rozyczka, J. Franco, R. Terlevich, G. Tenorio-Tagle, and W. Miller. On the evolution of ejecta fragments in compact supernova remnants. *MNRAS*, 283:419–430, November 1996.
33. D. Balsara. Adaptive Mesh Refinement in Computational Astrophysics – Methods and Applications. *Journal of Korean Astronomical Society*, 34:181–190, December 2001.

34. A. J. Mioduszewski, P. A. Hughes, and G. C. Duncan. Simulated VLBI Images from Relativistic Hydrodynamic Jet Models. *ApJ*, 476:649–+, February 1997.
35. S. S. Komissarov and S. A. E. G. Falle. Simulations of Superluminal Radio Sources. *MNRAS*, 288:833–848, July 1997.
36. K. Powell. *ICASE Report*, 94-24, 1994.
37. K. Powell et al. *AIAA Paper*, 95-1704-CP, 1995.
38. P. M. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *J. Comp. Phys.*, 43:357, 1981.
39. A. Harten, P. Lax, and B. Van Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM Rev.*, 25:1:35–61, 1983.
40. A. M. Khokhlov. Propagation of Turbulent Flames in Supernovae. *ApJ*, 449:695–+, August 1995.
41. M. Herant, W. Benz, W. R. Hix, C. L. Fryer, and S. A. Colgate. Inside the supernova: A powerful convective engine. *ApJ*, 435:339–361, November 1994.
42. M. Reinecke, W. Hillebrandt, and J. C. Niemeyer. Refined numerical models for multidimensional type Ia supernova simulations. *A&A*, 386:936–943, May 2002.
43. D. Arnett and E. Livne. The delayed-detonation model of a type IA supernovae. 1: The deflagration phase. *ApJ*, 427:315–329, May 1994.
44. P. Hoflich, A. M. Khokhlov, and J. C. Wheeler. Delayed detonation models for normal and subluminous type IA sueprnovae: Absolute brightness, light curves, and molecule formation. *ApJ*, 444:831–847, May 1995.
45. A. M. Khokhlov and E. Oran. *Combust. Flame*, 1999.
46. B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *ApJSupp*, 131:273–334, November 2000.
47. A. M. Khokhlov. Delayed detonation model for type IA supernovae. *A&A*, 245:114–128, May 1991.
48. M. Zingale, F. X. Timmes, B. Fryxell, D. Q. Lamb, K. Olson, A. C. Calder, L. J. Dursi, P. Ricker, R. Rosner, P. MacNeice, and H. M. Tufo. Helium Detonations on Neutron Stars. *ApJSupp*, 133:195–220, March 2001.
49. S. E. Woosley, A. Heger, and T. A. Weaver. The evolution and explosion of massive stars. *Reviews of Modern Physics*, 74:1015–1071, November 2002.
50. A. Burrows, J. Hayes, and B. A. Fryxell. On the Nature of Core-Collapse Supernova Explosions. *ApJ*, 450:830–+, September 1995.
51. A. Mezzacappa and TeraScale Supernova Initiative Collaboration. TeraScale Supernova Initiative. *Bulletin of the American Astronomical Society*, 34:687–+, May 2002.
52. K. Kifonidis, T. Plewa, H.-T. Janka, and E. Müller. non-spherical core collapse supernovae. I. Neutrino-driven convection, Rayleigh-Taylor instabilities, and the formation and propagation of metal clumps. *A&A*, 408:621–649, September 2003.
53. T. Plewa and E. Müller. The consistent multi-fluid advection method. *A&A*, 342:179–191, February 1999.
54. P. Colella and P. R. Woodward. The piecewise parabolic method (ppm) for gas-dynamical simulations. *J. Comp. Physics*, 54:174–201, 1984.
55. A. Y. Poludnenko, A. Frank, and E. G. Blackman. Hydrodynamic Interaction of Strong Shocks with Inhomogeneous Media. I. Adiabatic Case. *ApJ*, 576:832–848, September 2002.
56. M. J. Berger and R. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Num. Anal.*, 35:2298–2316, 1998.

57. R. Walder and D. Folini. Knots, filaments, and turbulence in radiative shocks. *A&A*, 330:L21–L24, February 1998.
58. R. A. Chevalier and J. N. Imamura. Linear analysis of an oscillatory instability of radiative shock waves. *ApJ*, 261:543–549, October 1982.
59. K.-H. A. Winkler and M. J. Newman. Formation of solar-type stars in spherical symmetry. I - The key role of the accretion shock. *ApJ*, 236:201–211, February 1980.
60. K.-H. A. Winkler and M. J. Newman. Formation of solar-type stars in spherical symmetry. II - Effects of detailed constitutive relations. *ApJ*, 238:311–325, May 1980.
61. A. Burkert and P. Bodenheimer. Fragmentation in a centrally condensed protostar. *MNRAS*, 280:1190–1200, June 1996.
62. S. Inutsuka and S. M. Miyama. Self-similar solutions and the stability of collapsing isothermal filaments. *ApJ*, 388:392–399, April 1992.
63. R. B. Larson. The physics of star formation. *Rep. Prog. Phys.*, 66:1651–97, 2003.
64. A. P. Boss, R. T. Fisher, R. I. Klein, and C. F. McKee. The Jeans Condition and Collapsing Molecular Cloud Cores: Filaments or Binaries? *ApJ*, 528:325–335, January 2000.
65. L. H. Howell and J. A. Greenough. Radiation diffusion for multi-fluid eulerian hydrodynamics with adaptive mesh refinement. *J. Comp. Phys.*, 184:53–78, 2003.
66. B. Reipurth and J. Bally. Herbig-Haro Flows: Probes of Early Stellar Evolution. *Ann. Rev. Astron. Astrophys.*, 39:403–455, 2001.
67. A. Ferrari. Modeling Extragalactic Jets. *Ann. Rev. Astron. Astrophys.*, 36:539–598, 1998.
68. M. J. Rees. The M87 jet - Internal shocks in a plasma beam. *MNRAS*, 184:61P–65P, September 1978.
69. A. C. Raga, L. Binette, J. Canto, and N. Calvet. Stellar jets with intrinsically variable sources. *ApJ*, 364:601–610, December 1990.
70. S. A. E. G. Falle. The Effect of Turbulence on the Largescale Structure of Radio Jets. *MNRAS*, 269:607–+, August 1994.
71. R. D. Blandford and A. Konigl. Relativistic jets as compact radio sources. *ApJ*, 232:34–48, August 1979.
72. G. C. Duncan and P. A. Hughes. Simulations of relativistic extragalactic jets. *ApJLett*, 436:L119+, December 1994.
73. J. M. A. Marti, E. Muller, J. A. Font, and J. M. Ibanez. Morphology and Dynamics of Highly Supersonic Relativistic Jets. *ApJLett*, 448:L105+, August 1995.
74. P. E. Hardee, A. Rosen, P. A. Hughes, and G. C. Duncan. Time-dependent Structure of Perturbed Relativistic Jets. *ApJ*, 500:599–+, June 1998.
75. A. Rosen, P. A. Hughes, G. C. Duncan, and P. E. Hardee. A Comparison of the Morphology and Stability of Relativistic and Nonrelativistic Jets. *ApJ*, 516:729–743, May 1999.
76. J. J. Quirk. Adaptive mesh refinement for shockwave simulations. *Ph.D. Thesis*, 1991.
77. R. Hockney and J. Eastwood. In *Computer Simulation Using Particles*. McGraw Hill, New York, 1988.
78. C. Jessop, M. Duncan, and W. Y. Chau. Multigrid methods for n-body gravitational systems. *J. Comp. Phys.*, 115:339–351, 1994.
79. A. Knebe, A. Green, and J. Binney. Multi-level adaptive particle mesh (MLAPM): a c code for cosmological simulations. *MNRAS*, 325:845–864, August 2001.
80. B. Moore, N. Katz, and G. Lake. On the Destruction and Overmerging of Dark Halos in Dissipationless N-Body Simulations. *ApJ*, 457:455–+, February 1996.
81. S. D. M. White. The dynamics of rich clusters of galaxies. *MNRAS*, 177:717–733, December 1976.

82. N. Katz, L. Hernquist, and D. H. Weinberg. Galaxies and gas in a cold dark matter universe. *ApJLett*, 399:L109–L112, November 1992.
83. R. Cen. A hydrodynamic approach to cosmology - Methodology. *ApJSupp*, 78:341–364, February 1992.
84. D. Ryu, J. P. Ostriker, H. Kang, and R. Cen. A cosmological hydrodynamic code based on the total variation diminishing scheme. *ApJ*, 414:1–19, September 1993.
85. W. Y. Anninos and M. L. Norman. Nonlinear hydrodynamics of cosmological sheets. 1: Numerical techniques and tests. *Astrophysical Journal*, 429:434–464, July 1994.
86. P. Anninos, M. L. Norman, and D. A. Clarke. Hierarchical numerical cosmology with hydrodynamics: Methods and code tests. *ApJ*, 436:11–22, November 1994.
87. G. L. Bryan, M. L. Norman, J. M. Stone, R. Cen, and J. P. Ostriker. A piecewise parabolic method for cosmological hydrodynamics. *Comp. Phys. Comm.*, 89:149–168, 1995.
88. P. M. Ricker, S. Dodelson, and D. Q. Lamb. COSMOS: A Hybrid N-Body/Hydrodynamics Code for Cosmological Problems. *ApJ*, 536:122–143, June 2000.
89. Y. Zhang, P. Anninos, and M. L. Norman. A Multispecies Model for Hydrogen and Helium Absorbers in Lyman-Alpha Forest Clouds. *ApJLett*, 453:L57+, November 1995.
90. A. V. Kravtsov. High-resolution simulations of structure formation in the universe. *Ph.D. Thesis*, 1999.
91. R. Teyssier. Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called RAMSES. *A&A*, 385:337–364, April 2002.
92. P. Anninos, Y. Zhang, T. Abel, and M. L. Norman. Cosmological hydrodynamics with multi-species chemistry and nonequilibrium ionization and cooling. *New Astronomy*, 2:209–224, August 1997.
93. C. S. Frenk, S. D. M. White, P. Bode, J. R. Bond, G. L. Bryan, R. Cen, H. M. P. Couchman, A. E. Evrard, N. Gnedin, A. Jenkins, A. M. Khokhlov, A. Klypin, J. F. Navarro, M. L. Norman, J. P. Ostriker, J. M. Owen, F. R. Pearce, U.-L. Pen, M. Steinmetz, P. A. Thomas, J. V. Villumsen, J. W. Wadsley, M. S. Warren, G. Xu, and G. Yepes. The Santa Barbara Cluster Comparison Project: A Comparison of Cosmological Hydrodynamics Solutions. *ApJ*, 525:554–582, November 1999.
94. A. V. Kravtsov, A. Klypin, and Y. Hoffman. Constrained Simulations of the Real Universe. II. Observational Signatures of Intergalactic Gas in the Local Supercluster Region. *ApJ*, 571:563–575, June 2002.
95. A. Refregier and R. Teyssier. Numerical and analytical predictions for the large-scale Sunyaev-Zel'dovich effect. *Phys. Rev. D*, 66:043002–+, August 2002.
96. J. Silk. The first stars. *MNRAS*, 205:705–718, November 1983.
97. G. L. Bryan, T. Abel, and M. L. Norman. In *Supercomputing 2001*, page http://www.sc2001.org. IEEE/ACM, 2001.
98. *www.amath.washington.edu/∼rjl/amrclaw/*.
99. *www.astro.phys.ethz.ch/staff/walder/private/codes/A-MAZE/AMRCART/AMRCART.html*.
100. *www.amath.unc.edu/Faculty/mitran/bearclaw.html*.
101. *seesar.lbl.gov/ANAG/chombo/index.html*.
102. *cosmos.ucsd.edu/enzo*.
103. *flash.uchicago.edu*.
104. *astronomy.swin.edu.au/staff/aknebe/MLAPM/*.
105. *www.aip.de/groups/MHD/projects/NIRVANA/nirvana.html*.
106. *ct.gsfc.nasa.gov/paramesh/Users_manual/amr.html*.
107. *www.llnl.gov/CASC/SAMRAI/*.

Recent Advances in the Collapse and Fragmentation of Turbulent Molecular Cloud Cores: The Formation of Low Mass Stars

Richard I. Klein[1,2], Robert T. Fisher[1], Christopher F. McKee[2,3], and Mark Krumholz[3]

University of California Lawrence Livermore National Laboratory, L-023, 5000 East Ave, Livermore, CA 94550
University of California at Berkeley Department of Astronomy, Berkeley CA 94720
rklein@astron.berkeley.edu
University of California at Berkeley, Department of Physics, Berkeley CA 94720

Summary. The formation of Giant Molecular Clouds (GMCs) sets the stage for the formation of protostellar systems by the gravitational collapse of dense regions within the GMC that fragment into smaller core components that in turn condense into stars. Developing a theory of low mass star formation remains one of the most elusive, and most important, goals of theoretical astrophysics. Inherent in the difficulty in attaining this goal is that the gravitational collapse depends critically upon initial conditions within the cores which only recently have been known with sufficient accuracy to permit a realistic theoretical attack on the problem. Observations of stars in the vicinity of the Sun show that binary systems are prevalent and appear to be a general outcome of the collapse and fragmentation process. Despite years of progress, theoretical studies have still not determined why binary stars occur with such frequency, or indeed, even what processes determine the transition from single stars to binaries and thence to multiple stellar systems.

One of the goals of this research is to understand the nature and physical properties of the formation of binary and multiple stellar systems with typical low mass stars 0.2 to 3 $M_{\odot}$. Basic questions concerning this process remain unanswered. What determines the fraction of an unstable cloud that will fragment into protostellar objects? What determines the pattern of stellar clustering into binaries and multiple systems? Even after fragmentation occurs, we have little understanding of the subsequent collapse. Consequently, it is unclear how the mass distribution of fragments maps onto eventual stellar masses, something we must understand to explain the stellar initial mass function (IMF).

We have developed a powerful numerical technology that will contribute to answering these questions. This technology consists of a parallel adaptive mesh refinement (AMR) self-gravitational radiation hydrodynamics code. Our 3-D AMR code dynamically and automatically inserts and removes patches of recursively finer mesh through computational space as dictated by the changing temporal and spatial resolution requirements of the simulation. This results in considerable computational efficiency over conventional codes when applied to problems involving gravitational collapse across many orders of magnitude in density at locations in the computational volume not determinable beforehand.

In this paper we present preliminary results for the investigation of the parameter space of marginally stable, turbulent molecular cloud cores as they evolve from larger scale turbulent clouds. We discuss our initial conditions for molecular cloud cores and how they relate to observations of cloud cores. We present preliminary results of the collapse and fragmentation of turbulent cores leading to the formation of binary stellar systems and a preliminary calculation of the collapse of a 100 $M_{\odot}$ spherical core leading to high mass star formation. Finally, we briefly describe new advances in our code where we have developed a general moving sink particle method with AMR and a newly developed unsplit Godunov MHD capability.

1 Introduction

Most stars exist in gravitationally bound binary and low-order multiple systems. Although several mechanisms have been put forth to account for binary star formation, fragmentation has emerged as the leading mechanism for the past decade ([bodetal2000]). This point of view has been strengthened by observations that have shown that the binary frequency among pre-main-sequence stars is comparable to or greater than that among nearby main-sequence stars ([duch99]). This suggests that most binary stars be formed during the protostellar collapse phase which points to fragmentation as the most probable formation mechanism.

Until very recently, the extreme variations in length scale inherent in the star formation process have made it difficult to perform accurate calculations of fragmentation and collapse, which are intrinsically three-dimensional in nature. In order to address the fundamental issues outlined above, we have developed a robust, parallel adaptive mesh refinement (AMR) self-gravitational hydrodynamics code, including the effects of multifluids, radiation transport, and self-gravity, and applied it to a number of models of low mass star formation.

Over the last few years, we have begun a program of research investigating the properties of marginally stable, turbulent molecular cloud cores. Using turbulent simulations, we have generated models with radii, masses, density contrasts, turbulent linewidths, projected aspect ratios, and projected velocity gradients consistent with observations of molecular cloud cores ([kleinetal2001], [burkbod2000], [kleinetal2003]). This work represents a significant improvement over the previous theoretical work on such cores, which typically assumed a uniform spherical core with an artificially imposed perturbation (i.e. 10% $m = 2$ or white-noise density perturbations), and rigid solid-body rotation. Crucially, the turbulent spectrum imposes a characteristic scale on the models, which is the scale at which the core linewidth becomes supersonic.

Answers to basic questions concerning the fragmentation and collapse process remain elusive:**(1)** Why do most stars form in binary systems? **(2)** What determines the efficiency of star formation? **(3)** What are the properties of protostellar disks formed in turbulent molecular cloud cores? Are they typically aligned or misaligned relative to one another? **(4)** What determines the distribution of fragment masses, which is related to the initial distribution of stellar masses, as well as the initial orbital angular momenta and rotational spins of the individual fragments?

Binary stars have a wide range of periods, ranging from less than a day to more than 10^6 yr ([bodetal2000]). The median period is 180 yr; for a total binary mass of 1 $M_\odot$, the corresponding separation is about 30 AU $= 4.5 \times 10^{14}$ cm. Since our study is aimed at understanding the formation of binary systems rather than the formation of individual stars, we shall not attempt to resolve structures less than about 10 AU in size.

It is crucial that the initial models which one uses faithfully describe those present in nature. High-resolution observations of molecular cloud cores (eg. [mott2001]) indicate that the mean column density of pre-stellar molecular cloud cores is close to that of a centrally condensed isothermal sphere supported primarily by thermal pressure. Moreover, observations of linewidths in star-forming regions indicate that the non-thermal linewidths are typically transonic on the scale of cores, and obey a power law linewidth-size relation ([lar81]). In the past year, we have generated self-consistent, initial conditions for cores in virial balance between gravity and thermal and turbulent pressures, which incorporate density fluctuations in addition to those present in the velocity field, and which match these key known observational properties of cores. (We describe the procedure by which we generate these models later in §2.2.) In contrast, nearly all calculations which appear in the literature deal with cores unrealistically initially far from equilibrium (e.g. [bossbod79], [boss91], [burkbod93]), without any turbulent support, leading to highly supersonic collapse velocities, and artificially symmetric collapses.

2 Computational Methodology

2.1 The Eulerian Code

Following the evolution of a collapsing molecular cloud as regions within it increase in density across many orders of magnitude is a formidable task. Conventional grid-based codes require that the finest-resolution gridding be applied over large volumes that may evolve to be devoid of fragments and thus not require the small zoning. Our 3-D adaptive code overcomes this problem.

First, the code employs a conservative higher-order Godunov scheme to solve the Euler equations of compressible gas dynamics using an optimized approximate Riemann solver ([toro97]). The algorithm is second-order accurate in both space and time for smooth flow problems, and it has a robust and accurate treatment of shocks and contact discontinuities. The code is capable of handling an arbitrary number of fluids.

The second major component of the code is a self-gravity solver. At each time step we solve a Poisson problem on the adaptive grid hierarchy to obtain the gravitational potential; we then apply the gradient of this potential as a source term in the momentum and energy equations ([true98], [klein99]). A multigrid iteration scheme is used to solve the linear system of equations resulting from the discretization of the Poisson equation on a level. These level solutions are then iterated to convergence to obtain a solution for the gravitational potential on all levels.

The third component is an adaptive, coupled radiation hydrodynamics solver using single-frequency flux-limited diffusion. The radiation transfer module uses a split method optimized for physical conditions where radiation-gas energy exchange by emission/absorption dominates the work done by the radiation field on the gas. First, the code solves a fully implicit system consisting of the emission/absorption and diffusion parts of the radiation and gas energy equations ([how2003]). It uses a Newton-Raphson iteration method, with an adaptive parallel multigrid method to find provisional solutions in each loop of the iteration. These solvers for hydrodynamics, self-gravity, and radiative transfer are coupled together within the adaptive mesh refinement infrastructure of our code. The adaptive mesh refinement scheme employs an automatic, dynamic regridding strategy based on an underlying rectangular discretization of the spatial domain ([bergol84], [bercol89]; [belletal94]). The overall algorithm conserves total energy, mass, and momentum because the time-step update applied at each grid at each level of refinement and the couplings at the interfaces between grids at different levels are all in conservation form.

A fourth component of our code is ideal MHD. We have developed a new fully 2nd order unsplit Godunov MHD ([crockettal2004]). It uses a conservative higher-order Godunov scheme similar to the hydrodynamics solver. We use a Poisson solve (using the same solver used by self-gravity) to ensure divergence-free fields. With our new methodology we have demonstrated much lower numerical dissipation than standard split scheme staggered mesh MHD codes currently in use. Our ideal MHD is currently being incorporated into our AMR framework and integrated with the rest of our code.

2.2 Sink Cell Development

Once fragmentation occurs in dense molecular clouds, each fragment begins to collapse to form stars. The computational timestep during collapse is determined by the free-fall timescale of the densest region, which is much shorter than the accretion timescale of the envelope because the free-fall time scale varies as $\rho^{-1/2}$. Therefore even with the power of the AMR code, following the evolution until most of the mass accretes is computationally demanding.

To handle this we have developed a sink particle technology for our code (Krumholz, McKee & Klein, poster this conference and [Krumetal2004]). We fix the maximum level of refinement at first protostellar core densities, and create a sink particle in any cell above that density that violates the Jeans condition ([true97]), $\Delta x > J\lambda_J \equiv J\left(\pi c_s^2/G\rho\right)^{1/2}$, where J is a constant of order unity, λ_J is the Jeans length in a cell of length Δx, and c_s and ρ are the sound speed and density in the cell. [true97] found that, to prevent artificial fragmentation, one generally cannot allow a cell to violate this criterion with $J = 0.25$. Sink particles can move through the computational grid, accreting gas and interacting gravitationally. We have tested our method against a number of problems with known solutions, inlcuding a collapsing isothermal sphere, Bondi accretion, Bondi-Hoyle accretion, and evolution of a Keplerian disk around a sink particle. Our method provides excellent agreement with

theoretical results, generally reproducing predicted accretion rates as well as density and velocity profiles to better than 5% accuracy.

2.3 Self-Consistent Initial Conditions

Truly accurate simulations for star formation must begin from realistic initial conditions, with self-consistent turbulence in both velocity and density. To generate such conditions, we start with a smooth Bonnor-Ebert sphere and perturb it on large scales; we adjust the energy injection rate of the perturbation to achieve the desired turbulent Mach number. This naturally produces the turbulent power spectrum $p(k) \propto k^{-4}$ predicted by theory and by Larson's law ([lar81]). After a few sound crossing times, we take the resulting object as our initial condition for a collapse calculation. We simulate observations of these cores to determine: the axis ratio ([myersetal91]); β, the ratio of rotational kinetic to gravitational potential energy ([goodetal93]); and γ, the exponent of the single-object linewidth-size relation ([ossen2002]). As shown in the table below, there is excellent agreement between our simulated cores and observations.

Property	Simulation	Observed
Axis Ratio	0.60	0.55 ± 0.04
β	0.023	~ 0.02
γ	0.49	0.5 ± 0.04

Our models are then evolved in time with fully coupled self-gravitational hydrodynamics with AMR using either a stiffened barotropic equation of state or alternatively using the fully coupled self gravitational radiation-hydrodynamics.

We have found that it sometimes necessary to allow for density as well as velocity perturbations. Here we again start with Bonnor Ebert spheres initially in virial equilibrium. We then generate velocity fields on density structures by generating power on large scales with a Gaussian random field and allowing density perturbations to develop. We then use smooth potentials to keep small regions from undergoing collapse until the cloud is well mixed. We allow the hydrodynamics to evolve the core and inject energy to maintain the turbulence while maintaining the Mach number. The initial simulation is run for a crossing time to check the spectrum of the velocity field; the aspect ratio; the rotational support and the linewidth size relation. it is only when all these aspects of the simulated core are in good agreement with core observations that we commence with a full scale collapse simulation.

In Figure 1 we show a Mach 1 simulated core typical of what we may find in the Taurus cloud region. The figure also shows what the observer would see after taking into account telescope beam smearing at a typical distance to Taurus. We note that the simulated core shows substantial structure, although this structure would not be readily observable. We have carefully examined the properties of our simulated cores and compared them with observations prior to evolving the cores under full self-gravitational collapse. Our simulated core has $\beta = 0.029, 0.001, 0.002$ in the x,

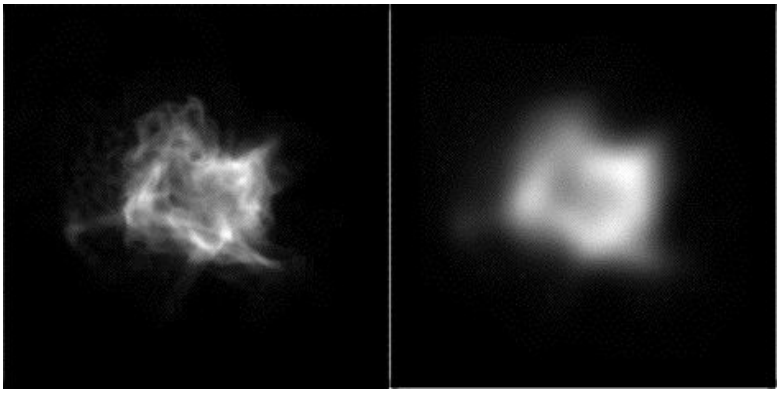


Fig. 1. The left panel shows a column density projection of a self- consistent (velocity and density perturbed) turbulent core correspsonding to velocity of Mach 1.0. The right panel shows a simulated 'observed' core obtained by beam smearing the simulated core with a 40 arc second beam at a distance of 140 parsecs, similar to the Taurus cloud. The range of column densities represented goes from 10^{-5} gm/cm^3 to 2 10^{-2} gm/cm^3.

y, and z projections with a mean value of 0.02 and a mode $\approx$ 0.0 in excellent agreement with the [goodetal93] sample. Likewise we find excellent agreement wih the axis ratio of [myersetal91]. A calculation of the power spectral density vs. wavenumber shows an average slope of -4.1 in excellent agreement with a k^{-4} spectrum. we have examined the linewidth-size relation of our simulated core and compared in detail with observations. First we filter the simulated core with a 40 arc sec beam. We then compute the velocity centroid in each pixel of the column density map of Figure 1. We compute the dispersion velocity of the velocity centorids within a Gaussian beam of varying width centered on the peak of the column density maps for each of the three projections. In Figure 2 we show the results of the velocity dispersion of the centroid vs. beam width at FWHM for the x, y and z projections of the simulated core. These slopes are compared with observed slopee of the sample from the [ossen2002] sample and the agreement appears to be excellent.

In this paper we discuss only barotropic calculations with initial models using velocity perturbations on turbulently perturbed Bonnor Ebert spheres ; our models with detailed radiation transfer and self-consistent turbulent cores with both velocity and density perturbations will be discussed elsewhere.

3 Simulations of Low Mass Star Formation from a Mach 3 Core

We have begun a detailed study of the collapse and fragmentation of isolated, low mass turbulent molecular cloud cores with detailed radiative transfer as well as models with barotropic equations of state. Beginning with a smooth density background distribution, we perturbed the velocity field of the core with a turbulent velocity spectrum. In a typical model with barotropic equation of state, we begin with an isolated

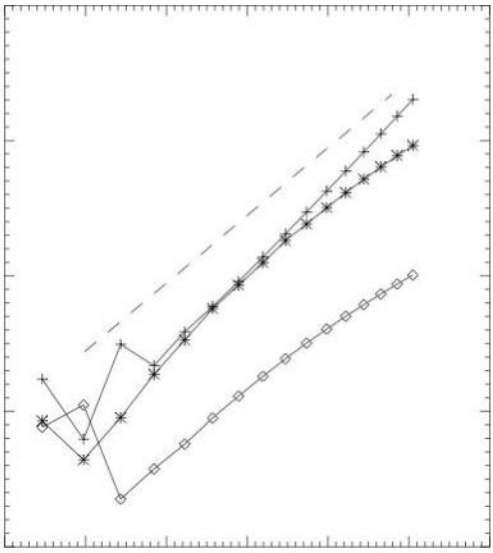


Fig. 2. The figure shows the velocity dispersion vs. log of beam width of the simulated core for 3 projections of the core. The line with crosses is the x projection, the line with asterisks is the y projection, and the line with diamonds is the z projection. The dashed line shows the slope of the linewidth size relation for observed cores based on the [ossen2002] sample.

11 $M_{\odot}$ core with a turbulent Mach number of $M = 3$, and formed a binary stellar system with an initial orbital separation of $\simeq 5000$ AU (see Figure 3). Using our sink particle methodology, we introduced sink particles at densities comparable to that of a first molecular protostellar core ($\sim 10^{-14}$ gm/cm^3). The stars begin unbound with respect to one another, and with a relatively high specific angular momentum $J/M \sim 5 \times 10^{20}$ cm^2/s, typical for prestellar molecular cloud cores, but high in comparison to known binaries. Additionally the system begins with a very high eccentricity; a generic outcome of filamentary collpase scenarios ([inut97]). At the point shown, some 164,000 yr after the beginning of the calculation, and 22,000 yr after the formation of the binary system the binary has accreted about 6% of the initital mass of the core. A key queston for this and other ongoing calculations, is whether the binary continued evolution will show that the binary will lower its J/M ratio to a value in closer agreement with observation via gravitational torques with the gas interior to the core, or whether large-scale tidal effects are effective in doing so.

In the left panel of Figure 3 we note the formation of the binary deep within the turbulent core. Our calculations with the full radiative transfer looks quite similar at this time since the gas surrounding the binary is just making the transition from optically thin to optically thick and radiative transfer effects are not yet prominent. Our future calculations will explore the full effects of radiative transfer on fragmentation and formation of binaries; for this paper we concentrate on barotropic models. The figure represents the log of the column density projection along the z axis in the 3D

simulation. Projections for the column density along the other ordinal directions are similar. The binary forms along filaments embedded within the highly non-spherical core. These filamentary structures are ubiquitous in all of our turbulent simulations. In the right hand panel we take a close look at the binary by telescoping into higher levels of AMR grid refinement. We note the outer region of the core shows evidence of the coarser grid structure with lower levels of refinement. The gas in these outer regions satisfies the Jean condition ([true97]) at lower densities and therefore lower refinement levels. The binaries however, are formed in much denser gas and the resolution seen here represents 9-10 levels of refinement with factors of 2 between refinement levels. Our AMR methodology allows us to take arbitrary refinement factors between levels. For these collapse calculations we have found that factor of 2 refinement between levels is an optimal choice between accuracy and computational efficiency. The binaries are surrounded by protostellar disks. The mass in the binaries is continuing to accrete from both the surrounding protostellar disks as well as being fed from the surrounding large scale filament. At about 176,000 yr the pair has formed an unequal mass binary system with each star accreting 0.4 and 0.2 $M\odot$ respectively. Calculations of the separation distance of the binary system show that they have are about 1000 AU apart at this time. We are continuing evolution of the system to determine the final masses and other properties of the binary system. We are also performing a parametric variation of the Mach number and turbulent realization to determine transition points that separate single star formation from binary formation and the transition from binary formation to multiple star formation.

4 Preliminary Simulation of High Mass Star Formation

We have begun studies of the formation of masssive stars from turbulent cores. A preliminary calculation is shown in Figure 5. Here we show the results of a simulation of a 100 $M_\odot$ spherical core with$\equiv$ 0.1 pc in radius. The core is initially in slow rotation with $\beta = 0.02$ and $\rho \propto r^{-2}$. We allow the core to collapse including the full effects of gravity, radiative transfer and radiation pressure. In this preliminary simulation the finest scale we resolve is 10 AU.

In Figure 5 we show the log density in a horizontal (edge on) and vertical(face on) slice). We note the formation of a disk surrounding the protostar. At this time in the simulation (4000 years), the massive star and disk is are still at a very early stage of development. The temperature has reached about 1200 K so that central dust has just begun to sublimate. The luminosity is only about 0.1 Eddington at this time ($L\approx 10^4 L - odot$).

5 Summary and Future Directions

We have begun detailed investigations of the formation of low mass and high mass stars from the collapse and fragmentation of turbulent molecular cloud cores using

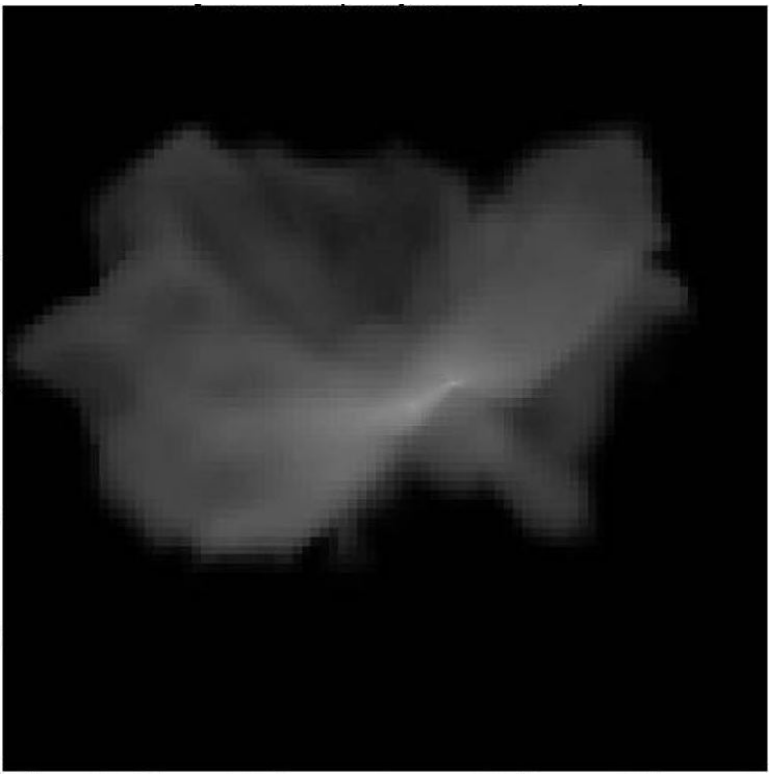


Fig. 3. This figure shows the log of the column density in z projection of a collapsed Mach 3 core at 176,000 yrs. The linear scale in each dimension is approximately 1/3 of a pc, and the peak density shown is 3 10^3 gm/cm^3. The formation of a binary system along a filament is evident. The next figure focuses in and shows the binary with its surrounding protostellar disks.

our high resolution self-gravitational radiation-hydrodynamics AMR code. We begin with realistic initial conditions for isolated cloud cores and demonstrate that by starting with a smooth Bonnor-Ebert sphere and perturbing it on large scales we can adjust the energy injection rate of the perturbation to achieve the desired turbulent Mach number. Our simulated cores reproduce the turbulent power spectrum predicted by theory and the linewidth-size relation of Larson's laws as well as showing excellent agreement with observed aspect ratio's of cores and rotation rates in excellent accord with observations.

We present preliminary calculations of the collapse of a Mach 3 turbulent core leading to the formation of a bound binary system. Our early results, including for the first time radiative transfer, show close agreement between the radiative and non-radiative results at times up until the collapse begins to transition from optically thin to optically thick flow. At these times, both members of the binary system appear to

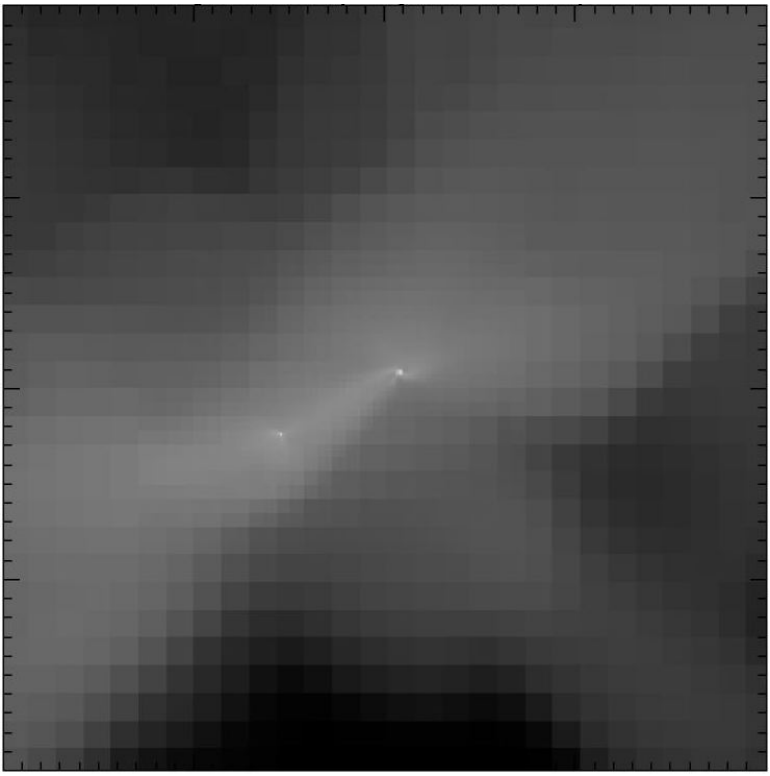


Fig. 4. This figure focuses in and shows the binary from the last figure with its surrounding protostellar disks.

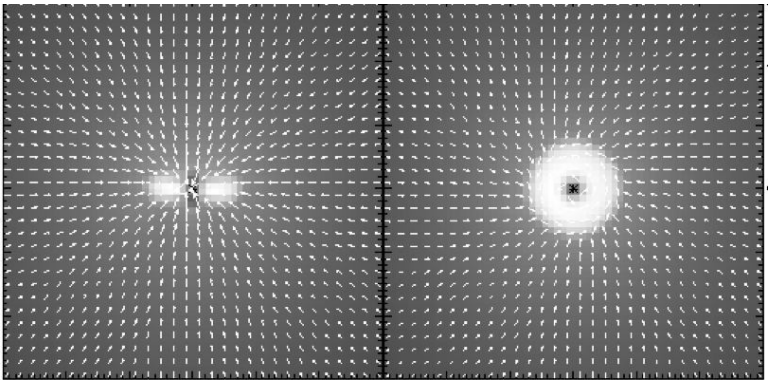


Fig. 5. This figures shows the early stages of the development of a high mass star. The first plot shows the log of the density of an edge on cut through the equatorial plane and the second panel is a face on view of the disk and star. The linear scales along each axis are 600 AU, and the range of densities represented is 10^6 cm^{-3} to 10^{11} cm^{-3}.

accrete gas from the surrounding filamentary structure they are part of as well as the surrounding protostellar disks. The binary appears to have a 2:1 mass ratio, but most of the mass accretion from the surrounding core has yet to have taken place. We have developed for the first time a new fully Lagrangian sink particle methodology with our full AMR framework to enable evolution of the collapsing fragmenting systems to long dynamical times. Several calculations are currently in progress to ascertain the role of turbulence in the low mass and high mass star formation process. Future work we are investigating will involve the role of MHD in determining the properties of low stars and high mass stars as they evolve from violent turbulent conditions in molecular clouds.

6 Acknowledgements

RIK and CFM acknowledge support from NASA ATP grant NAG 5-12042. CFM acknowledges support from NSF grant AST-0098365. Part of this work was supported under the auspices of the US Department of Energy at the Lawrence Livermore National Laboratory under contract W-7405-Eng-48. MRK acknowledges support from NASA GSRP grant NGT 2-52278. This research used computer resources awarded from grants at the NSF San Diego Supercomputer Center through an NPACI program grant UCB267 and the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

bodetal2000. Bodenheimer, P., Burkert, A., Klein, R. I., & Boss, A. P. 2000, in *Protostars and Planets IV*, ed. V. Mannings, A. P. Boss, & S. Russell (Tucson: University of Arizona Press), p. 675

duch99. Duchêne, G., Bouvier, J., & Simon, T. 1999, A&A, **343**, 831

kleinetal2001. Klein, R. I., Fisher, R. T., McKee, C. F., 2001, *IAU 200 International Conference on the Formation of Binary Stars*, eds. Zinnecker, H., & Matthieu

burkbod2000. Burkert, A. and Bodenhimer, P., 2000, ApJ, 543., 822

kleinetal2003. Klein, R. I., Fisher, R. T., Krumholz, M., and McKee, C. F. 2003, Revista Mexicana de Astronomia y Astrophysica, Conference series Wind, Bubbles and Explosions, eds. Arthur, J. & Henney, W. Vol. 15, 92

mott2001. Motte, F. & André, P. 2001, A&A, **365**, 440

lar81. Larson, R. 1981, MNRAS, **194**, 809

bossbod79. Boss, A., and Bodenheimer, P., 1979, ApJ, **234**, 289

boss91. Boss, A. P. 1991, Nature, 351, 298

burkbod93. Burkert, A. and Bodenheimer, P. 1993, MNRAS., 264, 798

toro97. Toro, E. 1997, *Riemann solvers and numerical methods for fluid dynamics : a practical introduction*, Berlin

true98. Truelove, J. K., Klein, R. I., McKee, C. F., Holliman, J. H., II, Howell, J. H., Greenough, J. A., & Woods, D. T. 1998, Ap.J., 495, 821

klein99. Klein, R. I. 1999, JCAM, **109**, 123
how2003. Howell, L. H. & Greenough, J. A. 2003, J. Comp. Phys. 184, 53
bergol84. Berger, M. J. & Oliger, J. 1984, J. Comp. Phys., 53, 484
bercol89. Berger, M. J. & Collela, P. 1989, J. Comp. Phys., 82, 64
belletal94. Bell, J., Berger, M., Saltzman, J., & Welcome, M. 1994, SIAM J. Sc. Comp., 15, 127
crockettal2004. Crockett, R., Collela. P., Mckee, C. F., & Klein, R. I. 2004, ApJ submitted
Krumetal2004. Krumholz, M., McKee, C. F., & Klein, R. I. 2004, ApJ submitted
true97. Truelove, J. K., Klein, R. I., McKee, C. F., Holliman, J. H., II, Howell, J. H., & Greenough, J. A. 1997, Ap.J., 489L, 179
myersetal91. Myers, P. C., Fuller, G. A. Goodman, A. A., &Benson, P. J. 1991, ApJ, 376, 561
goodetal93. Goodman, A. A., Benson, P. J., Fuller, G. A., &Myers, P. C., 1993, ApJ, 406, 528
ossen2002. Ossenkopf, V., &Mac Low, M.-M. 2002, A&A 300, 307
inut97. Inutsuka, S.. & Miyama, S. M. 1997, ApJ. 480, 681

3D AMR Simulations of Point-Symmetric Nebulae

Erik-Jan Rijkhorst, Vincent Icke, and Garrelt Mellema

Sterrewacht Leiden, P.O. Box 9513, 2300 RA, Leiden, The Netherlands
rijkhorst@strw.leidenuniv.nl

Summary. At the end of their lives low mass stars such as our Sun lose most of their mass. The resulting planetary nebulae show a wide variety of shapes, from spherical to highly bipolar. According to the generalized interacting stellar winds model, these shapes are due to an interaction between a very fast tenuous outflow, and a denser environment left over from an earlier slow phase of mass loss. Previous analytical and numerical work shows that this mechanism can explain cylindrically symmetric nebulae very well. However, many circumstellar nebulae have a multipolar or point-symmetric shape. With two-dimensional calculations, Icke showed that these seemingly enigmatic forms can be easily reproduced by a two-wind model in which the confining disk is warped, as is expected to occur in irradiated disks. Here, we present the extension to fully three-dimensional adaptive mesh refinement simulations of such an interaction.

1 Introduction

In the final phases of stellar evolution, low mass stars, such as our Sun, first swell up and shed a dense, cool wind in the asymptotic giant branch (AGB) phase. This episode is followed by a fast, tenuous wind that is driven by the exposed stellar core, the future white dwarf. The planetary nebulae (PNe) resulting from this expulsion phase come in a wide variety of shapes, from spherical to highly bipolar. Some even have a multipolar or point-symmetric shape. Balick ([Bal87]) proposed that such forms arise due to an interaction between a slow disk-shaped inner AGB nebula and the fast 'last gasp' of the star. Analytical ([Ick88]; [IPB89]) and numerical ([SL89]; [Ick91]; [MEI91]) work shows that this Generalized Interacting Stellar Winds (GISW) mechanism works very well (for an up-to-date review, see [BF02]). Several scenarios for obtaining a disk around a PN exist, and it is in general assumed in the models that the shape of the dense gas around the star is a disk or a toroid.

Icke ([Ick03]) proposed that the point-symmetric shapes observed in a number of PNe are formed in an interaction between a spherical stellar wind and a *warped* disk. It is prossible to produce such a warp around a single star, through the combined effects of irradiation and cooling (e.g. [Pri96]; [MBP96]). Whereas Icke's computations were restricted to a two-dimensional proof-of-principle, we now present a first

series of fully three-dimensional hydrodynamic computations of such a wind-disk interaction using the technique of Adaptive Mesh Refinement (AMR).

2 Point-symmetric nebulae

Work on cylindrically symmetric nebulae showed ([Ick88]; [Ick91]; [IMB92]) that sharply collimated bipolar flows are a frequent and natural by-product of the GISW model. However, many circumstellar nebulae have a multipolar or point-symmetric (i.e. antisymmetric) shape ([Sch93]; [ST98]). The nebulae that are formed in the wind-disk interaction would naturally acquire the observed antisymmetry if the disk that confines the fast wind is warped, instead of symmetric under reflection about the equatorial plane. Several mechanisms have been proposed for warping a disk surrounding a single star. The most interesting one for our purposes invokes radiative instability ([Pet77]; [IP90]; [Pri96]; [MBP96]).

Livio & Pringle already proposed ([LP96]; [LP97]) that the precession of warped disks might be responsible for point-symmetric nebulae. They proved conclusively that the various physical scales for mass, accretion, luminosity and precession match the observations. The production of the nebulae proper they attributed to an unspecified 'jet' mechanism. While leaving open the possibility that jets may be responsible for additional structures, we show that the interaction between a warped disk and a spherically symmetric wind suffices. The lobes of point-symmetric nebulae ([Sch93]; [ST98]) look as if they were produced almost simultaneously. This is difficult in the case of a precessing jet, which would make a corkscrew-like nebula of a type not readily apparent in post-AGB shells, although some objects do show features that are likely to be due to precession ([Sch93]).

3 Radiation driven warping

When an accretion disk is subject to external torques it may become unstable to warping ([BP75]; [Pet77]; [PP83]) and when irradiated by a sufficiently luminous central star even an initially flat disk will warp ([IP90]; [Pri96]; [MBP96]; [MBN98]). The difference in radiation pressure on slightly tilted annuli at different radii will induce the warp. Essential is that the disk is optically thick for the stellar radiation and for its own cooling flux. This restricts the disks to a specific subclass of high density and low temperature.

Analytical considerations lead to expressions for growth and precession rates and morphologies of the warp whereas numerical calculations including the effects of self-shadowing show that the non-linear evolution of the warp can produce highly distorted shapes, even with an inverted, counter-rotating inner disk region ([Pri97]). Applications of warped disk theory range from active galactic nuclei ([Pri97]; [MBN98]) to X-ray binaries ([MB97]), protostellar disks ([AP97]), and PNe ([LP96]; [LP97]).

Since we intend to study warped disks in PNe, we need a mechanism to form accretion disks in these systems. Plausible scenarios include the coalescence of compact binaries ([SL94]), absorption of planetary systems, or the formation of a disk due to a main-sequence companion being out of equilibrium when emerging from a common envelope (CE) phase with a primary AGB star. In this case, the companion loses most of the mass it accreted during the CE phase which subsequently forms a disk around the primary ([SL94]) and which, in a later stage, can get radiatively warped when illuminated by the central star of the PN.

For a PN the luminosity of the central star alone is sufficiently high to induce a radiation driven warp. Following [Pri97], an expression for the radius R_{crit} beyond which the disk is unstable to radiation driven warping is found from comparing the timescales of the viscous and radiation torques, leading to

$$R_{crit} = (2\pi/A)^2 \,, \tag{1}$$

with the contant A defined by $A^2 \equiv 1/4\, c^{-2} G^{-1} M_*^{-1} L_*^2 \eta^{-2} \dot{M}_{acc}^{-2}$. Here c is the speed of light, G the gravitational constant, $\eta \equiv \nu_2/\nu_1$ is the ratio of the azimuthal to the radial viscosity, M_* is the mass and L_* the luminosity of the central star and $\dot{M}_{acc}$ is the disk's accretion rate. We assumed a surface density $\Sigma_d = \dot{M}_{acc}/(3\pi\nu_1)$ (e.g. [Pri81]).

In a Cartesian coordinate system, the warped disk surface is given by ([Pri96])

$$x(R,\phi) = R \begin{pmatrix} \cos\phi\sin\gamma + \sin\phi\cos\gamma\cos\beta \\ -\cos\phi\cos\gamma + \sin\phi\sin\gamma\cos\beta \\ -\sin\phi\sin\beta \end{pmatrix} , \tag{2}$$

with local disk tilt angle $\beta(R,\phi)$, and orientation angle of the line of nodes $\gamma(R,\phi)$. Here, R and ϕ are the non-orthogonal radial and azimuthal coordinates respectively, pointing to the surface of the disk (cf. [Pri96]). In our model calculations we adopt the case of a steady precessing disk with no growth and zero torque at the origin for which we have in the precessing frame that $\gamma = A\sqrt{R}$ and $\beta = \sin\gamma/\gamma$, with the constant A defined by Eq. (1) ([MBP96]).

4 Time scales

Since radiative cooling plays such an important role in our model (see the next section), we need to compare the cooling time scale to three other time scales related to the disk: the precession and growth time scales of the warp, and the shock passing time. The latter is defined as the time the expanding wind blown bubble takes to travel to the outer disk radius R_d.

Adopting a cooling function of the form $\Lambda = \Lambda_1 T_s^{-1/2}$ and using the jump conditions for a strong shock, the cooling time of the shocked gas is given by (cf. [Kah76]; [KM92]) $t_c = C v_s^3/\rho_e$, with v_s the shock speed, and ρ_e the pre-shock environment density. When assuming a fully ionized gas of cosmic abundances, the constant C has a value of $6.0 \times 10^{-35}\,\mathrm{g\,cm^{-6}\,s^4}$.

Analytical relations for the radius $R_s(t)$ and speed $v_s(t)$ of the outer shock as functions of time can be derived (e.g. [LC99]) and with these the shock passing time readily follows from setting $R_s(t_{sp}) = R_d$ as

$$t_{sp} = (2/3\pi\rho_e)^{1/2} \dot{M}_w^{-1/2} v_w^{-1/2} R_d^2 \,. \tag{3}$$

The precession time scale of the disk is given by ([MBP96])

$$t_p = 48\pi^2 c G^{1/2} M_*^{1/2} L_*^{-1} R_d^{3/2} \Sigma_d \,, \tag{4}$$

where we assumed Keplerian rotation. The time scale for the initial growth of the warp is of the same order. When we use Eq. (1) for the critical radius as a typical disk radius, we find that the different time scales scale as

$$t_p \propto M_*^2 L_*^{-4} \dot{M}_{acc}^3 \eta^3 \Sigma_d \tag{5}$$

$$t_{sp} \propto \rho_e^{1/2} \dot{M}_w^{-1/2} v_w^{-1/2} M_*^2 L_*^{-4} \dot{M}_{acc}^4 \eta^4 \tag{6}$$

$$t_c \propto \rho_e^{-5/2} \dot{M}_w^{3/2} v_w^{3/2} M_*^{-3} L_*^6 \dot{M}_{acc}^{-6} \eta^{-6} \tag{7}$$

We are quite limited in our choice of M_*, L_*, v_w, and $\dot{M}_w$ since values for these parameters are strongly constrained by stellar evolution and wind models (e.g. [PPK88]; [Blo95]) but since the dependence of t_p, t_{sp}, and t_c on $\dot{M}_{acc}$ is so strong, a proper choice of this latter parameter leads to the desired proportion between the different time scales.

For our calculations we used $\dot{M}_w = 10^{-8} M_\odot \,\mathrm{yr}^{-1}$, $v_w = 200\,\mathrm{km\,s}^{-1}$, $M_* = 0.6 M_\odot$, $L_* = 10^4 L_\odot$, $\rho_e = 10^{-15}\mathrm{g\,cm}^{-3}$, $\dot{M}_{acc} = 10^{-7} M_\odot \,\mathrm{yr}^{-1}$, $\Sigma_d = 1\mathrm{g\,cm}^{-2}$, and $\eta = 1$ resulting in $R_{crit} \simeq 2\mathrm{AU}$, $t_p \simeq 17\mathrm{yr}$, $t_{sp} \simeq 0.4\mathrm{yr}$, $t_c \simeq 10^{-8}\mathrm{yr}$, and density contrast $\chi \equiv \rho_d/\rho_e \simeq 300$ with ρ_d the disk density. So $t_c \ll t_{sp} \ll t_p$ showing that cooling will indeed be of importance and that we can safely ignore the disk's precession.

5 Mechanism

The mechanism behind the formation of the multipolar lobes seen in the simulations is as follows (see also [Ick03]). As the central wind impinges on the inner rim of the disk, a three-dimensional bow shock develops around it. One branch of this shock flies off into space creating a lobe jutting out from the nebula, whereas the other slams into the concave side of the disk, scooping up disk material and thereby producing a set of smaller, unstable lobes (see Fig. 1). The opening angle of the shock depends inversely on the Mach number of the wind. Due to the cooling of the gas, the swept up shell is highly compressed and therefore thin, and the ram pressure from the wind directly drives the shell outwards, which are necessary ingredients for the bow shock to produce the lobes. When the density of the disk is not too high, the wind breaks through the concave part of the disk, producing another pair of lobes.

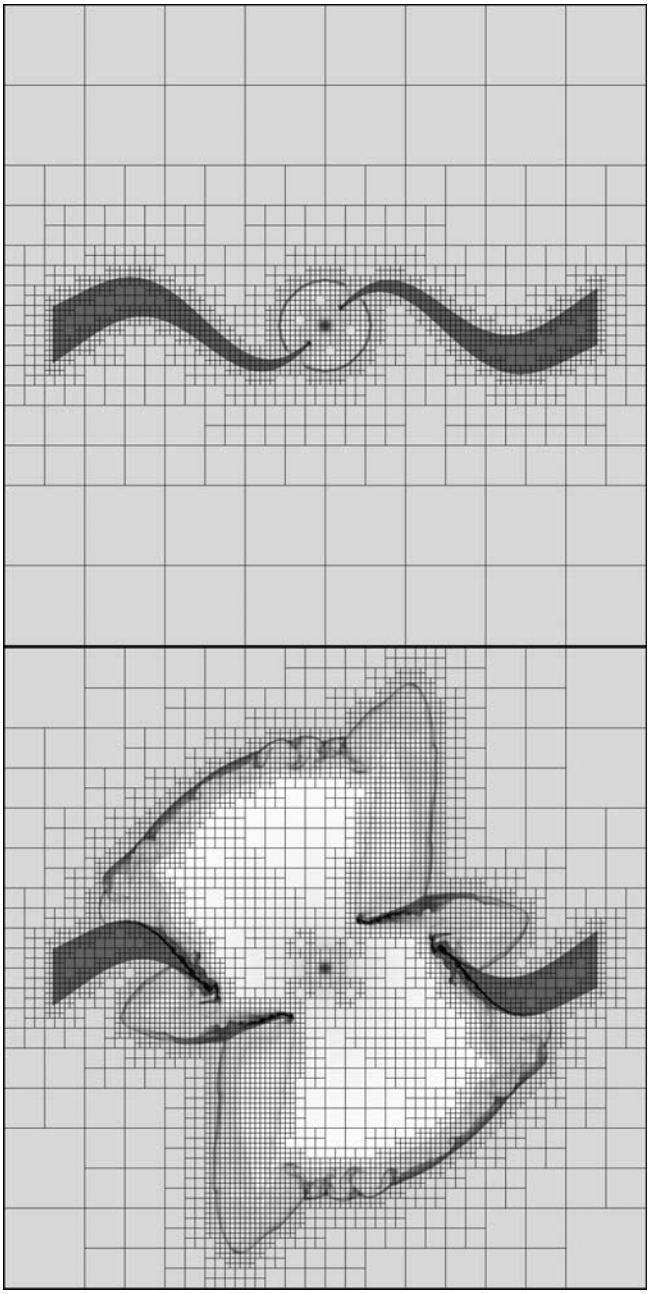


Fig. 1. 2D example of a wind-disk interaction showing the evolving AMR grid structure superimposed on a plot of the logarithm of the density. Every square represents a grid of 8x8 cells. Five different levels of refinement are visible. The effective resolution of this simulation is 1024x1024 cells.

6 Numerical implementation

We used the three-dimensional hydrocode *Flash* ([FOR00]) to model the interaction between a spherical wind and a warped disk. This parallelized code implements block-structured AMR ([BO84]; [BC89]) and a PPM type hydrosolver ([WC84]; [CW84]).

Besides implementing the proper initial and boundary conditions, we also added our own cooling module in order to model radiative cooling using a cooling curve ([DM72]; [MKR02]). This curve gives the energy loss rate as a function of temperature for a low density gas in collisional ionization equilibrium. The radiative losses were implemented through operator splitting and if the hydro timestep was larger than the cooling time, the former was subdivided into smaller steps when calculating the cooling.

To construct the warped disk, Eq. (1) was combined with a constant 'wedge angle' θ_d and a proper value for A, i.e. R_d was taken to be a few times R_{crit}, see Eq. (2). This disk was given a constant density which, through the density contrast χ, resulted in a value for the environment density n_e. The spherical wind was implemented as an inner boundary condition and given a $1/r^2$ density profile and a constant wind velocity v_w. The pressure was calculated from an equation of state with a constant adiabatic index γ.

7 2D wind-disk simulations

To check the implementation of the wind-disk interaction model into the AMR code described above, we repeated a number of the two-dimensional calculations previously done by [Ick03] (see Fig. 1). Because he used a different hydrosolver (FCT/LCD) and the outcome of the calculations strongly depend on turbulent processes in the gas, the simulations did not agree in every single detail but the overall point-symmetric morphologies were retrieved. All these simulations were run using a small value for the adiabatic index ($\gamma = 1.1$) resulting in 'momentum driven' bubbles.

To see what happens when more realistic cooling is applied, we ran some simulations with the cooling curve module and an adiabatic index $\gamma = 5/3$. This showed that, apart from the production of the by now familiar point-symmetric lobes, the outer shell of swept up gas is thinner and unstable and developes into a number of smaller lobes merging with one another as the shell expanded.

8 3D AMR Simulations

Following the two-dimensional trial calculations, we ran wind-disk simulations in three dimensions on a Cartesian grid with an effective resolution of up to 512^3 cells using five levels of refinement. Since we found in our two-dimensional calculations that simulations with cooling applied through a cooling curve did not result in a

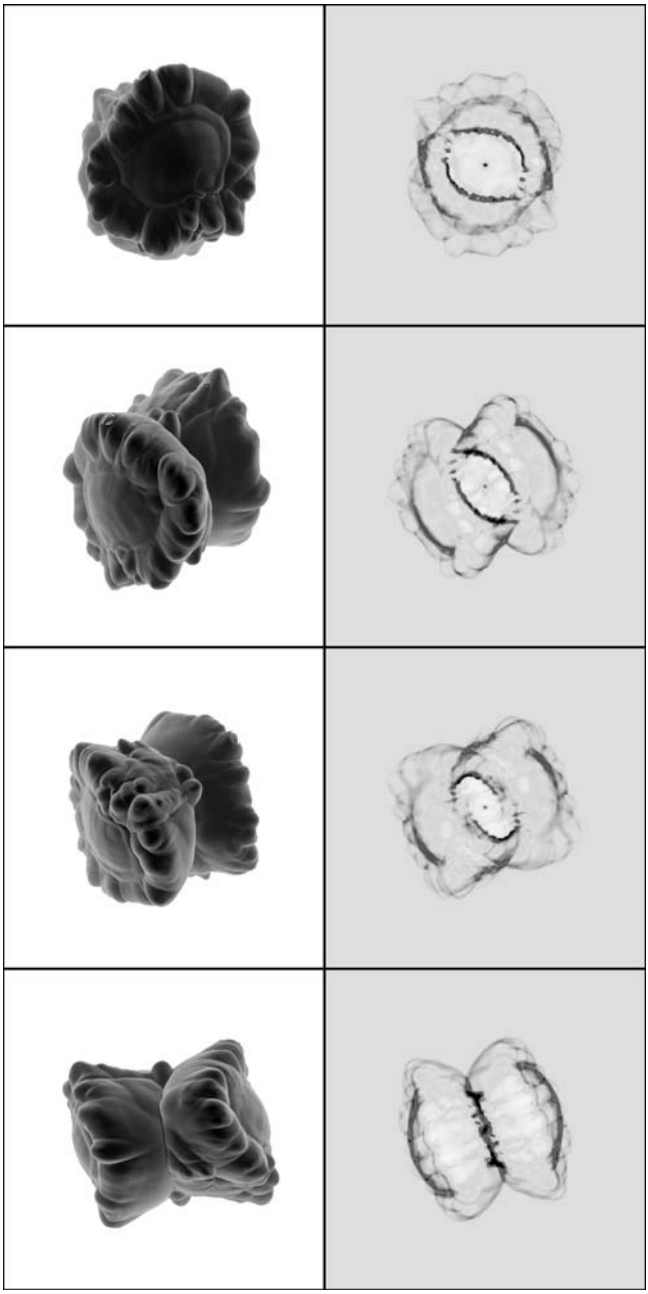


Fig. 2. Isosurfaces of the density at the end of the wind-disk simulation as seen from different angles (*left column*) and the corresponding synthesized Hα images (*right column*).

qualitatively different morphological outcome compared to calculations with a low value for the adiabatic index ($\gamma = 1.1$), we opted not to use the cooling curve module for our three-dimensional simulations to save computational time.

We used the following parameters: $\gamma = 1.1$, $n_e = 5 \times 10^8\,\mathrm{cm}^{-3}$, $\chi = 100$, $\theta_d = 5^o$, and $v_w = 200\,\mathrm{km\,s}^{-1}$. In Fig. 2 we present a visualization of the three-dimensional shape of the swept up shell through isosurfaces at different viewing angles. Also shown are the corresponding synthesized Hα images, derived by projecting the three-dimensional data cube onto the plane of the sky. For this, we simply integrated the density squared along the line of sight and used this as a rough estimate for the emission.

9 Conclusions

Our computations show that the wind-disk interaction model in which the confining disk is warped results in a wide variety of point-symmetric shapes. Nebulae that show 'punched holes', such as NGC 7027 ([CHM02]), are readily accommodated in this model. Other candidates for our model are the quadrupolar PNe K3-24 and NGC 7026, in which the inner disk is still visible, while the outer nebula shows clear point-symmetric structures. Also, the shapes of the proto-PNe M1-26 and M 4-18 (and maybe He 2-47) can be explained with our model. As a further application, large-scale explosions in non-planar disks, such as might occur in active galaxies, are expected to show similar patterns, provided that the disk material can cool rapidly enough. Movies of these simulations can be found at `http://www.strw.leidenuniv.nl/AstroHydro3D/`.

Acknowledgements V.I. expresses his gratitude to Raghvendra Sahai and Hugo Schwarz for lively discussions that were the primary cause for taking up this subject.

The research of G.M. has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences.

The software used in this work was in part developed by the DOE-supported ASCI/Alliance Center for Astrophysical Thermonuclear Flashes at the University of Chicago.

Our work was sponsored by the National Computing Foundation (NCF) for the use of supercomputer facilities, with financial support from the Netherlands Organization for Scientific Research (NWO), under grant number 614.021.016.

References

AP97. Armitage, P.J. and Pringle, J.E.: Radiation-induced Warping of Protostellar Accretions Disks. ApJL, **488**, L47+ (1997)

Bal87. Balick, B.: The evolution of planetary nebulae. I - Structures, ionizations, and morphological sequences. AJ, **94**, 671-678 (1987)

BF02. Balick, B. and Frank, A.: Shapes and Shaping of Planetary Nebulae. ARAA, **40**, 439-486 (2002)
BP75. Bardeen, J.M. and Petterson, J.A.: The Lense-Thirring Effect and Accretion Disks around Kerr Black Holes. ApJL, **195**, L65+ (1975)
BO84. Berger, M. and Oliger, J.: Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations J. Comp. Phys., **53**, 484 (1984)
BC89. Berger, M. and Colella, P.: Local Adaptive Mesh Refinement for Shock Hydrodynamics J. Comp. Phys., **82**, 64 (1989)
Blo95. Blöcker, T.: Stellar evolution of low- and intermediate-mass stars. II. Post-AGB evolution. A&A, **299**, 755 (1995)
CW84. Colella, P. and Woodward, P.: The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations J. Comp. Phys., **54**, 174 (1984)
CHM02. Cox, P., Huggins, P.J., Maillard, J.-P., Habart, E., Morisset, C., Bachiller, R. and Forveille, T.: High resolution near-infrared spectro-imaging of NGC 7027. A&A, **384**, 603-619 (2002)
DM72. Dalgarno, A. and McCray, R.A.: Heating and Ionization of HI Regions. ARAA, **10**, 375 (1972)
FOR00. Fryxell, B., Olson, K., Ricker, P., Timmes, F.X., Zingale, M., Lamb, D.Q., MacNeice, P., Rosner, R., Truran, J.W. and Tufo, H.: FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. ApJS, **131**, 273-334 (2000)
Ick88. Icke, V.: Blowing bubbles. A&A, **202**, 177-188 (1988)
IPB89. Icke, V., Preston, H.L. and Balick, B.: The evolution of planetary nebulae. III - Position-velocity images of butterfly-type nebulae. AJ, **97**, 462-475 (1989)
Ick91. Icke, V.: The hydrodynamics of aspherical planetary nebulae. I - Analytic evaluation of hydrodynamic difference schemes. A&A, **251**, 369-381 (1991)
IMB92. Icke, V., Mellema, G., Balick, B., Eulderink, F. and Frank, A.: Collimation of astrophysical jets by inertial confinement. Nature, **355**, 524-526 (1992)
Ick03. Icke, V.: Blowing up warped disks. A&A, **405**, L11-L14 (2003)
IP90. Iping, R.C. and Petterson, J.A.: Quasi-periodic precession of disks in X-ray binaries, with possible application to Centaurus X-3. A&A, **239**, 221-226 (1990)
Kah76. Kahn, F.D.: The temperature in very old supernova remnants. A&A, **50**, 145-148 (1976)
KM92. Koo, B. and McKee, C.F.: Dynamics of Wind Bubbles and Superbubbles. II. Analytic Theory. ApJ, **388**, 103 (1992)
LC99. Lamers, H.J.G.L.M. and Cassinelli, J.P.: Introduction to stellar winds. Cambridge, UK: Cambridge University Press (1999)
LP96. Livio, M. and Pringle, J.E.: The Formation of Point-symmetric Nebulae. ApJL, **465**, L55 (1996)
LP97. Livio, M. and Pringle, J.E.: Wobbling Accretion Disks, Jets, and Point-symmetric Nebulae. ApJ, **486**, 835 (1997)
MBP96. Maloney, P.R., Begelman, M.C. and Pringle, J.E.: Radiation-driven Warping: The Origin of Warps and Precession in Accretion Disks. ApJ, **472**, 582 (1996)
MB97. Maloney, P.R. and Begelman, M.C.: The Origin of Warped, Precessing Accretions Disks in X-Ray Binaries. ApJL, **491**, L43 (1997)
MBN98. Maloney, P.R., Begelman, M.C. and Nowak, M.A.: Radiation-driven Warping. II. Nonisothermal Disks. ApJ, **504**, 77 (1998)
MEI91. Mellema, G., Eulderink, F. and Icke, V.: Hydrodynamical models of aspherical planetary nebulae. A&A, **252**, 718-732 (1991)

MKR02. Mellema, G., Kurk, J.D. and Röttgering, H.J.A.: Evolution of clouds in radio galaxy cocoons. A&A, **395**, L13-L16 (2002)
PP83. Papaloizou, J.C.B. and Pringle, J.E.: The time-dependence of non-planar accretion discs. MNRAS, **202**, 1181-1194 (1983)
PPK88. Pauldrach, A., Puls, J., Kudritzki, R.P., Mendez, R.H. and Heap, S.R.: Radiation-driven winds of hot stars. V - Wind models for central stars of planetary nebulae. A&A, **207**, 123-131 (1988)
Pet77. Petterson, J.A.: Twisted accretion disks. I - Derivation of the basic equations. ApJ, **214**, 550-559 (1977)
Pri81. Pringle, J.E.: Accretion discs in astrophysics. ARAA, **19**, 137-162 (1981)
Pri96. Pringle, J.E.: Self-induced warping of accretion discs. MNRAS, **281**, 357-361 (1996)
Pri97. Pringle, J.E.: Self-induced warping of accretion discs - Non-linear evolution and application to AGN. MNRAS, **292**, 136 (1997)
ST98. Sahai, R. and Trauger, J.T.: Multipolar Bubbles and Jets in Low-Excitation Planetary Nebulae: Toward a New Understanding of the Formation and Shaping of Planetary Nebulae. AJ, **116**, 1357-1366 (1998)
Sch93. Schwarz, H.E.: Mass loss on the AGB and beyond. (1993)
SL89. Soker, N. and Livio, M.: Interacting winds and the shaping of planetary nebulae. ApJ, **339**, 268-278 (1989)
SL94. Soker, N. and Livio, M.: Disks and jets in planetary nebulae. ApJ, 1994, **421**, 219-224
WC84. Woodward, P. and Colella, P.: The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks J. Comp. Phys., **54**, 115 (1984)

Mesh Refinement Calculations of Gravitational Waves and Black Holes in 3-Dimensions

Dae-Il (Dale) Choi

Laboratory for High Energy Astrophysics, NASA Goddard Space Flight Center, Greenbelt, MD, 20771
Universities Space Research Association, 7501 Forbes Boulevard, #206, Seabrook,MD 20706

Summary. Here, we report our current effort to apply adaptive mesh refinement techniques to the simulations of black hole spacetimes and gravitational waves. We solve Einstein's equations written in the first-order in time and second-order in space form. We demonstrate that using quadratic-order guardcell filling along with "refluxing" of first order derivatives of the variables as interface conditions at the refinement jumps are essential for accurate evolutions of gravitational waves. Some preliminary results for the head-on collisions of binary black holes are also given.

1 Gravitational Wave Astrophysics

Study of binary black hole spacetimes constitutes a fundamental two body problem in general relativiy. It also promises to provide a great opportunity to learn astrophysics through the "eyes" of gravitational radiation generated by such exotic compact objects from the hearts of galaxies. Numerical relativity which aims to accurately calculate black hole dynamics and waveforms generated from such system using computational simulations, becomes essential tools to any researchers who wants to explore the physics of strongly gravitating systems and astrophysics that is enabled by gravitational waves.

Gravitational waves carry astrophysical information about the sources that are otherwise unobtainable. For example, gravitaional waves in the low frequency regime of $10^{-4} - 10^{-1}$ Hz carry information about the supermassive black holes at the center of galaxies and the host galaxies. They also shed lights on the study of structure formation, galaxy merger history, and cosmology. In many cases, gravitational wave signals are the only probe to study them because of lack of electromagnetic signals. Even in the situations where information carried by electromagnetic signals such as γ-ray and X-ray is available, gravitational waves are expected to provide a fundamentally new "window" to the universe because they carry a very different kind of information than electromagnetic signals.

With the operation of ground-based gravitational wave detectors including LIGO in US and the space-borne LISA detector which is scheduled to be launched in 2012,

gravitational wave astrophysics will soon be an exciting new branch of astronomy. Being strongest sources of gravitational signals detectable with these observatories, detailed understanding of the strong field dynamics of binary black holes and the expected waveforms generated from them is important not only to help interpret the gravitational wave signals, but also to help guide design of the detectors themselves.

2 Challenges in Simulating Colliding Black Holes and Gravitational Waves

Inspiraling and merging black hole binaries are believed to be strongest astrophysical sources of gravitational radiation. Study of these sources requires the full numerical solutions of Einstein's equation of general relativity. However, solving Einstein's equations poses a number of difficult challenges.

- First of all, lack of symmetry for general binary systems requires to solve the equations in full 3-dimensions, which is computationally very expensive. This is a particually severe problem if one were to use uniform grid.
- Secondly, gravitational potential of the binary black hole and the gravitational waves generated from it have very different spatial scales. Typically,

$$\lambda_{GW} \sim (10 - 100) M_{BH} \tag{1}$$

  where λ_{GW} is a wavelength of gravitational waves and M_{BH} total mass of binary black hole spacetimes. Schemes that are based on a simple uniform mesh are not adequate to handle this discrepancy in scales.
- Thirdly, the presence of steep gradients and diverging fields around the black hole physical singularities makes it extremely difficult to maintain accuracy and stability.

Therefore, detailed calculations of black hole binary mergers in 3-dimensions will benefit greatly from the application of mesh refinement techniques. This motivates our current 3-dimensional mesh refinement code developement efforts at NASA/GSFC to solve Einstein's equations.

3 Einstein's Equations

A full simulation of binary black hole merger and gravitational waves from the merger is still a daunting task even with mesh refinement. A number of open issues which are still under active research makes it haders to solve the full problem at once. Therefore, to begin with, we adopt a strategy that devides the full problem into two regimes and tackle the sub-problems seperately: study of dynamics of the black holes near the source region and study of the wave propagation in the wave zone farther away from the black holes.

- Source region: This is where a highly nonlinear and complex strong field dynamics occurrs and will benefit by the application of *adaptive* mesh refinement (AMR).
- Wave zone: This is a *linear regime* where effects from the nonlinear terms are small or negligible. It is expected that the use of *fixed* mesh refinement is enough to set up a grid with successitve coarser resolutions farther away from the sources.

Building on the experience gained here, we will later simulate dynamics of merging binary black holes and wave propagation together.

In many commonly used numerical approaches, Einstein's equations are expressed as a system of 10 or more coupled nonlinear partial differential equations. We use an approach based on the "3 + 1" spacetime split [1], in which the initial data is specified on some spacelike slice and then evolved forward in time. Within this framework, the spacetime metric takes the form

$$ds^2 = -\alpha^2 dt^2 + g_{ij}(dx^i + \beta^i dt)(dx^j + \beta^j dt). \tag{2}$$

The geometry of the given spacelike slice is described by the 3–metric g_{ij}. The lapse function α governs the advance of proper time across the surface and the shift vector β^i the motion of the spatial coordinates within the hypersurface as the data is evolved forward in time. α and β are freely–specifiable functions of space and time. (For simplicity, we explicitly set the shift vector $\beta^i = 0$ here.) Then, the Einstein's equations essentially becomes evolution equations of the metric, g_{ij}, and the extrinsic curvature of the hypersurface, K_{ij}, $= -\frac{1}{2}\frac{\partial g_{ij}}{\partial t}$. In reality, we use a slightly different formalism, so-called BSSN formalism [2, 3]. Here original variables $\{g_{ij}, K_{ij}\}$ are decomposed into the conformal variables $\{\psi, K, \tilde{g}_{ij}, \tilde{A}_{ij}, \tilde{\Gamma}^i\}$ defined as follows:

$$e^{4\psi} \equiv \det(g_{ij})^{1/3} \tag{3}$$

$$\tilde{g}_{ij} \equiv e^{-4\psi} g_{ij} \tag{4}$$

$$K \equiv g^{ij} K_{ij} \tag{5}$$

$$\tilde{A}_{ij} \equiv e^{-4\psi}(K_{ij} - \frac{1}{3} g_{ij} K) \tag{6}$$

$$\tilde{\Gamma}^i \equiv -\partial_j \tilde{g}^{ij} \,. \tag{7}$$

Evolution equations for the conformal variablse are cast into the system of equations that are first-order in time and second-order in space.

In the calculations presented below, we use finite difference method with spatial derivatives being calculated by centered differencing and time updating by (iterative) Crank-Nicholson scheme [4].

4 Block-based Adaptive Mesh Refinement

We use PARAMESH software [5] to implement parallelization and the mesh refinement. PARAMESH works on logically Cartesian, or structured, grids and carries out mesh refinement on grid blocks rather than individual cells. The underlying mesh refinement technique is similar to that of Ref. [6], in which grid blocks are bisected in each coordinate direction when refinement is needed.

The grid blocks all have the same logical structure, with *nxb* zones in the $x-$direction, and similarly for *nyb* and *nzb*. Thus, refinement of a block in 1-D yields two child blocks, each having *nxb* zones but with zone sizes a factor of two smaller than in the parent block. When needed, refinement can continue on the child blocks, with the restriction that the grid spacing can change only by a factor of two, or one refinement level, at any location in the spatial domain.

Each grid block is surrounded by a number of guard cell layers that are used in computing finite difference spatial derivatives near the block's boundary. These guard cells must be filled using data from the interior cells of the given block and the adjacent block.

PARAMESH handles the creation of grid blocks, and builds and maintains the data structures needed to track the spatial relationships between blocks. It takes care of all inter-block communications and keeps track of physical boundaries on which particular conditions are set, guaranteeing that the child blocks inherit this information from the parent blocks. In a parallel environment, PARAMESH distributes the blocks among the available processors to achieve load balance, maximize block locality, and minimize inter-processor communications.

5 Gravitational Waves

Here we propagate weak gravitational waves through the computational domain that has fixed mesh refinement boundaries, which simulates a situation where gravitational waves that are generated from the source region propagate through grids where resolutions becomes coarser and coarser farther away from the sources. To set up ghost zones at the refinement jumps, we apply the following conditions.

- Quadratic interpolation to set up fine grid guardcells
- Reflux to set up coarse grid guardcells by considering first derivatives of the variables as "fluxes" and match them, i.e., copy first derivatives computed in the fine grid to the coarse grid.

We found them to be essential for an accurate and stable evolution [7, 8]. This is shown in Figs. (1) and (2). In the figures, we show FMR evolution of gravitational waves. Initial data (metric at $t = 0$) is given by the Teukolsky solution [9]. Analytic solution is such that gravitational waves are linear and progate without spreading leaving flat space behind as it moves across the computational domain. Geodesic coordinate conditions, $\alpha = 1, \beta^i = 0$ for all t, is used. To save computational resources,

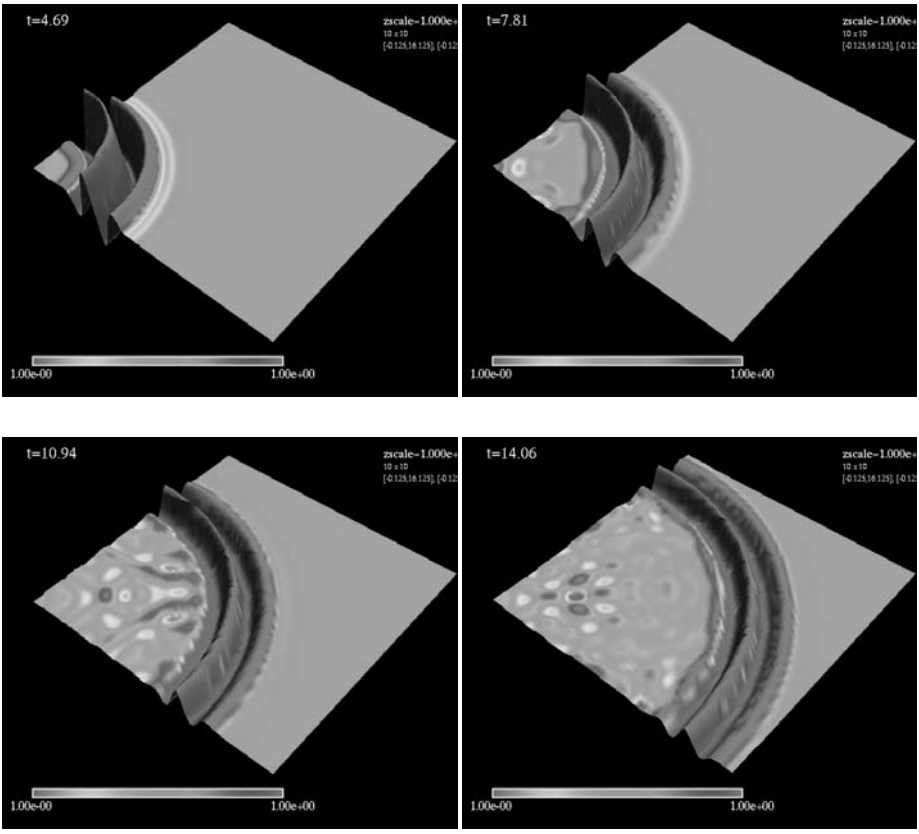


Fig. 1. Gravitational waves: $g_{zz}(x, y)$ on $z = 0$ at 4 different times. Gravitational wave propagates through 2 (fixed) refinement boundaries located near $r = 4.0$ and $r = 8.0$. Linear interpolation used at the refinement jumps and no "refluxing" is applied. Numerical errors of reflected waves are evident.

we carry out the time integration only in one octant of the computational volume. At the outer edge of the grids, simple outgoing wave condition is used.

In Fig. (1), interface conditions at the refinement jump are linear interpolations and no refluxing. This is a usual interface condition when equations are written in fully first order forms. However, for a system of equations that has second derivatives, this interface condition gives rise to reflected waves from the refinement jumps

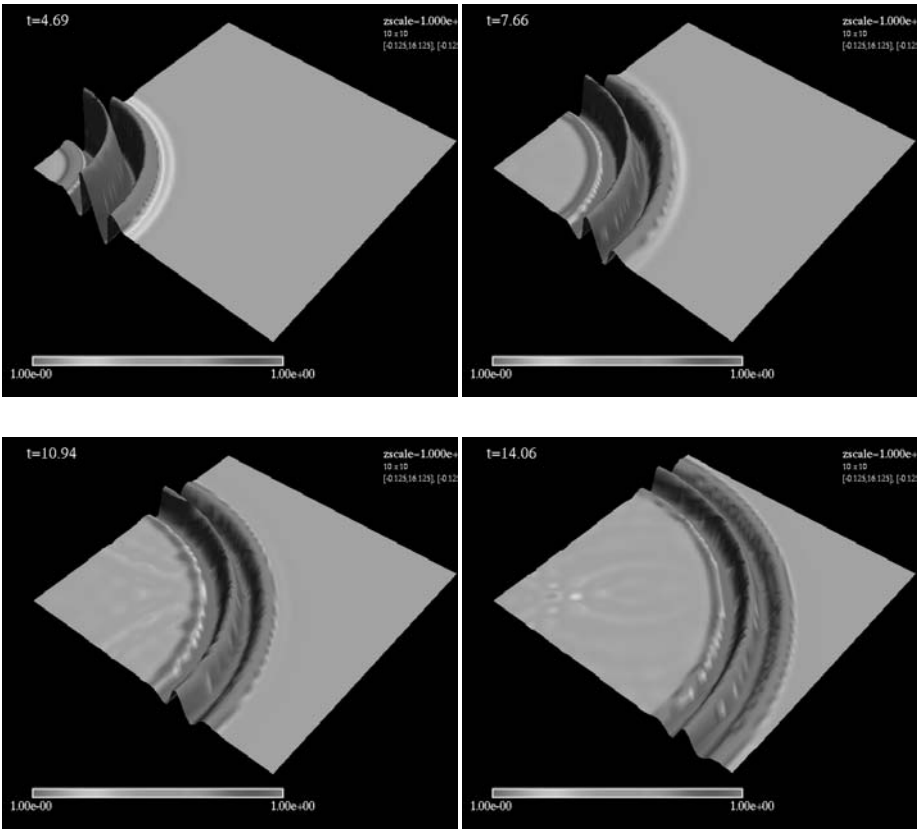


Fig. 2. Gravitational waves: $g_{zz}(x, y)$ on $z = 0$ at 4 different times. Gravitational wave propagates through 2 (fixed) refinement boundaries located near $r = 4.0$ and $r = 8.0$. Quadratic interpolation used at the refinement jumps and "refluxing" is applied.

which is unphysical. The error is basically attributed to the fact that linear interpolation only smoothly connect the functions, not first derivatives.

In Fig. (2), we use quadratic interpolations and refluxing of variables to set up ghost cells at the refinement jumps. Reflection waves are reduced dramatically. Quadratic interpolation and refluxing ensure that the functions and their first derivatives are smoothly connected over the refinement jumps.

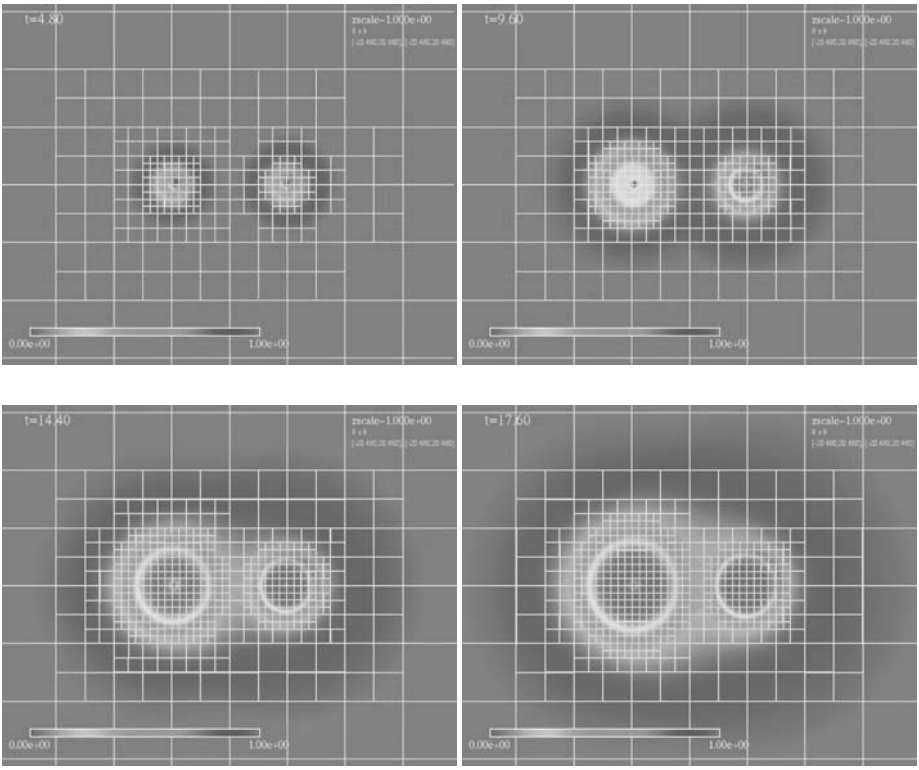


Fig. 3. Black holes: Lapse function $\alpha(x,y)$ on $z=0$ plane for a head-on collision of black holes with unequal masses at 4 different times: a black hole with mass $M=2$ is located at $x=-5$ and a black hole with mass $M=1$ is locate at $x=5$. Holes have zero velocity and zero spin initially. Grid boundaries at ± 20. Yellow lines are block boundaries. Each block has 8^3 cells.

6 Binary Black Holes

Here we apply AMR techniques to a problem of head-on collisions of binary black holes. Black hole singularity is treated analytically using so-called puncture technique [10]. Initial data is two initially non-moving non-spinning black holes of different mass. "1+log" slicing condition is used for lapse function, α. Refinement cri-

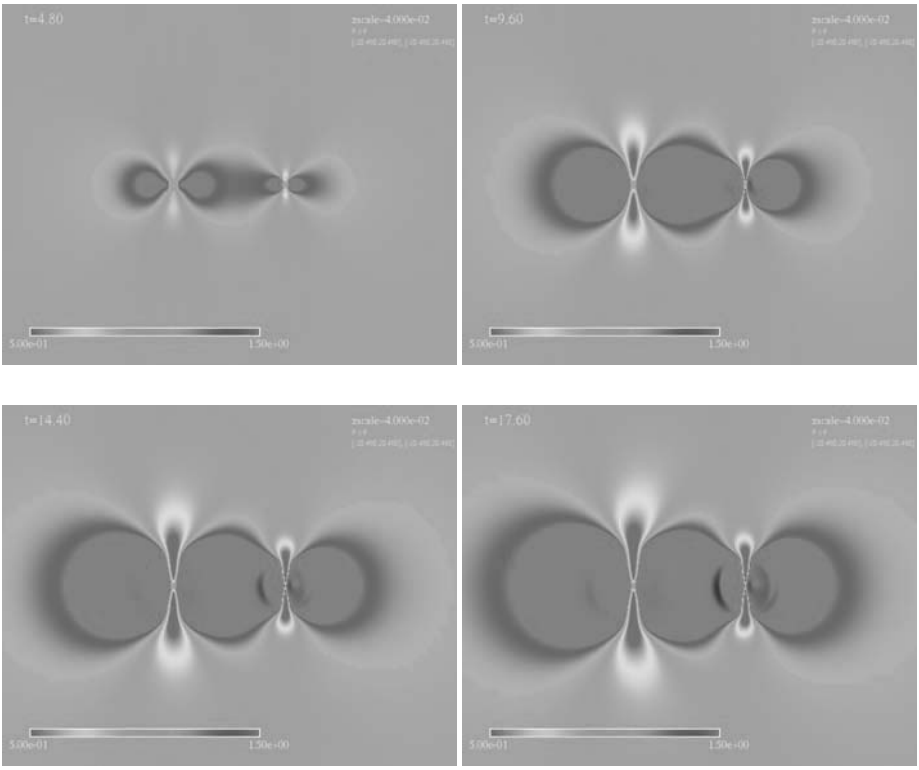


Fig. 4. Metric component g_{xx} on $z = 0$ plane at 4 different times.

teria is based on the error measure computed by first derivatives of the lapse function, α.

Fig. (3) shows expansion of the refined region as lapse function collapses near the centers of each black hole. Here block-boundaries are indicated by yellow lines. Fig. (4) shows metric function g_{xx} on $Z = 0$ plane for 4 different times.

7 Acknowledgments

The author thanks D. Brown, B. Imbiriba, J. Baker, and J. Centrella for helpful discussions.

References

1. C. Misner, K. Thorne, and J. Wheeler, *Gravitation* (W. H. Freeman, New York, 1973)
2. M. Shibata and T. Nakamura, Evolution of three-dimensional gravitational waves: Harmonic slicing case, *Phys. Rev.* **D52**, 5428 (1995).
3. T. Baumgarte and S. Shapiro, Numerical integration of Einstein's field equations, *Phys. Rev.* **D59**, 024007 (1998).
4. S. Teukolsky, On the stability of the iterated Crank-Nicholson method in numerical relativity, *Phys. Rev.* **D61**, 087501 (2000).
5. P. MacNeice, K. Olson, C. Mobarry, R. de Fainchtein, and C. Packer, PARAMESH: A parallel adaptive mesh refinement community toolkit, *Computer Physics Comm.* **126**, 330 (2000).
6. D. DeZeeuw and K. Powell, An Adaptively Refined Cartesian Mesh Solver for the Euler Equations, *J. Computat. Phys.* **104**, 56 (1993).
7. D. Choi, D. Brown et al, Interface Conditions for Wave Propagation Through Mesh Refinement Boundaries. To appear in Journal of Computational Physics (2003).
8. K. New, D. Choi et al, Three-dimensional adaptive evolution of gravitational waves in numerical relativity. Phys. Rev. **D62** 084039 (2000).
9. S. Teukolsky, Linearized quadrupole waves in general relativity and the motion of test particles Equation using Adaptive Mesh Refinement, *Phys. Rev.* **D26**, 745 (1982).
10. S. Brandt and B. Brügmann, Phys. Rev. Lett. **78**, 3606 (1999).

AstroBEAR: AMR for Astrophysical Applications - II: Tests and Applications

Varnière, P.[1], Poludnenko, A.[1], Cunningham, A.[1], Frank, A.[1], Mitran. S.[2]

[1] University of Rochester pvarni,wma,ajc4,afrank@pas.rochester.edu
[2] University of North Carolina at Chapel Hill mitran@amath.unc.edu

1 Introduction

In this contribution we present tests and an application of a new AMR code for astrophysical applications called AstroBEAR. The code is designed on the BEARCLAW framework (Boundary Embedded Adaptive Mesh Refinement for Conservation Laws) [M04] and it offers generalized adaptive facilities including mesh refinement in a grid-based formalism [BL98]. The code is flexible and efficient being designed specifically for multi-physics applications in which processes operating under different time and length scales can be simultaneously simulated. In paper I [PVCFM04] we described the plan of the code including the Riemann solvers, method for stiff source terms and the various procedures being adopted for dealing with errors in $\nabla \cdot B = 0$ condition. In this paper we provide results from selected tests of the MHD code as well as an application of the hydro code to an astrophysical problem.

2 Magneto-Hydrodynamic Tests

1-D Tests: We have carried out a number of shock-tube problem tests to check the accuracy of both the Riemann solver and the behavior of the AMR grid. Note that these problems were actually carried out on a 3-D grid in which only quantities in the x direction were allowed to change. To confirm the behavior of the AMR code we compared high resolution uni-grid simulations to those carried out with different levels of adaptive refinement. We also have compared our AMR simulations to analytic or semi-analytic results where possible ([RJ95], [R02])

Table 1. Initial left and right states for test 1.

	ρ	V_x	V_y	V_z	P	B_x	B_y	B_z
left	1.08	1.2	0.01	0.5	0.95	$2.0/\sqrt{4\pi}$	$3.6/\sqrt{4\pi}$	$2/\sqrt{4\pi}$
right	1.0	0	0	0	1.0	$2.0/\sqrt{4\pi}$	$4.0/\sqrt{4\pi}$	$2/\sqrt{4\pi}$

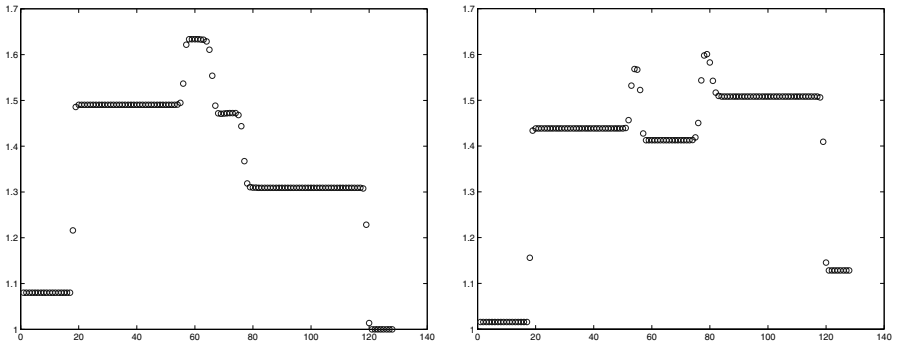


Fig. 1. 1-D shock tube test of AstroBEAR AMR Code. Shown are density (left) and B_y (right). The shock propagates in the x direction.

Figure 1 shows the evolution of 7 waves driven from the initial state presented in the table 1. This test, taken from [B98], uses $\gamma = 1.667$. Examination of results of the simulation demonstrates that the approximate Riemann solver we use does a satisfactory job recovering the published results.

Table 2. Initial left and right states for test 1.

	ρ	V_x	V_y	V_z	P	B_x	B_y	B_z
left	1	0	0	0	1	0.7	0	0
right	0.3	0	0	1	0.2	0.7	1	0

Figure 2 shows the flow driven from an initial state given in table 2 ([RJ95]). In this test 6 waves are generated: a hydro rarefaction; a switch-on slow shock; a contact discontinuity; a slow shock; a rotational discontinuity; a fast rarefaction. Examination of figure 2 again demonstrates that our solver can capture the correct results to better than a a few percent including the shock speed. In addition, here we see that the grid generation algorithm in AstroBEAR accurately tracks the evolution of the steep gradients.

2-D Tests: We have run our 1-D tests obliquely through the grid to test the 2-D performance of the code. We find the waves shown in figures 1 and 2 remain well resolved and accurately tracked. In order to test the ability of our divergence cleaning method however we have also run a true 2-D test in the form of the Orszag-Tang vortex. In paper 1 we presented details of a new cell centered projection method for correcting the magnetic field and removing the $\nabla \cdot \mathbf{B}$ errors and the Orszag-Tang vortex provides a test of this method.

Figure 3 shows a close up of the density in a single vortex of our Orszag-Tang simulation at $t_1 = 0.6$. On the left the simulation is done without any divergence treatment added to the Riemann solver. The effect of the divergence errors on the simulation are apparent in the irregular striations in the vortex. Shortly after t_1 the com-

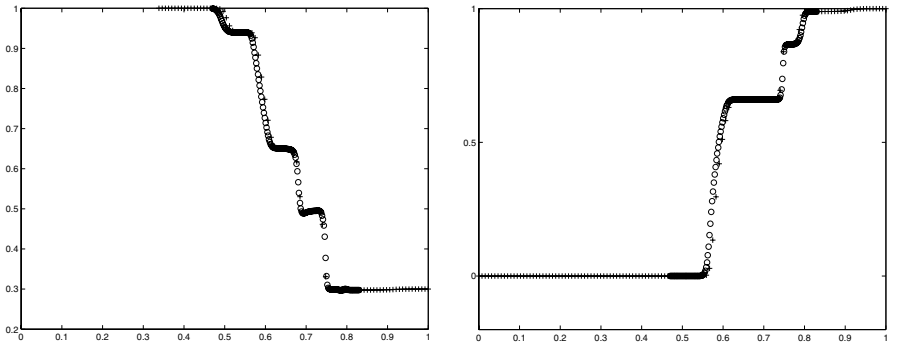


Fig. 2. 1-D shock tube test of AstroBEAR AMR Code. Shown are density (left) and B_y (right). The shock propagates in the x direction. The locations of the refined zones are marked with circles

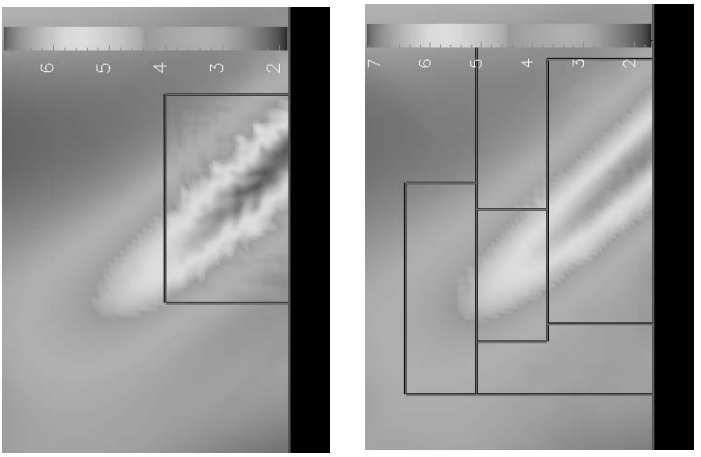


Fig. 3. A comparison of the density fields from the Orszag-Tang vortex $\mathbf{u} = (-\sin y, \sin x, 0)$, $\mathbf{B} = (-\sin y, \sin 2x, 0)$ problem at time $t_1 = 0.6$. The non-physical variations due to divergence build-up seen in the left plot are removed by the projection method as seen in the right plot.

putation terminates due to build-up of divergence errors. On the right we show the simulation in which divergence cleaning with the projection method was used. The flow is smooth and well behaved. This simulation was able to proceed to arbitrary termination times. Both figures show the three-level AMR grids used in the computation. When discontinuities are present, the standard discretization of the Laplace operator through a five-point stencil can be extended. This can be done by incorporating any jumps across the stencil arms as a RHS term in the Poisson equation used in the divergence clean-up step. This approach is still being developed at this time.

3 Hydrodynamic Application: Hypersonic Radiative Bullets

In this section we provide an example of an AstroBEAR application. We choose an astrophysical problem which demonstrates the codes' abilities and performance in terms of the solvers and integration schemes as well as the AMR engines' capacity to resolve various flow features on multiple scales.

Massive outflows are a ubiquitous phenomena associated with both the birth and death of stars. In many cases these flows are both highly structured on large scales and heterogeneous on smaller scales ([Fr99]). Luminious Blue Variables (LBVs) are massive, highly variable stars which experience periods of intense mass loss. The LBV η Carinae is one of the most massive stars in the galaxy and shows many of the characterstics common to all stellar outflows including a prominent bipolar outflow [MDB98].

Of the many questions surrounding the nature of mass loss from η Carinae, the origin of the long thin "strings" observed in the outer nebular regions remain particularly vexing. First observed by [MBW96] and further studied by [WDC99] the strings show two remarkable properties: very high values of length-to-width ratios and the presence of the Hubble-type flows, i.e. linear velocity increase from the base of the strings to their tip. The strings also show a decrease in surface brightness toward the string-head. The true tips may be invisible at optical wavelengths. These features represent the key challenges that must be met by any model that hopes to offer a successful explanation of the string origin and evolution.

Several models have been suggested for the strings including jets [GLRF99] and ionization shadows [S01]. A particularly promising model has been proposed by [RMH02] who suggest that a single hypersonic bullet propagating in the ambient medium generates each of the strings. Since the "Great Eruption" that ejected the main nebula around 1840 was know to be an impulsive event with most of the mass and momentum directed in the polar directions [SGH03] it is plausible that fragments or bullets could have emerged from that event as well.

In what follows we present simulations of radiative hypersonic bullets appropriate to the strings of η Carinae using the AstroBEAR hydrodynamics code. This problem poses a number of computational challenges. The Mach numbers of the bullets are high $10 < M_b < 200$. The initial conditions demand an instanteous acceleration of the clump which leaves a strong vacuum region in the wake. The cooling in the clump, via radiative losses, will produce disparate timescales between the radiative and hydrodynamic processes with $t_{cool}/t_{hydro} < 10^{-5}$ in some cases. Finally, the fragmentation of the flow will leave many high density "clumplets" whose evolution will have to be tracked. Together these issues act as a good test of the entire code.

3.1 Simulation Set-up

There are three dimensionless parameters that describe the evolution of the system. The first two are the density contrast between the bullet and ambient medium $\chi_b \equiv \rho_b/\rho_a$ and the Mach number M_b of the bullet velocity at time $t = 0.0$. These

determine the hydrodynamic regime of the system evolution. The cooling parameter ψ_b, describes the effects of cooling and is defined as

$$\psi_b = \frac{t_{cool}}{t_{hydro}}. \tag{1}$$

Here the hydrodynamic timescale t_{hydro} is defined as the bullet crushing time (or clump crushing time t_{cc} as described by [PFB02])

$$t_{hydro} = \left(\chi_b^{1/2}\left(F_{c1}F_{st}\right)^{-1/2}\right)\frac{2r_b}{v_b}, \tag{2}$$

where r_b is bullet radius, v_b is bullet velocity, $F_{c1} \approx 1.3$ and

$$F_{st} \simeq 1 + \frac{2.16}{1+6.55\chi^{-1/2}}. \tag{3}$$

The two factors F_{c1} and F_{st} relate the unperturbed upstream conditions with the internal bullet post-shock ones and are described in [PFB02]. The interaction of the bullet and ambient medium launches a shock which propagates into the bullet (with post-shock temperature T_{ps} and density ρ_{ps}). The cooling rate Λ_{ps} for such a temperature can be found using the standard cooling curve of [DM72]. The cooling timescale is then estimated as follows

$$t_{cool} = \frac{T_{ps} - T_{min}}{(\gamma-1)\rho_{ps}\Lambda_{ps}} k m_H, \tag{4}$$

where $T_{min} = 100K$ is the minimum temperature gas can cool down to. Using the Rankine-Hugoniot relations the internal bullet post-shock temperature T_{ps} is

$$T_{ps} = T_b\left(1 + \frac{2\gamma}{\gamma+1}\left(M_{is}^2 - 1\right)\right)\left(1 - \frac{2}{\gamma+1}\left(1 - \frac{1}{M_{is}^2}\right)\right), \tag{5}$$

where T_b is the unshocked bullet temperature. Internal bullet post-shock density ρ_{ps} is

$$\rho_{ps} = \rho_b\left(1 - \frac{2}{\gamma+1}\left(1 - \frac{1}{M_{is}^2}\right)\right)^{-1}. \tag{6}$$

M_{is} is the internal bullet shock Mach number described by the expression

$$M_{is} = M_b\left(F_{c1}F_{st}\right)^{1/2}. \tag{7}$$

In our study we carried out simulations covering bullet Mach numbers in the range $M_b = 10 - 200$. For the cooling parameter we have explored a range of conditions from the quasi-adiabatic regime $\psi_b \gtrsim 1$ to the extremely strongly cooled regime $\psi_b \approx 10^{-5}$.

As an example of the behavior of the code we first present a simulation in the strongly cooled regime. Figure 4 shows a Schlieren image (gradient of the density logarithm) in a simulation with $M_b = 10$, $\chi_b = 100$ and $\psi_b = 2.5 \cdot 10^{-2}$. This run

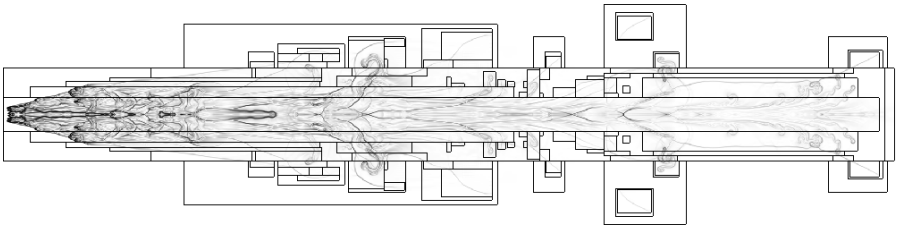


Fig. 4. Radiative clump simulations with $M_b = 10$, $\chi_b = 100$ and $\psi_b = 2.5 \cdot 10^{-2}$. Solid lines show embedded refinement levels.

was carried out in cylindrical symmetry with the symmetry axis defined along the x direction. We use an adaptive grid with 4 levels of refinement. This provided a resolution of 128 cells per cloud radius or an equivalent resolution of 1024×7680 cells. We will discuss the detailed behavior of the flow in radiative bullets below but here we simply wish to note that one can clearly see that the AMR engine is tracking the regions where shocks create high densities and where fragmentation of the clump takes place and leads to the formation of smaller clumplets.

We now focus on a simulation which best matches the conditions observed in the strings of η Carine. This simulation was initialized with bullet radius $r_b = 3.0 \cdot 10^{15}$ cm and initial density $\rho_b = 10^5\ cm^{-3}$ translating into an initial bullet mass $m_b \approx 0.5 \cdot 10^{-5} M_\odot$. The initial bullet Mach number was $M_b = 20$, or $v_b \approx 235$ km s^{-1}. The computational domain was $1.8 \cdot 10^{17}$ cm $\times$ $2.4 \cdot 10^{16}$ cm, or expressed in terms of bullet radii $60 r_b \times 8 r_b$. The ambient density was $\rho_a = 100\ cm^{-3}$ and the ambient temperature $T_a = 10^4 K$. It is assumed that initially the bullet is in pressure equilibrium with the ambient material.

This run has the same resolution as that shown in Figure 4. This resolution is sufficient to place it in the converged regime in the adiabatic case [KMC94]. In the cooling case one has to be more precise in one's definition of convergence. Since the key process is bullet fragmentation via instabilities at the upstream surface, our criterion of convergence was the constancy of the initial fragmentation spectrum. In this sense the above resolution ensured that the instability wavelength, and therefore the number of fragments into which the bullet breaks up, does not change with increasing resolution. On the other hand, it should be mentioned that such resolution might not be sufficient to track the contraction of each fragment due to radiative cooling to its smallest size defined by the equilibrium between cooling and external heating. Outflow boundary conditions were prescribed on all boundaries.

For the simulation described above the cooling parameter was $\psi_b = 2.8 \cdot 10^{-5}$. The hydrodynamic timescale, i.e. the bullet crushing time, was $t_{hydro} = 2.54 \cdot 10^9\ s \approx 80\ yrs.$, the cooling timescale was $t_{cool} = 70.9 \cdot 10^5\ s \approx 82\ days$. The total run time was $8.26 \cdot 10^9\ s = 261.6\ yrs$. Thus the evolution of the system was in a regime strongly dominated by cooling.

When the interaction of an inhomogeneity proceeds in the adiabatic regime, the dominant process is the lateral clump re-expansion [KMC94, PFB02]. The internal shock compresses the clump and once such compression is completed the re-

expansion begins into regions of lowest total pressure, i.e. the lateral direction. Such re-expansion is responsible for the destruction of the clump, (when acting in combination with instabilities at the upstream clump surface). When more than one clump is present lateral re-expansion also drives interclump interactions via merging and formation of larger structures that subsequently alter the global flow [PFB02].

The fundamental distinction between the radiatively cooled inhomogeneous systems, including individual bullets, and the adiabatic ones is the minimal role of lateral re-expansion. Instead, the dominant process is the formation of instabilities at the upstream surface of the bullet with a wavelength significantly smaller than the one observed in the adiabatic case. As the bullet drives through the ambient medium hydrodynamic instabilities (Richtmeyer-Meshkov, Rayleigh-Taylor) produce the initial instability seed. The resulting density variations quickly trigger the onset of thermal instabilities which then become the dominant process responsible for bullet fragmentation. In our simulation such instabilities start developing with a wavelength of about 10% of the initial bullet radius or $3.0 \cdot 10^{14}$ cm $\approx$ 20 a.u. This initial scale is crucial for subsequent evolution since it determines not only the continuing process of bullet fragmentation but also the structure of the downstream flow. As was mentioned above, we did not observe any significant changes in the fragmentation spectrum with changing grid resolution, therefore we believe that we observed the true fragmentation spectrum corresponding to the given flow conditions. Of course this conclusion may be altered with fully 3-D simulations.

One noteworthy point is that the evolution of the bullet and its fragments appear to proceed in such a manner that there is constant mass loss of bullet material. This implies a steady "mass loading" of the downstream flow via the hydrodynamic ablation. Mass loading has been claimed to be an important process in all clumpy hydrodynamic systems [HDPS86, HD88].

Since the wavelength of the initial fragmentation spectrum is approximately constant along the bullet radius, the fragments produced by the instability are of different mass with the most massive being closest to the symmetry axis and the outermost being the lightest. As a result the fragments are "peeled off" from the bullet one by one starting with the outermost in radius. Each fragmentation event results in the formation of a distinct "ring-like" feature in the bullet wake. The formation or rings is a consequence of the axisymmetry of these simulations. In 3-D it is likely that the rings would themselves fragment [KBPB03].

Figure 5 shows the computational domain at time $t = 261.6$ yrs. In Figure 5a the synthetic Schlieren image (gradient of the density logarithm) is shown illustrating the shock and vortex sheet structure in the flow. Figure 5b shows the synthetic observation image of the computational domain. Since our simulation did not track the full ionization dynamics of the flow, the image represents the total radiative energy losses summed along each ray. The 2D distribution of the state vector obtained in the simulation was extended using cylindrical symmetry to a $2048 \times 2048 \times 7680$ cells 3D data cube. Thus the synthetic observation image represents the 2D projected distribution of emissivity I integrated in the z-direction according to the formula

$$I_{ij} = \sum_k n_{ijk}^2 \Lambda_{ijk}(T_{ijk}), \tag{8}$$

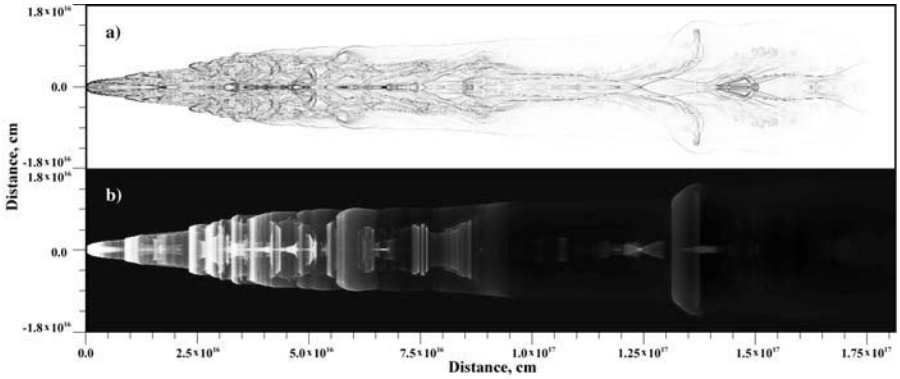


Fig. 5. a) Synthetic Schlieren image of the computational domain at time 261.6 yrs. Shown is the gradient of the density logarithm. b) Synthetic observation image of the computational domain for the same time as in a). Note the periodic ring-like structures in the domain resulting from individual fragmentation episodes.

where i, j, and k are the cell indices in the x-, y-, and z-direction respectively, and the cooling rate $\Lambda(T)$ here, as well as in the simulation, was determined based on the cooling curve described by [DM72].

The second process that determines the structure of the flow in the wake is gas re-expansion into the cavity excavated by the bullet. The highest temperature reached by gas in the system is at the tip of the central bullet fragment and is about $3.5 \cdot 10^5$ K. Gas passing through the bullet bowshock cools down very rapidly. That weakens the bowshock and makes it more oblique, which in its turn prevents further heating of the ambient gas. As it can be seen in Figure 5a, the bowshock essentially disappears half-way downstream from the bullet head. Beyond the turbulent region immediately behind the bullet head and in between the stripped bullet fragments, the gas tends to re-expand essentially at the sound speed of the ambient gas and fill the cavity. Such re-expansion causes gas to rebound on the axis resulting in a reflected shock. The effect of this shock can be seen in Figure 5a around the symmetry axis as a collection of features that are narrower than the bowshock.

3.2 Velocity Distribution and Hubble-type Flow

Another key property of the strings is the presence of the Hubble-type flows in the bullet wake. Left panel of Figure 6 shows the distribution of the total velocity $v_{tot}(x)$ along the symmetry axis of the bullet/wake as a function of distance from the bullet head. The linear velocity decrease from the maximum value of 210 km s^{-1} from head to base is clear aside from some minor fluctuations arising due to the unsteadiness of the downstream flow. Note that very little deceleration of the bullet head has occured. Given that the initial bullet velocity is 235 km s^{-1} we see that after 260 yrs. in which the bullet material traveled the distance of $1.8 \cdot 10^{17}$ cm ≈ 0.058 pc, the bullet material lost only $\approx 10\%$ of its original velocity. That is due to the fact that the

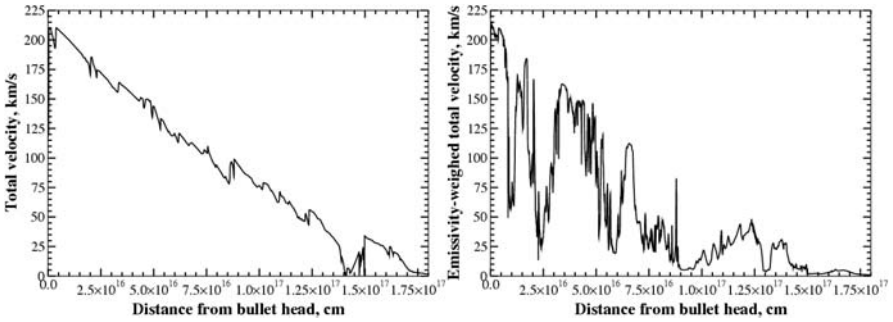


Fig. 6. *Left:* distribution of total velocity along the symmetry axis of the bullet at the time 261.6 yrs. (same time in the simulation as the one shown in Figure 5. *Right:* distribution of the emissivity-weighed total velocity along the symmetry axis of the bullet at the same time.

central fragment retains most of the original bullet mass while it has a rather small cross-section due to the cooling-induced contraction.

One should be cautious, however, in directly comparing the total velocity distribution shown in the left panel of Figure 6 to the observationally determined velocity distributions in the strings of η Carinae (cf. Figure 6 in [WDC99]). Quantities, more closely resembling the ones that are obtained observationally, should be employed in order to make such comparison relevant. For example, one can create emissivity-weighed total velocity $v_{emis}(x)$ maps of the flow. This distribution along the x-axis is shown in the right panel of Figure 6. The distribution of this quantity is significantly more noisy than the total velocity cross-cut. However, the local maxima of $v_{emis}(x)$ roughly tend to fall on the same line as that in the previous figure giving some indication of the presence of the Hubble-type flow.

Acknowledgments This research was sponsored by DOE/NNSA Cooperative Agreement DE-F03-02NA00057 with Cornell University by subcontract NO 41843-7012. as well as by NSF grant AST-9702484, NASA grant NAG5-8428 and the Laboratory for Laser Energetics under DOE sponsorship.

References

B98. Balsara, D., ApJSS 116:13-153, 1998
BL98. M.J. Berger, R.J. LeVeque, *SIAM J. Numer. Anal.*, 35(6):2298, 1998
DM72. A. Dalgarno, R.A. McCray, *Ann. Rev. Astron. & Astrophys.*, 10:375, 1972
Fr99. Frank, A. 1999, "Bipolar Outflows and the Evolution of Stars", New Astronomy Reviews, 43,31
GLRF99. G. García-Segura, N. Langer, M. Różyczka, J. Franco, *Astrophys. J.*, 517:767, 1999
HD88. T.W. Hartquist, J.E. Dyson, 1988, *Astrophys. & Space Sci.*, 144:615, 1988
HDPS86. T.W. Hartquist, J.E. Dyson, M. Pettini, L.J. Smith, 1986, *Mon. Not. Royal Astron. Soc.*, 221:715, 1986
KBPB03. R.I. Klein, K.S. Budil, T.S. Perry, D.R. Bach, 2003, *Astrophys. J.*, 583:245, 2003

KMC94. R.I. Klein, C.F. McKee, P. Colella, *Astrophys. J.*, 420:213, 1994
MBW96. J. Meaburn, P. Boumis, J.R. Walsh, W. Steffen, A.J. Holloway, R.J.R. Williams, M. Bryce, *Mon. Not. Royal Astron. Soc.*, 282:1313, 1996
M04. S. Mitran, *These proceedings*, 2004
MDB98. J.A. Morse, K. Davidson, J. Bally, D. Ebbets, B. Balick, A. Frank, *Astrophys. J.*, 116:2443, 1998
PFB02. A.Y. Poludnenko, A. Frank, E.G. Blackman, *Astrophys. J.*, 576:832, 2002
PVCFM04. A.Y. Poludnenko, P. Varnière, A. Cunningham, A. Frank, S. Mitran, *These proceedings*, 2004
RMH02. M.P. Redman, J. Meaburn, A.J. Holloway, *Mon. Not. Royal Astron. Soc.*, 332:754, 2002
RJ95. D.S. Ryu & T.W. Jones, *Astrophys. J.*, 442:228, 1995
R02. J.A. Rossmanith, *Ph.D. Thesis*, University of Washington,2002
SGH03. N. Smith, R.D. Gehrz, P.M. Hinz, W.F. Hoffmann, J.L. Hora, E.E. Mamajek, M.R. Meyer, 2003, *Astrophys. J.*, 125:1458, 2003
S01. N. Soker, *Astron. & Astrophys.*, 377:672, 2001
WDC99. K. Weis, W.J. Duschl, Y.-H. Chu, *Astron. & Astrophys.*, 349:467, 1999

Parallel, AMR MHD for Global Space Weather Simulations

Kenneth G. Powell, Darren L. De Zeeuw, Igor V. Sokolov, Gábor Tóth, Tamas I. Gombosi, and Quentin Stout

Center for Space Environment Modeling
University of Michigan
Ann Arbor, MI 48109 powell@umich.edu

Summary. This paper presents the methodology behind and results of adaptive mesh refinement in global magnetohydrodynamic models of the space environment. Techniques used in solving the governing equations of semi-relativistic magnetohydrodynamics (MHD) are presented. These techniques include high-resolution upwind schemes, block-based solution-adaptive grids, explicit, implicit and partial-implicit time-stepping, and domain decomposition for parallelization. Recent work done in coupling the MHD model to upper-atmosphere and inner-magnetosphere models is presented, along with results from modeling a solar coronal mass ejection and its interaction with Earth's magnetosphere.

1 Introduction

Space weather is the term that has been coined to describe the intricate processes coupling the Sun to the geospace environment. Conditions on the Sun, in the solar wind, and in the Earth's magnetosphere, ionosphere and thermosphere can influence the performance and reliability of space- and ground-based technological systems, and can endanger human life and health. Global computational models for space weather, based on first principles, are beginning to be developed by a number of research groups. The hope for such models is that they will eventually fill the gaps left by measurements, extending the spatially and temporally limited observational database into a self-consistent global understanding of our space environment.

Presently, and in the foreseeable future, magnetohydrodynamic (MHD) models are the only models that can span the enormous distances present in the Earth-Sun system. In addition, adaptive mesh refinement (AMR) is a necessity, due to the enormous range of scales: the distance from Earth to Sun is 23,456 R_E; the Earth's bow shock standoff distance is around 15 R_E; the length of the magnetotail is around 100 R_E.

However, it should not be forgotten that even generalized MHD equations are only a relatively low-order approximation to more complete physics; they provide only a simplified description of natural phenomena in space plasmas. To address this

fact, current work in space-weather modeling is, in part, focused on coupling global MHD models for the region between the Earth and the Sun to models specifically developed for the near-Sun and near-Earth regions. Both the global AMR MHD model developed by the authors, and the coupling to other models for use in modeling the entire space-weather problem, are discussed below.

2 Non-Relativistic Magnetohydrodynamics

The governing equations for an ideal, non-relativistic, compressible plasma may be written in a number of different forms. While the different forms of the MHD equations describe the same physics at the differential equation level, there are important practical differences when one solves discretized forms of the various formulations.

2.1 Fully Conservative Form

The fully conservative form of the equations is

$$\frac{\partial \mathbf{U}}{\partial t} + (\nabla \cdot \mathbf{F})^{\mathrm{T}} = 0 \,, \tag{1}$$

where $\mathbf{U}$ is the vector of conserved quantities and $\mathbf{F}$ is a flux diad,

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho \mathbf{u} \\ B \\ E_{mhd} \end{pmatrix} \tag{2}$$

$$\mathbf{F} = \begin{pmatrix} \rho \mathbf{u} \\ \rho \mathbf{u}\mathbf{u} + \left(p + \frac{1}{2\mu_0} B^2 \right) \mathbf{I} - \frac{1}{\mu_0} \mathbf{B}\mathbf{B} \\ \mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u} \\ \mathbf{u} \left(E_{mhd} + p + \frac{1}{2\mu_0} B^2 \right) - \frac{1}{\mu_0} (\mathbf{u} \cdot \mathbf{B}) \mathbf{B} \end{pmatrix}^{\mathrm{T}} \tag{3}$$

where E_{mhd} is the magnetohydrodynamic energy, given by

$$E_{mhd} = \frac{1}{2} \rho u^2 + \frac{1}{\gamma - 1} p + \frac{1}{2\mu_0} B^2 \tag{4}$$

2.2 Symmetrizable Formulation

Symmetrizable systems of conservation laws have been studied by a number of authors, including Godunov [11] and Harten [14]. One property of the symmetrizable form of a system of conservation laws is that an added conservation law

$$\frac{\partial (\rho s)}{\partial t} + \frac{\partial (\rho s u_x)}{\partial x} + \frac{\partial (\rho s u_y)}{\partial y} + \frac{\partial (\rho s u_z)}{\partial z} = 0$$

for the entropy s can be derived by a linear combination of the system of equations. For the ideal MHD equations, as for the gasdynamic equations, the entropy is $s = \log(p/\rho^\gamma)$. Another property of symmetrizable systems is that they are Galilean invariant; all waves in the system propagate at speeds $u \pm c_w$ (for MHD, the possible values of c_w are the Alfvén, magnetofast and magentoslow speeds). Neither of these properties holds for the fully conservative form of the MHD equations.

Godunov showed that the fully conservative form of the MHD equations (eq. 1) is not symmetrizable [11]. The symmetrizable form may be written as

$$\frac{\partial \mathbf{U}}{\partial t} + (\nabla \cdot \mathbf{F})^{\mathrm{T}} = \mathbf{Q}\,, \tag{5}$$

where

$$\mathbf{Q} = -\nabla \cdot \mathbf{B} \begin{pmatrix} 0 \\ \frac{1}{\mu_0}\mathbf{B} \\ \mathbf{u} \\ \frac{1}{\mu_0}\mathbf{u} \cdot \mathbf{B} \end{pmatrix} \tag{6}$$

Vinokur separately showed that eq. (5) can be derived starting from the Euler equations and Maxwell equations, if no stipulation is made about $\nabla \cdot \mathbf{B}$ in the derivation. Powell showed that this symmetrizable form can be used to derive a Roe-type approximate Riemann solver for solving the MHD equations in multiple dimensions [20].

The MHD eigensystem arising from Eq. (1) or Eq. (5) leads to eight eigenvalue/eigenvector pairs. The eigenvalues and associated eigenvectors correspond to an entropy wave, two Alfvén waves, two magnetofast waves, two magnetoslow waves, and an eighth eigenvalue/eigenvector pair that depends on which form of the equations is being solved. This last wave (which describes the jump in the normal component of the magnetic field at discontinuities) has a zero eigenvalue in the fully conservative case, and an eigenvalue equal to the normal component of the velocity, u_n, in the symmetrizable case. The expressions for the eigenvectors, and the scaling of the eigenvectors, are more intricate than in gasdynamics [30].

The symmetrizable formulation (given by Eq. 5) is formally not fully conservative. Terms of order $\nabla \cdot \mathbf{B}$ are added to what would otherwise be a divergence form. The danger of this is that shock jump conditions may not be correctly met, unless the added terms are small, and/or they alternate in sign in such a way that the errors are local, and in a global sense cancel in some way with neighboring terms. This downside, however, has to be weighed against the alternative; a system (i.e., the one without the source term) that, while conservative, is not Gallilean invariant, has a zero eigenvalue in the Jacobian matrix, and is not symmetrizable. In practice, using the symmetrizable formulation, and adding a technique to keep the magnitude of $\nabla \cdot \mathbf{B}$ small, is a viable approach, and the one used in this work.

3 Semi-Relativistic Plasmas

While the solar-wind speed remains non-relativistic in the solar system, the intrinsic magnetic fields of several planets in the solar system are high enough, and the density of the solar wind low enough, that the Alfvén speed,

$$V_{\mathrm{A}} = \sqrt{\frac{B^2}{\mu_0 \rho}}$$

can reach appreciable fractions of the speed of light. In the case of Jupiter, the Alfvén speed in the vicinity of the poles is of order ten times the speed of light! Even Earth has a strong enough intrinsic magnetic field that the Alfvén speed reaches twice the speed of light in Earth's near-auroral regions.

For these regions, solving the non-relativistic ideal MHD equations does not make sense. Having waves in the system propagating faster than the speed of light, besides being non-physical, causes a number of numerical difficulties. However, solving the fully relativistic MHD equations is not practical, nor really necessary. What is called for is a semi-relativistic form of the equations, in which the flow speed and acoustic speed are non-relativistic, but the Alfvén speed can be relativistic. A derivation of these semi-relativistic equations from the fully relativistic equations is given in [12]: the final result is presented here.

3.1 The Semi-Relativistic MHD Equations

The semi-relativistic ideal MHD equations are of the form

$$\frac{\partial \mathbf{U}_{sr}}{\partial t} + (\nabla \cdot \mathbf{F}_{sr})^{\mathrm{T}} = 0 \tag{7}$$

where the state vector, $\mathbf{U}_{sr}$, and the flux diad, $\mathbf{F}_{sr}$, are

$$\mathbf{U}_{sr} = \begin{pmatrix} \rho \\ \rho \mathbf{u} + \frac{1}{c^2}\mathbf{S}_{\mathrm{A}} \\ \mathbf{B} \\ \frac{1}{2}\rho u^2 + \frac{1}{\gamma - 1} p + e_{\mathrm{A}} \end{pmatrix} \tag{8}$$

$$\mathbf{F}_{sr} = \begin{pmatrix} \rho \mathbf{u} \\ \rho \mathbf{u}\mathbf{u} + p\mathbf{I} + \mathbf{P}_{\mathrm{A}} \\ \mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u} \\ \left(\frac{1}{2}\rho u^2 + \frac{\gamma}{\gamma - 1} p \right) \mathbf{u} + \mathbf{S}_{\mathrm{A}} \end{pmatrix}^{\mathrm{T}} \tag{9}$$

In the above,

$$\mathbf{S}_{\mathrm{A}} = \frac{1}{\mu_0} (\mathbf{E} \times \mathbf{B}) \tag{10}$$

$$e_{\mathrm{A}} = \frac{1}{2\mu_0} \left(B^2 + \frac{1}{c^2} E^2 \right) \tag{11}$$

$$\mathbf{P}_{\mathrm{A}} = e_{\mathrm{A}} \mathbf{I} - \frac{1}{\mu_0} \mathbf{B}\mathbf{B} - \frac{1}{\mu_0 c^2} \mathbf{E}\mathbf{E} \tag{12}$$

are the Poynting vector, the electromagnetic energy density, and the electromagnetic pressure tensor, respectively. The electric field **E** is related to the magnetic field **B** by Ohm's law.

3.2 Lowering the Speed of Light

This new system of equations has wave speeds that are limited by the speed of light; for strong magnetic fields, the modified Alfvén speed (and the modified magnetofast speed) asymptote to c. The modified magnetoslow speed asymptotes to a, the acoustic speed. This property offers the possibility of a rather tricky convergence-acceleration technique for explicit time-stepping schemes, first suggested by Boris [6]; the wave speeds can be lowered, and the stable time-step thereby raised, by artificially lowering the value taken for the speed of light. This method is known as the "Boris correction."

The semi-relativistic MHD equations given above are valid in physical situations in which $V_A > c$. A slight modification yields a set of equations, the steady-state solutions of which are independent of the value taken for the speed of light. Defining the true value of the speed of light to be c_0, to distinguish it from the artificially lowered speed of light, c, the equations are:

$$\frac{\partial \mathbf{U}_{sr}}{\partial t} + (\nabla \cdot \mathbf{F}_{sr})^{\mathrm{T}} = \mathbf{Q}_{c_0} \tag{13}$$

where the state vector, $\mathbf{U}_{sr}$, and the flux diad, $\mathbf{F}_{sr}$, are as defined above, and the new source term in the momentum equation is

$$\mathbf{Q}_{c_0} = \frac{1}{\mu_0}\left(\frac{1}{c_0^2} - \frac{1}{c^2}\right)\mathbf{E}\nabla \cdot \mathbf{E}$$

An implementation of the semi-relativistic equations has been made in the work presented here. Typically, the non-relativistic form of the equations is solved everywhere but near planets with intrinsic magnetic fields, where the semi-relativistic form is solved.

4 Solution Techniques

4.1 The Upwind Finite-Volume Scheme

The MHD equations are well suited for finite volume methods when the governing equations are integrated over a computational cell i, yielding

$$\frac{d\mathbf{U}_i}{dt} = -\frac{1}{V_i}\sum_{\text{faces}} \mathbf{F} \cdot \hat{\mathbf{n}} A - \frac{\mathbf{Q}_i}{V_i}\sum_{\text{faces}} \mathbf{B} \cdot \hat{\mathbf{n}} A\,, \tag{14}$$

where V_i is the volume of cell i, A is the surface area of the faces forming the computational cell, $\hat{\mathbf{n}}$ is the unit vector normal to the cell faces, $\mathbf{U}_i$ is the cell-averaged conserved solution vector, and $\mathbf{Q}_i$ is given by

$$\mathbf{Q}_i = - \begin{bmatrix} 0 \\ \frac{1}{\mu_0}\mathbf{B}_i \\ \mathbf{u}_i \\ \frac{1}{\mu_0}\mathbf{u}_i \cdot \mathbf{B}_i \end{bmatrix} . \tag{15}$$

The numerical face fluxes, $\mathbf{F} \cdot \hat{\mathbf{n}}$, are defined in terms of the left and right interface solution states, $\mathbf{U}_\mathrm{L}$ and $\mathbf{U}_\mathrm{R}$, as follows

$$\mathbf{F} \cdot \hat{\mathbf{n}} = \mathcal{F}\left(\mathbf{U}_\mathrm{L}, \mathbf{U}_\mathrm{R}, \hat{\mathbf{n}}\right) , \tag{16}$$

where $\mathbf{U}_\mathrm{L}$ and $\mathbf{U}_\mathrm{R}$ are the state vectors at the left and right sides of the interface.

Because the MHD equations are a system of hyperbolic conservation laws, many of the techniques that have been developed for the Euler equations can be applied relatively straightforwardly. In particular, the high-resolution finite-volume approach of van Leer [33] (i.e. approximate Riemann solver + limited interpolation scheme + multi-stage time-stepping scheme) is perfectly valid. The Rusanov/Lax-Friedrichs approximate Riemann solver can be applied directly; no knowledge of the eigensystem of the MHD equations is required other than the fastest wave speed in the system. A Roe-type scheme [29] can be constructed for non-relativistic MHD [21], but requires more work, because of the complexity of the eigensystem. In addition, an HLLE-type Riemann solver has been derived by Linde [17]; it is less dissipative than the Rusanov/Lax-Friedrichs scheme, but less computationally intensive than the Roe scheme. Whichever approximate Riemann solver is chosen to serve as the flux function, standard interpolation schemes and limiters can be used to construct a finite-volume scheme.

One way in which the numerical solution of the MHD equations differs from that of the gasdynamic equations is the constraint that $\nabla \cdot \mathbf{B} = 0$. Enforcing this constraint numerically, particularly in shock-capturing codes, can be done in a number of ways, but each way has its particular strengths and weaknesses. This issue is explained more fully in a number of references such as [7, 20, 21, 8, 18, 9]. Tóth has published a numerical comparison of many of the approaches for a suite of test cases [31].

5 Block-Based AMR for MHD

For typical solar-wind flows, length scales can range from tens of kilometers in the near-Earth region to the Earth-Sun distance (1.5×10^{11} m), and timescales can range from a few seconds near the Sun to the expansion time of the solar wind from the Sun to the Earth ($\sim 10^5$ s). The use of AMR is not only extremely beneficial, but a virtual necessity for solving problems with such disparate spatial and temporal scales.

Building on prior work by Berger [1, 3, 4, 5] and work by Quirk [22, 23], and keeping in mind the desire for high performance on massively parallel computer architectures, a relatively simple yet effective block-based AMR technique has been developed and is used in conjunction with the finite-volume scheme described above. Here the governing equations are integrated to obtain volume-averaged solution quantities within rectangular Cartesian computational cells. The computational

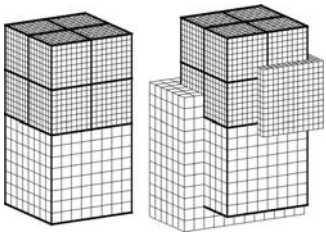


Fig. 1. (left) Self-similar blocks used in parallel block-based AMR scheme. (right) Self-similar blocks illustrating the double layer of ghost cells for both coarse and fine blocks.

cells are embedded in regular structured blocks of equal sized cells. The blocks are geometrically self-similar with dimensions $\tilde{\ell}_x \times \tilde{\ell}_y \times \tilde{\ell}_z$ and consist of $N_x \times N_y \times N_z$ cells, where $\tilde{\ell}_x$, $\tilde{\ell}_y$, and $\tilde{\ell}_z$ are the nondimensional lengths of the sides of the rectangular blocks and N_x, N_y, and N_z are even, but not necessarily all equal, integers. Typically, blocks consisting of anywhere between $4 \times 4 \times 4 = 64$ and $12 \times 12 \times 12 = 1728$ cells are used (see Figure 1). Solution data associated with each block are stored in standard indexed array data structures. It is therefore straightforward to obtain solution information from neighboring cells within a block.

Computational grids are composed of many self-similar blocks. Although each block within a grid has the same data storage requirements, blocks may be of different sizes in terms of the volume of physical space that they occupy. Starting with an initial mesh consisting of blocks of equal size (i.e., equal resolution), adaptation is accomplished by the dividing and coarsening of appropriate solution blocks. In regions requiring increased cell resolution, a "parent" block is refined by dividing itself into eight "children" or "offspring." Each of the eight octants of a parent block becomes a new block having the same number of cells as the parent and thereby doubling the cell resolution in the region of interest. Conversely, in regions that are deemed overresolved, the refinement process is reversed, and eight children are coarsened and coalesced into a single parent block. In this way, the cell resolution is reduced by a factor of 2. Standard multigrid-type restriction and prolongation operators are used to evaluate the solution on all blocks created by the coarsening and division processes, respectively.

The decision of where to adapt is made based on user-determined adaptation criteria. The user can choose up to three criteria from a large list (typical criteria are $|\nabla \times \mathbf{B}|$, $|\nabla \times \mathbf{u}|$, $|\nabla p|$). Each block is assigned a value equal to the maximum over all cells of the quantity being used in the adaptation criterion. The blocks are then sorted by these values. The top of the list is flagged for refinement; the bottom of the list is flagged for coarsening. The number of blocks flagged for refinement and coarsening is based on user-specified rules for the percentage of blocks to be refined or coarsened.

Two neighboring blocks, one of which has been refined and one of which has not, are shown in Figure 1. Any of the blocks shown in Figure 1 can in turn be refined, and so on, leading to successively finer blocks. In the present method, mesh

refinement is constrained such that the cell resolution changes by only a factor of 2 between adjacent blocks and such that the minimum resolution is not less than that of the initial mesh.

In order that the update scheme for a given iteration or time step can be applied directly to all blocks in an independent manner, some additional solution information is shared between adjacent blocks having common interfaces. This information is stored in an additional two layers of overlapping "ghost" cells associated with each block as shown in Figure 1. At interfaces between blocks of equal resolution, these ghost cells are simply assigned the solution values associated with the appropriate interior cells of the adjacent blocks. At resolution changes, restriction and prolongation operators, similar to those used in block coarsening and division, are employed to evaluate the ghost cell solution values. After each stage of the multistage time-stepping algorithm, ghost cell values are reevaluated to reflect the updated solution values of neighboring blocks. With the AMR approach, additional interblock communication is also required at interfaces with resolution changes to strictly enforce the flux conservation properties of the finite-volume scheme [1, 2, 3]. In particular, the interface fluxes computed on more refined blocks are used to correct the interface fluxes computed on coarser neighboring blocks so as to ensure that the fluxes are conserved across block interfaces.

6 Parallel Implementation

6.1 Explicit Time-Stepping

The parallel block-based AMR solver was designed from the ground up with a view to achieving very high performance on massively parallel architectures. The underlying upwind finite-volume solution algorithm, with explicit time stepping, has a very compact stencil and is therefore highly local in nature. The hierarchical data structure and self-similar blocks make domain decomposition of the problem almost trivial and readily enable good load-balancing, a crucial element for truly scalable computing. A natural load balancing is accomplished by simply distributing the blocks equally among the processors. Additional optimization is achieved by ordering the blocks using the Peano-Hilbert space filling curve to minimize inter-processor communication. The self-similar nature of the solution blocks also means that serial performance enhancements apply to all blocks and that fine-grain parallelization of the algorithm is possible. The parallel implementation of the algorithm has been carried out to such an extent that even the grid adaptation is performed in parallel.

Other features of the parallel implementation include the use of FORTRAN 90 as the programming language and the message-passing interface (MPI) library for performing the interprocessor communication. Use of these standards greatly enhances the portability of the code and leads to very good serial and parallel performance. The message passing is performed in an asynchronous fashion with gathered wait states and message consolidation.

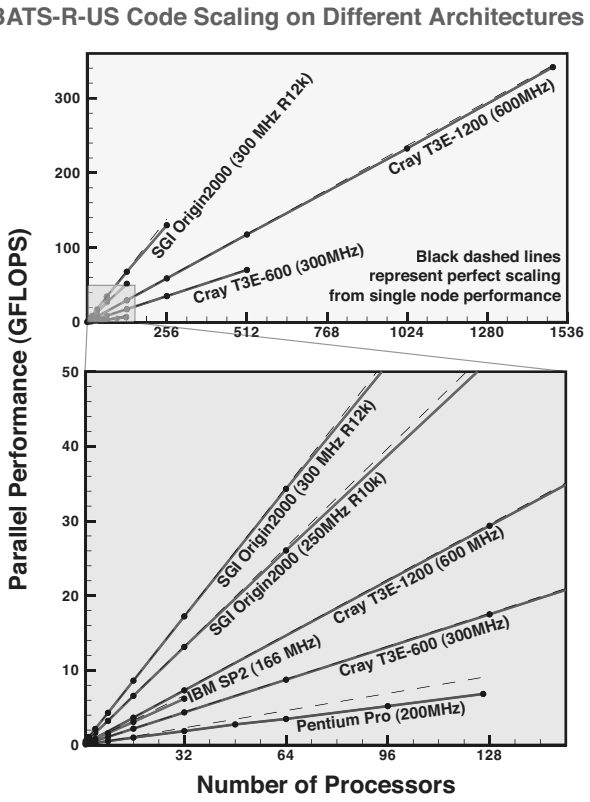


Fig. 2. Parallel speedup of MHD code on various architectures. Black dashed lines represent perfect scaling from single node performance.

Implementation of the algorithm has been carried out on Cray T3E supercomputers, SGI and Sun workstations, on Beowulf type PC clusters, on SGI shared-memory machines, on a Cray T3D, and on several IBM SP2s. The code scales nearly perfectly to 1,500 processors and a sustained speed of 342 GFlops has been attained on a Cray T3E-1200 using 1,490 PEs. For each target architecture, simple single-processor measurements are used to set the size of the adaptive blocks. The scaling on various architectures is shown in Figure 2.

6.2 Implicit Time-Stepping

A number of time-stepping algorithms have been implemented in the AMR MHD solver. The simplest and least expensive scheme is a multistage explicit time stepping, for which the time step is limited by the CFL stability condition. An uncon-

ditionally stable fully implicit time stepping scheme [32, 16] has also been implemented. The second-order implicit time discretization (BDF2) requires the solution of a non-linear system of equations for all the flow variables. This can be achived by the Newton-Krylov-Schwarz approach: a Newton iteration is applied to the non-linear equations; a parallel Krylov type iterative scheme is used to solve the linear systems; the convergence of the Krylov solver is accelerated with a Schwarz type preconditioning. Two Krylov solvers have been implemented: BiCGSTAB and GMRES. A modified block incomplete LU (MBILU) preconditioner is applied on a block by block basis. Since every block has a simple Cartesian geometry, the preconditioner can be implemented very efficiently. The resulting implicit scheme requires about 20-30 times more CPU time per time step than the explicit method, but the physical time step can be 1,000 to 10,000 times larger. This implicit algorithm has a very good parallel scaling due to the Krylov scheme and the block by block application of the preconditioner.

In addition, it is possible to combine explicit and implicit time stepping. Magnetosphere simulations include large volumes where the Alfvén speed is quite low (tens of km/s) and the local CFL number allows large explicit time steps (tens of seconds to several minutes). In these regions implicit time stepping is a waste of computational resources. Since the parallel implicit technique is fundamentally block based, only those blocks where the CFL condition would limit the explicit time step to less than the selected time step (typically ~ 10 s) are treated implicitly. Needless to say, this combined explicit-implicit time stepping represents more computational challenges (such as separate load balancing of explicit and implicit blocks). Overall, this solution method seems to be a very promising option, though further work remains to be done to optimize the combined explicit-implicit technique.

7 Coupling to Other Models

The method described in the preceding sections is for the ideal and semi-relativistic MHD equations. While these equations are a suitable model for much of the solar-wind physics between Sun and Earth, other physical processes dominate in the near-Sun and near-Earth regions.

To address this, the group at University of Michigan is working to couple the parallel, adaptive MHD code described above to other models specifically tailored to their respective regimes. In particular, the following models have been incorporated:

- The Rice Convection Model (RCM), which models the dynamic behavior of the particles and the electric fields and currents of Earth's inner magnetosphere (the region inside the closed magnetic field lines). The physics of this region are complicated, because it contains overlapping particle distributions with a wide range of energies and characteristics. These different coexisting particle populations cannot be treated as a single fluid, because they all move differently. The RCM represents the particles in terms of 30-200 separate fluids. Its equations and numerical methods have been specifically designed for accurate treatment of the inner magnetosphere [15, 13, 34, 10], including the flow of electric currents along

magnetic field lines to and from the conducting ionosphere. The RCM does its primary calculations on a 2D grid on a spherical shell in the ionosphere. Values in the magnetosphere are computed by mapping out along magnetic field lines. The ionospheric grid is fine, typically about 0.5 degrees latitude in the auroral zone. The RCM computes the currents and the associated electric fields self-consistently. The essential limitation of the RCM is that it only describes the inner magnetosphere: one needs a global model for the magnetosphere's magnetic field. This can be achieved by coupling RCM to the global MHD code described above. The two codes communicate dynamically, providing a more full picture of the inner magnetosphere and the global magnetosphere than would be possible with either code alone.

- Two ionospheric models: a simple height-integrated electrostatic model [25, 28, 26], in which an empirically-driven conductance tensor is used to solve for the potential in the ionosphere, with plasma density, temperature and velocity computed by the global model; and a sophisticated model known as the thermosphere – ionosphere – electrodynamics general circulation model (TIEGCM) [24] which has been used in numerous studies of the ionosphere - thermosphere system. The TIEGCM solves for the thermospheric and ionospheric composition, temperature, and winds. It solves for mass mixing ratios of the neutral major species O_2, N_2, and O using full transport and chemistry, while the minor species $N(^2D)$, $N(^4S)$, NO, He, and Ar are obtained by assuming that they are in local equilibrium with the major species. For the ions, the O_2^+ dynamics are considered, while the species N_2^+, NO^+, and N are considered to be in local equalibrium with O_2^+. The TIEGCM is a full three-dimensional code with 5° latitude by 5° longitude by 0.5 scale height altitude cells. There are 29 pressure levels in the model such that the simulation spans from $\sim$ 95 km to 650 km in altitude. The TIEGCM/global MHD coupling has been fully implemented and tested the first results obtained with the coupled model have also been reported [27].

8 Applications

The figures in this section show results from the most ambitious simulation our group has carried out to date. The calculation tracks a coronal mass ejection (CME) from its generation at the Sun to its interaction with the Earth's magnetosphere-ionosphere system.

The earth's magnetosphere is highly sensitive to the orientation of the incoming solar wind, as shown in Figures 3 and 4. The first shows the steady solution for the Earth's magnetosphere when the incoming solar wind has a northward component to the interplanetary magnetic field (IMF); the second shows the result for a southward IMF. As can be seen, the magnetic-field topology (the white lines) and the pressure (color contours) differ dramatically between the two cases. In the northward-IMF case, the tail extends tens of Earth radii, and the pressure distribution in the equatorial plane is nearly axisymmetric. In the southward-IMF case, the magnetic-field topology is totally different, and there is a build-up of pressure in the tail.

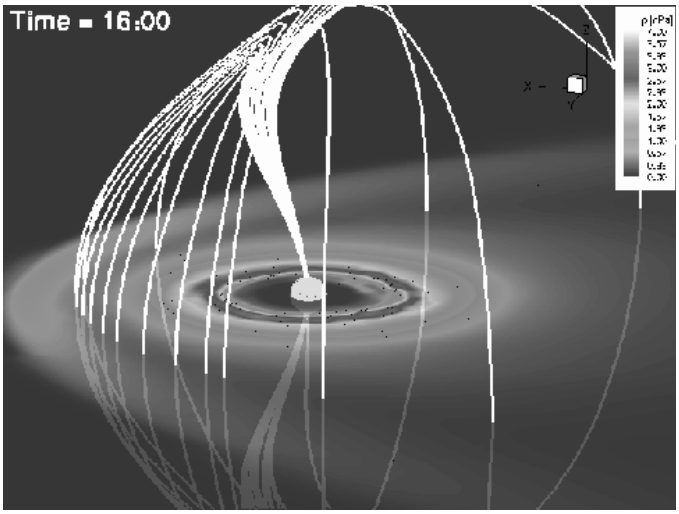


Fig. 3. Magnetospheric topology for northward interplanetary magnetic field. White lines represent magnetic-field lines; color contours represent pressure.

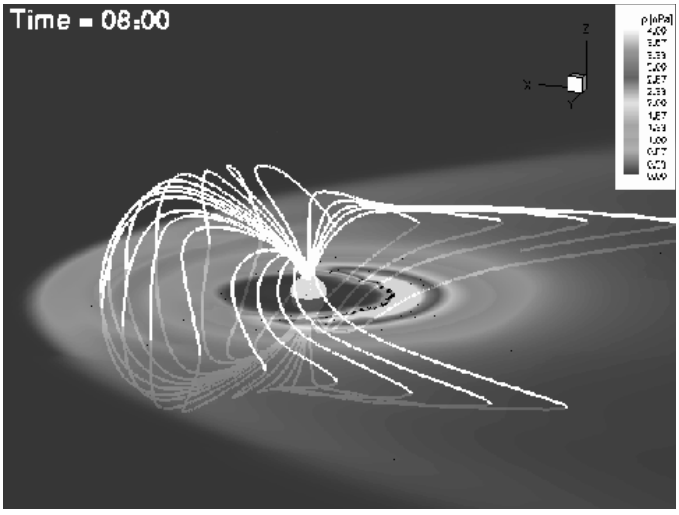


Fig. 4. Magnetospheric topology for southward interplanetary magnetic field. White lines represent magnetic-field lines; color contours represent pressure.

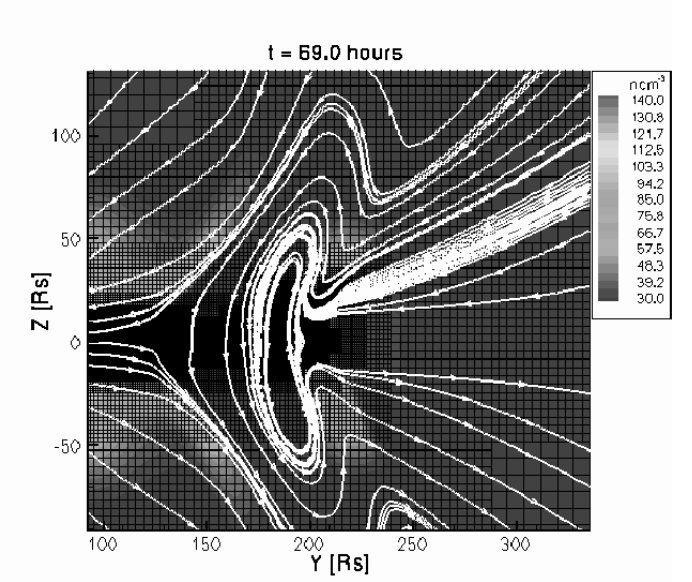


Fig. 5. Global View of Coronal Mass Ejection Calculation (t=69 hours). White lines represent projection of magnetic-field lines onto the meridional plane; color contours represent number density.

In actuality, the magnetic-field orientation of the solar wind is far from steady. In particular, if a coronal mass ejection takes place, and moves towards Earth, the Earth's magnetosphere sees a rapid change in density, pressure and magnetic-field orientation in the incoming solar wind, and responds accordingly. The calculation carried out here is for a CME that takes approximately three days to reach Earth. Up to 14 million cells were used in the calculation: the mesh-refinement criteria led to varying amounts of cells in time, and to a large range in cell size. The smallest cell was 1/32 solar radii (21,750 km); the largest was 4 solar radii (2.785×10^6 km). Figures 5 and 6 give an idea of the use of AMR in the calculation; the second plot is a 2000 times magnification of the first plot.

The calculation is begun with a steady-state solar-wind solution, the magnetic-field topology of which is shown in Figure 7. A flux rope described by a similarity solution [19] is introduced as a perturbation on this steady state, and the evolution in time of the resulting flow is computed. Figures 8 and 9 show two snapshots in time of the magnetic field (white lines) and the flow speed (color contours).

The grid, which is highly adapted along the Sun-Earth line and coarser away from it, allows the CME to be tracked all the way to Earth. There, the coupling with the magnetosphere and ionosphere codes allow the geoeffectiveness of the CME to be calculated. Figure 10 shows noon-midnight meridional cuts through the magnetosphere solutions at four times. The white lines are magnetic-field lines projected on the plane; the color contours show the pressure.

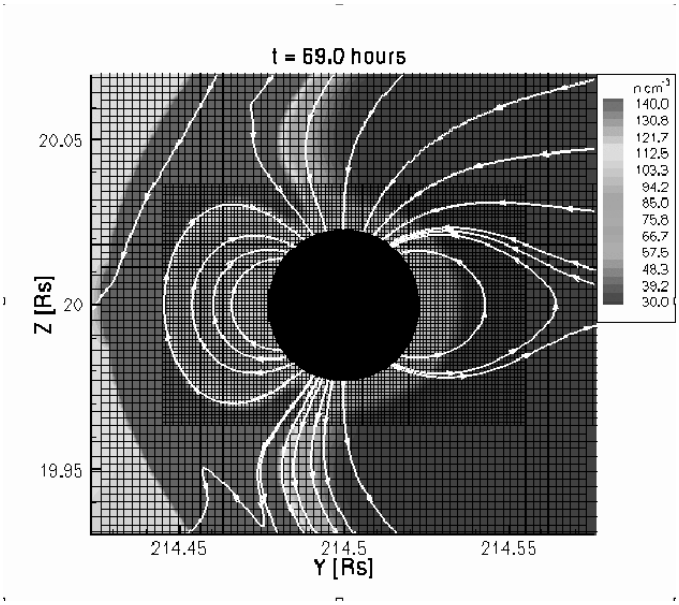


Fig. 6. Zoomed-In View of Coronal Mass Ejection Calculation. (t=69 hours). White lines represent projection of magnetic-field lines onto the meridional plane; color contours represent number density.

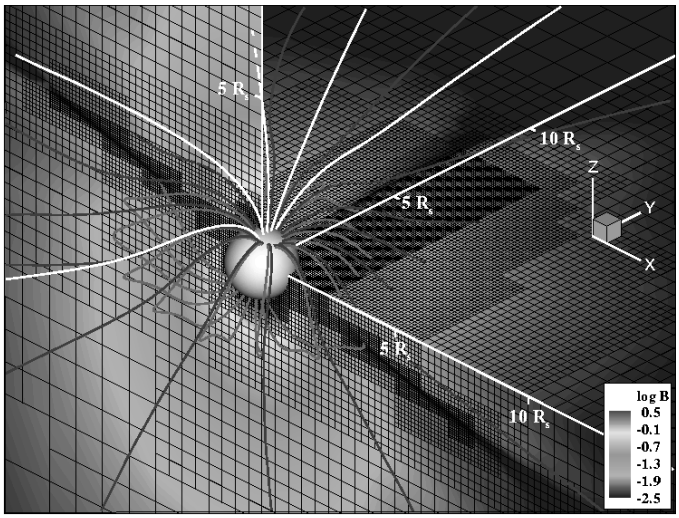


Fig. 7. Steady Corona. Magnetic-field lines and color contours of magnetic-field magnitude are shown.

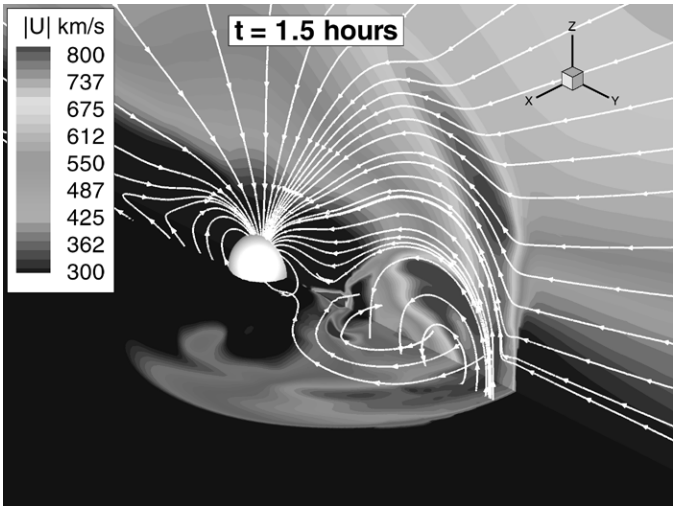


Fig. 8. Coronal Mass Ejection Evolution (t=1.5 Hours). Magnetic-field lines and color contours of flow speed are shown.

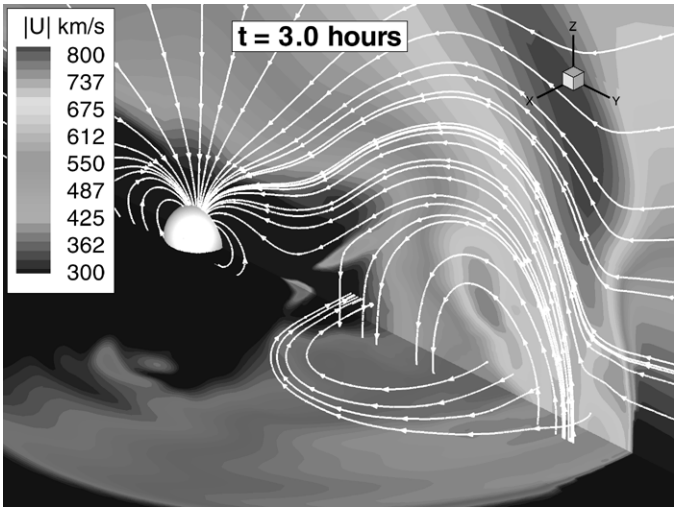


Fig. 9. Coronal Mass Ejection Evolution (t=3.0 Hours). Magnetic-field lines and color contours of flow speed are shown.

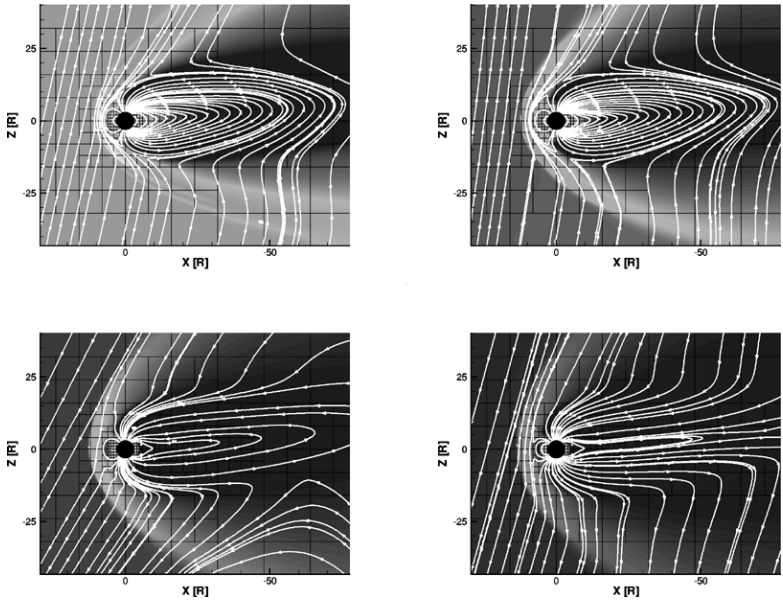


Fig. 10. Response of the Magnetosphere to the Coronal Mass Ejection. Projection of the magnetic-field lines on the meridional plane and color countours of pressure are shown at four different times.

This calculation, while still short of the ultimate goal of predictive calculations of space weather, is an important step on the way to that goal. The adaptive mesh allows a global simulation to be carried out while sufficiently resolving the near-Earth region, the near-Sun region, and the evolving CME. The parallel implementation allows the daunting computational task to be carried out in a reasonable wall-clock time. Finally, the coupling to an inner magnetosphere and ionosphere model assure that the appropriate physical processes are being modeled in each region.

Concluding Remarks

With the combination of adaptive mesh refinement, domain-decomposition parallelization, and robust finite-volume solvers, methods for solving the ideal MHD equations are developing into powerful tools for a number of applications. With attention to some issues particular to solar-wind modeling (high Alfvén speeds, strong embedded magnetic fields, pressure positivity and divergence control), these tools are becoming quite sophisticated. Much of the work to be done in improving these tools is in coupling them to solvers for regions in which semi-relativistic ideal MHD is not a sufficient model. The results presented in this paper, while preliminary, hint at the new abilities and insights that can be gained from this approach.

Acknowledgements

This work was supported by DoD MURI grant F49620-01-1-0359, NSF KDI grant NSF ATM-9980078, NSF CISE grant ACI-9876943, and NASA AISRP grant NAG5-9406. While the author list includes the primary developers of the AMR and parallel aspects of the code, a good deal of development and computationa shown here have been done by space scientists working in the group. In particular, Aaron Ridley, Chip Manchester, Ilia Roussev and KC Hansen have been instrumental in development and testing of the code. We also acknowledge the contributions of the developers of TIEGCM and RCM to the coupled magnetosphere model. In particular, the contributions of Ray Roble, Stanislav Sazykin and Richard Wolf are acknowledged and appreciated.

References

1. M. J. Berger. *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations*. PhD thesis, Stanford Univ., Stanford, Calif., January 1982.
2. M. J. Berger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.
3. M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:67–84, 1989.
4. M. J. Berger and R. J. LeVeque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. In *Proc. 9th AIAA Computational Fluid Dynamics Conference*. AIAA Paper No. 89-1930, Buffalo, NY, June 1989.
5. M. J. Berger and S. Saltzman. AMR on the CME-2. *Appl. Numer. Math.*, 14:239–253, 1994.
6. J. P. Boris. A physically motivated solution of the Alfvén problem. Technical Report NRL Memorandum Report 2167, Naval Research Laboratory, Washington, D.C., 1970.
7. J.U. Brackbill and D.C. Barnes. The effect of nonzero $\nabla \cdot \mathbf{B}$ on the numerical solution of the magnetohydrodynamic equations. *J. Comput. Phys.*, 35:426–430, 1980.
8. W. Dai and P. R. Woodward. A simple finite difference scheme for multidimensional magnetohydrodynamic equations. *J. Comput. Phys.*, 142:331, 1998.
9. A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, and M. Wesenberg. Hyperbolic divergence cleaning for the MHD equations. *J. Comput. Phys.*, 00:00–00, 2001. submitted.
10. G. M. Erickson, R. W. Spiro, and R. A. Wolf. The physics of the harang discontinuity. *J. Geophys. Res.*, 96:1633–1645, 1991.
11. S. K. Godunov. Symmetric form of the equations of magnetohydrodynamics (in Russian). In *Numerical Methods for Mechanics of Continuum Medium*, volume 1, pages 26–34. Siberian Branch of USSR Acad. of Sci., Novosibirsk, 1972.
12. T. I. Gombosi, G. Tóth, D. L. De Zeeuw, K. C. Hansen, K. Kabin, and K. G. Powell. Semi-relativistic magnetohydrodynamics and physics-based convergence acceleration. *J. Comput. Phys.*, 177:176–205, 2002.
13. M. Harel, R. A. Wolf, P. H. Reiff, R. W. Spiro, W. J. Burke, F. J. Rich, and M. Smiddy. Quantitative simulation of a magnetospheric substorm 1, Model logic and overview. *J. Geophys. Res.*, 86:2217–2241, 1981.

14. A. Harten. High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 49:357–393, 1983.
15. R. K. Jaggi and R. A. Wolf. Self-consistent calculation of the motion of a sheet of ions in the magnetosphere. *J. Geophys. Res.*, 78:2842, 1973.
16. R. Keppens, G. Tóth, M. A. Botchev, and A. van der Ploeg. Implicit and semi-implicit schemes: Algorithms. *Int. J. for Num. Meth. in Fluids*, 30:335–352, 1999.
17. T. J. Linde. *A Three-Dimensional Adaptive Multifluid MHD Model of the Heliosphere*. PhD thesis, Univ. of Mich., Ann Arbor, May 1998.
18. P. Londrillo and L. Del Aanna. High-order upwind schemes for multidimensional magnetohydrodynamics. *Astrophys. J.*, 530:508–524, 2000.
19. W. B. Manchester, T. I. Gombosi, I. Roussev, D. L. De Zeeuw, I. V. Sokolov, and K. G. Powell. Thre-dimensional mhd simulation of a flux-rope driven cme. *J. Geophys. Res.*, 109, 2004.
20. K. G. Powell. An approximate Riemann solver for magnetohydrodynamics (that works in more than one dimension). Technical Report 94-24, Inst. for Comput. Appl. in Sci. and Eng., NASA Langley Space Flight Center, Hampton, Va., 1994.
21. K. G. Powell, P. L. Roe, T. J. Linde, T. I. Gombosi, and D. L. De Zeeuw. A solution-adaptive upwind scheme for ideal magnetohydrodynamics. *J. Comput. Phys.*, 154(2):284–309, September 1999.
22. J. J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics*. PhD thesis, Cranfield Inst. of Technol., Cranfield, England, January 1991.
23. J. J. Quirk and U. R. Hanebutte. A parallel adaptive mesh refinement algorithm. Technical Report 93-63, ICASE, August 1993.
24. A. D. Richmond, E. C. Ridley, and R. G. Roble. A thermosphere/ionosphere general circulation model with coupled electrodynamics. *Geophys. Res. Lett.*, 19:601–604, 1992.
25. A. J. Ridley, D. L. De Zeeuw, T. I. Gombosi, and K. G. Powell. Using steady-state MHD results to predict the global state of the magnetosphere-ionosphere system. *J. Geophys. Res.*, 106:30,067–30,076, 2001.
26. A. J. Ridley, T. I. Gombosi, and D. L. De Zeeuw. Ionospheric control of the magnetosphere: Conductance. *J. Geophys. Res.*, 00:00–00, 2002. in preparation.
27. A. J. Ridley, T. I. Gombosi, D. L. De Zeeuw, C. R. Clauer, and A. D. Richmond. Ionospheric control of the magnetosphere: Neutral winds. *J. Geophys. Res.*, 00:00–00, 2002. in preparation.
28. A. J. Ridley, K. C. Hansen, G. Tóth, D. L. De Zueew, T. I. Gombosi, and K. G. Powell. University of Michigan MHD results of the GGCM metrics challenge. *J. Geophys. Res.*, 107:000–000, 2002. in press.
29. P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.
30. P. L. Roe and D. S. Balsara. Notes on the eigensystem of magnetohydrodynamics. *SIAM J. Appl. Math.*, 56(1):57–67, February 1996.
31. G. Tóth. The $\nabla \cdot \mathbf{B}$ constraint in shock capturing magnetohydrodynamic codes. *J. Comput. Phys.*, 161:605–652, 2000.
32. G. Tóth, R. Keppens, and M. A. Botchev. Implicit and semi-implicit schemes in the Versatile Advection Code: Numerical tests. *Astron. Astrophys.*, 332:1159–1170, 1998.
33. B. van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *J. Comput. Phys.*, 32:101–136, 1979.
34. R. A. Wolf. The quasi-static (slow-flow) region of the magnetosphere. In R. L. Carovillano and J. M. Forbes, editors, *Solar Terrestrial Physics*, pages 303–368. D. Reidel Publishing, Hingham, MA, 1983.

Adaptive Mesh Refinement for MHD Fusion Applications

R. Samtaney[1], S. C. Jardin[1], P. Colella[2] and D. F. Martin[2]

[1] Princeton Plasma Physics Laboratory, Princeton, NJ 08543 samtaney,jardin@pppl.gov
[2] Lawrence Berkeley National Laboratory, Berkeley, CA 94720
pcolella,dmartin@lbl.gov

Summary. We present results from adaptive mesh refinement (AMR) simulations of two MHD applications relevant to magnetic fusion. The applications are: pellet injection in tokamaks (PI) and magnetic reconnection (MR). For PI, AMR is essential to provide the resolution required to simulate realistic pellet sizes relative to device dimensions (typical ratios are $O(10^{-3})$). We present results from 3D AMR simulations in a tokamak geometry with emphasis on differences between the low-field and high-field side pellet launches. For MR, We present results from 2D AMR simulations in which both the outer "inviscid" region and the inner "resistive" region are well-resolved. AMR provides an efficient means of resolving the near-singular current sheets in the inner region where the actual breaking and reconnection of the magnetic field lines takes place. The numerical method developed is an explicit unsplit upwinding treatment of the 8-wave formulation, coupled with a MAC projection method to enforce the solenoidal property of the magnetic field. The Chombo framework is used for AMR.

1 Introduction

Magneto-hydrodynamic fusion applications often exhibit a wide range of spatial and temporal scales. In this paper, we focus on two MHD applications relevant to magnetic confinement fusion. The applications chosen are: (a) pellet injection (PI) in tokamaks and (b) magnetic reconnection (MR). Both applications benefit from adaptive mesh refinement (AMR) in the sense that we are able to resolve small features while still being able to compute MHD flows in a practical manner.

PI: Injecting small pellets of frozen hydrogen into a tokamak is a proven method of fueling. Experimentally, it is known that the density distribution, after the pellet ablates upon encountering the high temperatures in a tokamak, is not consistent with the distribution inferred from assuming that the ablated material remains on the flux surfaces where the ablation occurred. The subsequent redistribution of mass is believed to be due to anomalous MHD processes. The mass redistribution is observed to be a sensitive function of the angle (with respect to the mid-plane) in which the pellet is injected [1, 2]. It is this phenomenon which we seek to explain.

MR: Magnetic reconnection refers to the breaking and reconnecting of oppositely directed magnetic field lines in a plasma. In the process, magnetic field energy is converted to plasma kinetic and thermal energy. MR occurs in many contexts: for example, in the sawtooth-like oscillations observed in the operation of a tokamak, and in solar coronal events. In general, in MR, two regions are distinguished: an outer "inviscid" region and an inner "resistive" region whose width scales with $\eta^{\frac{1}{2}}$, where the actual breaking and reconnecting of the magnetic field lines takes place. In this paper, we present results from simulations of MR in an idealized canonical two dimensional setting in which both the inner and outer regions are well-resolved. AMR is an obvious computational methodology to resolve the near-singular current sheets.

2 Pellet Injection in Tokamaks

2.1 Physical Problem

The physical problem we are dealing with involves the injection of frozen fuel pellets into a tokamak. The physical processes are broadly distinguished into the following two stages. The first stage is the ablation of mass at the pellet surface due to the high temperature background plasma encountered by the pellet. The ablated pellet mass, which is a neutral gas, is rapidly heated by electrons and ionizes to form plasma. The second stage is the redistribution of the ablated pellet material by free streaming along the magnetic field lines and by anomalous MHD processes which cause mass flow across field lines and flux surfaces. The pellet ablation phenomenon of the first stage is considered well-understood [3, 4], and as such we use existing ablation models. The thrust of the work described here is an accurate and efficient simulation of the second phase.

2.2 Mathematical Model

Our mathematical model consists of single fluid MHD equations with source terms in the continuity equation to model the mass injected into the system by the pellet, and source (sink) terms in the energy equations to model electron heating and corresponding cooling on flux surfaces. The equations are written below.

$$\frac{\partial U}{\partial t} + \frac{\partial F_j(U)}{\partial x_j} = \frac{\partial F_{v,j}(U)}{\partial x_j} + S_T(U) + S_{\nabla\cdot\mathbf{B}}(U) + S_{pellet}(U), \tag{1}$$

where the solution vector $U \equiv U(x_1, x_2, x_3, t) \equiv U(R, z, R_0\phi, t)$ is $U = \{\rho, \rho u_i, B_i, e\}^T$, and the flux vector $F_j(U)$ is given by

$$F_j(U) = \left\{ \begin{array}{c} \rho u_j \\ \rho u_i u_j + p_t \delta_{ij} - B_i B_j + B_T B_3 \delta_{ij} - B_i B_T \delta_{3j} - B_j B_T \delta_{i3} \\ u_j B_i - u_i B_j + B_T \delta_{i3} u_j - B_T \delta_{3j} u_i \\ (e + p + \frac{1}{2} B_k B_k) u_j - B_j (B_k u_k) + B_T B_3 u_j - (B_k u_k) B_T \delta_{3j} \end{array} \right\}. \tag{2}$$

In the above equations, R, z, ϕ, are the radial, axial and toroidal coordinates, $R0$ is the major radius, ρ is the density, u_i is the velocity, B_i is the magnetic field, p and p_t are the pressure and total pressure, respectively, and e is the total energy per unit volume of the plasma. For numerical stability and robustness, we have subtracted out the equilibrium toroidal component of the initial equilibrium magnetic field, $B_T(x_i,0) \equiv g_0/R$. These equations are closed by the perfect gas equation of state,

$$e = \frac{p}{\gamma - 1} + \frac{\rho}{2} u_k u_k + \frac{1}{2} B_k B_k, \tag{3}$$

which we note does not include the contribution $1/2B_T^2$. The flux vector $F_{v,j}(U)$ corresponds to the diffusive resistivity/viscosity terms and is omitted in the interest of brevity. The toroidal geometry terms are modeled in the source terms as

$$S_T(U) = -\frac{1}{R} \begin{pmatrix} \rho u_R \\ \rho u_R^2 - \rho u_\phi^2 - B_R^2 + B_\phi^2 + 2B_\phi B_T \\ \rho u_R u_Z - B_R B_z \\ 2\rho u_R u_\phi - 2B_R B_\phi - 2B_R B_T \\ 0 \\ u_R B_z - B_R u_z \\ 0 \\ (e + p_t) u_R - (\mathbf{B} \cdot \mathbf{u}) B_R + B_T B_\phi u_R \end{pmatrix} + \left(\frac{1}{R_0} - \frac{1}{R}\right) \frac{\partial F_\phi}{\partial \phi}. \tag{4}$$

For a large aspect ratio tokamak, $S_T(U)$ is small but it contains essential toroidal effects which cause the in-out asymmetry discussed in Section 2.5. The source terms $S_{\nabla \cdot \mathbf{B}}(U)$, written below,

$$S_{\nabla \cdot \mathbf{B}}(U) = -\nabla \cdot \mathbf{B}(\{0, B_R, B_\phi, B_z, u_R, u_z, u_\phi, u_z, (B \cdot u)\}^T), \tag{5}$$

are included because we use the symmetrization procedure of Godunov [5] which leads to the 8-wave formulation. This formulation was also used by Powell et al. [6] in their AMR implementation of ideal MHD. Finally, the source terms $S_{pellet} = \{S_n/n_0, 0, 0, 0, 0, 0, 0, S_e/n_0\}^T$, where n_0 is some reference number density, correspond to the mass source and energy source/sink terms, and are described next.

2.3 Pellet Ablation Model

In the present model, the pellet is described by a sphere of frozen molecular hydrogen of radius r_p. The trajectory $x_p(x_i,t)$ of the pellet is prescribed with a given initial location $x_{p0} \equiv x_p(x_i,0)$ and constant velocity u_p. The density source term arises from the ablation of the pellet and is written in terms of number density, (i.e., atoms per unit volume per unit time) as

$$S_n = \dot{N} \delta(x - x_p), \tag{6}$$

where the delta function is approximated as a Gaussian distribution centered over the pellet with a characteristic size equal to $10r_p$. The ablation rate of the pellet,

originally derived by Parks and Turnbull [3] and modified for hydrogen pellets by Kuteev [7] is given below (in atoms/sec)

$$\dot{N} = -4\pi r_p^2 \frac{dr_p}{dt} 2n_m = 1.12 \times 10^{16} n_e^{0.333} T_e^{1.64} r_p^{1.33} M_i^{-0.333}, \tag{7}$$

where n_e is the background plasma density in cm^{-3}, T_e is the background plasma electron temperature in eV, M_i is the atomic mass number in atomic units and $n_m = 2.63 \times 10^{22}/cm^3$ is the molecular density of frozen hydrogen. A useful approximation which eliminates the electron timescale from the problem is to consider the electron heat flux as being instantaneous compared to the other processes being computed. The time-asymptotic effect of the large electron heat flux is to make the temperature uniform along field lines, i.e., $T \equiv T(\psi)$. Thus, for single fluid equations, the temperature $T(\psi)$ in the volume V_ψ between flux surfaces ψ and $\psi + d\psi$ will equilibrate as the density changes while still conserving energy in the volume V_ψ. This leads to the following energy source terms in the energy equation

$$S_e = 3\left(S_n T(\psi) + n\dot{T}(\psi)\right). \tag{8}$$

The first term in S_e corresponds to the localized increase in energy due to the heating of the ablated pellet mass, while the second term corresponds to a global adiabatic cooling of the entire flux surface. In practice, we compute the contribution due to the second term by separately solving a 1D model for the pellet injection assuming only classical processes are present. We then use table lookup and interpolation to compute the term $\dot{T}(\psi)$ in our 3D AMR simulation.

2.4 Initial and Boundary Conditions

The initial condition is a static equilibrium state. The initial magnetic field is written in terms of two function $\psi(R,z)$ and $g(R,z)$, i.e.,

$$B = \frac{1}{R}(\hat{\phi} \times \nabla\psi + g\hat{\phi}). \tag{9}$$

These functions satisfy the Grad-Shafranov equation,

$$R\frac{\partial}{\partial R}\frac{1}{R}\frac{\partial \psi}{\partial R} + \frac{\partial^2 \psi}{\partial z^2} + R^2 \frac{dp}{d\psi} + g\frac{dg}{d\psi} = 0, \tag{10}$$

where $p \equiv p(\psi)$ and $g \equiv g(\psi)$. For a torus with rectangular cross-section of radial extent $2a$ and axial extent of $2b = 2\kappa a$ we may write $\psi(R,z) = f(R)\cos(\pi z/\kappa a)$. Further, with $g(R,z) = g_0 = constant$ we get

$$R\frac{d}{dR}\left(\frac{1}{R}\frac{df}{dR}\right) + \left(\frac{R_0\pi}{a}\right)^2 \left(\alpha R^2 - \frac{1}{4\kappa^2}\right) f = 0 \tag{11}$$

which permits a Frobenius-type series solution. The value of α is determined by imposing the boundary conditions $\psi = 0$. The pressure is written as $p = \bar{p} + p_0\psi^2$

where $\bar{p}$ is a small background pressure to avoid zero ion-acoustic speeds and $p_0 = \alpha\pi^2/(2a^2R_0^2)$. The toroidal field function is $g_0 = R_0\alpha\pi^2|\psi|_{max}q_0/(2ab)$, where $q_0(\approx 1)$ is the on-axis safety factor. Boundary conditions imposed are perfectly conducting walls in the radial/axial directions and periodic in the toroidal direction. In our simulations we use $\kappa = 1$, $a/R0 = \pi/9$, for which $\alpha = 0.481509$.

2.5 Simulation Results

In this section, we present preliminary results from early to intermediate stages of pellet injection. The results discussed here correspond to a midpoint toroidal field of 0.23 T, $\beta \approx 0.1$, and a pellet of 1 *mm* radius moving radially with a velocity of 3200 m/s in a tokamak with minor radius of $a = 0.26$ m. Two cases are discussed: one in which the pellet is initialized on the high field side (HFS or the so-called inside launch case) and the other in which the pellet is injected from the low field side (LFS or the "outside" launch case). Because the temperature of the plasma is low near the edges of the tokamak, we initialize the pellet at some radial distance inside the tokamak. This is merely to save computational effort and have interesting dynamics take place relatively quickly. In both the LFS and the HFS cases, the initial location of the pellet is on the same flux surface so that the pellet encounters the same initial temperature. Based on preliminary tests which suggested that the energy sink term provides only a small contribution, but is nonetheless computationally expensive to evaluate and occasionally leads to noisy solutions, we omitted the sink term in the results presented here.

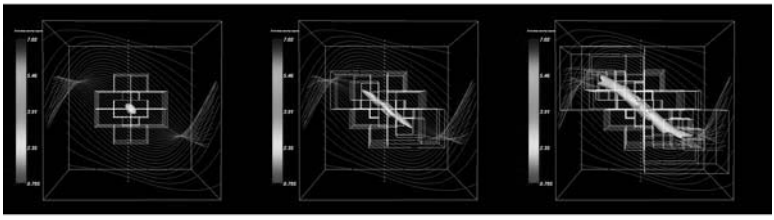


Fig. 1. Density isosurface ($\rho = 2$) for a HFS pellet launch. (a) $t = 2$ (b) $t = 20$ (c) $t = 60$ viewed radially inwards. The magnetic field lines are shown in red. The box outlines depict the meshes. Time is normalized by the Alfvèn wave transit time. (Note that although the domain is a torus the visualizations are presented in a cube)

Fig. 1 shows a density isosurface, viewed radially inwards, at times $t = 2, 20, 60$ (time is normalized by the Alfvén time) for the HFS case. The outlines of the various meshes in the calculation are also shown in Fig. 1. At $t = 2$ the pellet ablated mass is roughly in the shape of an ellipsoid with its major axis aligned along the magnetic field lines. The pellet cloud is a localized region of high β with the dominant mass motion being along the magnetic field lines. As time progresses, the ablated mass

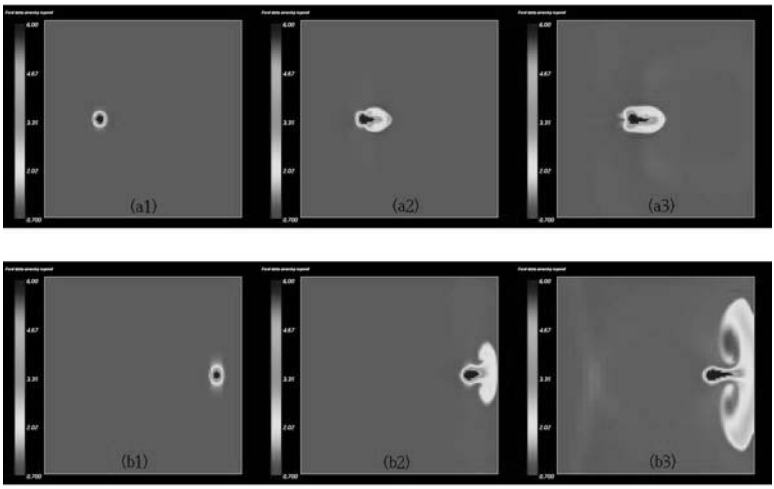


Fig. 2. Density field in a poloidal cross-section. (a1) HFS $t = 2$, (a2) HFS $t = 20$, (a3) HFS $t = 60$, (b1) LFS $t = 2$, and (b2) LFS $t = 20$, and (b3) LFS $t = 60$.

moves parallel to the magnetic field at speeds of about one-third of the local acoustic speed. In addition to the "classical" parallel transport there is clear evidence of "anomalous" transport perpendicular to the flux surfaces.

We now examine this phenomenon in more detail and compare and contrast between HFS and LFS pellet launches. Fig. 2 shows poloidal slices at the mean toroidal pellet location for the HFS and LFS cases. At time $t = 2$ we observe the ablated mass in a cloud around the mean pellet position. At later times ($t = 20, 60$), the pellet ablated mass has a significant outward radial displacement compared to the mean pellet location.

The LFS case shows a dramatic turning around of the mass due to the zero mass flux boundary conditions (Fig. 2-b3) It is conjectured that an outflow boundary condition would lead to a substantial loss of the ablated mass and thus poor fueling efficiency in the LFS case. The observed outward displacement implies that HFS launches are more favorable for refueling tokamaks as opposed to the LFS launches, consistent with observed behavior in experiments [8, 1]. We may reconcile this seemingly "anomalous" transport by appealing to the model by Parks [9] which notes that magnetic curvature and ∇B-induced charged particle drifts cause a local separation of charges in the pellet cloud. This leads to an axially-oriented electric field, and so the $E \times B$ drift is radially outward in both the LFS and the HFS cases.

It is instructive to examine the flow pattern of the perpendicular drift velocity $v_\perp = E \times B/|B|^2$. The radial component of $v_\perp$ is the dominant one and is shown in Fig. 3 in a poloidal slice. For the HFS case, in Fig. 3(a), there is a dominant outward radial $v_\perp$ carrying the bulk of the pellet mass outward. This is flanked on either side in the axial direction by inward radial motion resulting in a nearly incompressible flow pattern. So the simple picture of only outward radial $v_\perp$ drift is augmented by this

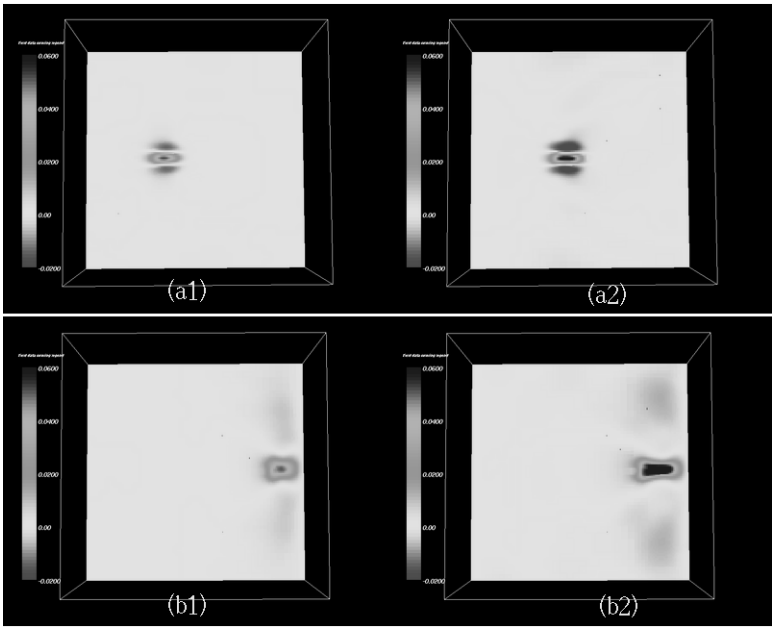


Fig. 3. Perpendicular drift velocity $v_\perp$ in a Poloidal cross-section. (a1) HFS $t = 20$, (a2) HFS $t = 60$, (b1) LFS $t = 20$, and (b2) LFS $t = 60$.

somewhat smaller turning around of the mass which leads to the mushroom-shaped structure in the poloidal plane.

For the LFS case too, the outward radial $E \times B$ drift grows with time and is clearly seen in Fig. 3(b). The perpendicular transport of the ablated mass brings into question some of the assumptions made in the earlier section. In calculation of the ablation rate, we assumed that the ablated mass is heated instantaneously to the flux surface temperature. However, the motion of the ablated mass radially outwards in the HFS case means that the temperature the pellet encounters will actually be smaller than that assumed. Furthermore, the energy sink term in the equations, which are based on a one-dimensional parallel transport model, will need to be modified.

3 Magnetic Reconnection in 2D

3.1 Physical Problem and Mathematical Model

We are interested in simulating magnetic reconnection in two dimensions in which we resolve both the inner reconnection region and the outer global region with appropriate precision. Two *compressible* plasma columns are allowed to merge and the reconnection rate is calculated. Our mathematical model for this is the single fluid resistive MHD equations written below in conservation form:

$$\frac{\partial U}{\partial t} + \frac{\partial F_j(U)}{\partial x_j} = \frac{\partial F_{v,j}(U)}{\partial x_j}, \tag{12}$$

where the solution vector $U \equiv U(x,y,t)$ is,

$$U = \{\rho, \rho u_i, B_i, e\}^T,$$

and the flux vectors $F_j(U)$ and $F_{v,j}(U)$ are given by

$$F_j(U) = \Big\{ \rho u_j, \rho u_i u_j + (p + \frac{1}{2} B_k B_k)\delta_{ij} - B_i B_j, u_j B_i - u_i B_j, \\ (e + p + \frac{1}{2} B_k B_k) u_i - B_i (B_k u_k) \Big\}^T,$$

and

$$F_{v,j}(U) = \Big\{ 0, Re^{-1}\tau_{ij}, S^{-1}(\eta \frac{\partial B_i}{\partial x_j} - \eta \frac{\partial B_j}{\partial x_i}), \\ Re^{-1}\tau_{ij} u_i + Pe^{-1}\kappa \frac{\partial T}{\partial x_j} + \eta(\frac{1}{2}\frac{\partial B_k B_k}{\partial x_j} - Bi \frac{\partial B_j}{\partial x_i}) \Big\}^T,$$

where Re, S, Pe are, respectively, the Reynolds number, Lundquist number and Peclet number. The non-dimensional plasma properties are: the viscosity μ, the conductivity κ, and the resistivity η. The stress tensor is related to the strain as

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3}\mu \frac{\partial u_k}{\partial x_k}\delta_{ij},$$

Note that we have included equations for the perpendicular magnetic field and momentum above (some times referred to as a 2.5D model).

3.2 Initial and Boundary Conditions

The initial condition is comprised of an MHD equilibrium state with uniform density ($\rho = 1$) and uniform pressure ($p = 0.2$). The magnetic flux is $\psi(x,y,0) = -\cos k_x x \sin k_y y$ and the perpendicular magnetic field is $B_z(x,y,0) = -(k_x^2 + k_y^2)^{\frac{1}{2}} \psi(x,y,0)$, with $k_x = 3\pi/2$, $k_y = 2\pi$, defined on a domain $[-1:1] \times [0:1]$. The boundary conditions are reflecting at the top/bottom, and open at left/right. Furthermore, the top/bottom boundaries are perfectly conducting with specified Dirichlet conditions for temperature. The non-dimensional resistivity is a nonlinear function given by

$$\eta = \eta^- + (\eta^+ - \eta^-)\left[1 - \exp(-177.69\psi^2)\right] \times max(0, -sign(\psi))$$

where $\eta^- = 1$, $\eta^+ = 0.1/S$. The sole purpose of this nonlinear resistivity is to annihilate the middle flux tube and set up appropriate conditions which lead to the reconnection of the outer flux tubes. The initial conditions and the nonlinear resistivity are similar to those used by Breslau [10] in studies of magnetic reconnection.

3.3 AMR Simulations Results

At relatively small values of Lundquist number, $S = 10^3$, the entire process can be divided into three phases: (a) decay of middle flux tube, followed by (b) reconnection, and (c) decay of the reconnection layer. This is shown in Figure 4. We now discuss some recent observations at $S = 10^4$. A time sequence of the vertical component of the magnetic field is shown in Figure 5. As the two flux tubes move in closer we observe an intensification of the perpendicular current. During reconnection, the current layer becomes unstable and results in the ejection of high pressure plasma with the ejection direction alternating between the top and bottom (See Figure 6). The peak reconnection rate is plotted in Figure 7 where we see that, for $S = 10^4$, before the current layer becomes unstable, the peak reconnection rate is consistent with the expected Sweet-Parker scaling, but that after the onset of the current layer instability the peak reconnection rate is higher than the theoretical reconnection rate.

4 Numerical Method

In this section, we focus on the evaluation of the hyperbolic flux terms, $F_j(U)$, in Eqn. (1) and Eqn. (12). We use a finite volume technique wherein each variable is stored at the cell center. The numerical fluxes of conserved quantities are obtained at the cell faces using a combination of the 8-wave formulation [5] and unsplit upwinding [11, 12]. We define a vector of "primitive" variables $W = \{\rho, u_i, B_i, p\}^T$. Given the conserved quantities and all the source terms, i.e.,U_i^n, S_i^n (in this notation, i is a 3-tuple corresponding to the three dimensions), we want to compute a second-order accurate estimate of the fluxes: $F^{n+\frac{1}{2}}_{i+\frac{1}{2}e^d}$ (d indicates the d-th direction, $0 \le d < 2$ for PI and $0 \le d < 1$ for MR). The first step is to compute W_i^n in each cell, followed by fitting a linear profile in each cell subject to slope limiting. We then extrapolate the primitive and conserved variables at the cell faces using the normal derivative terms and the source terms at the cell centers, as follows.

$$W_{i,\pm,d} = W_i^n + \frac{1}{2}(\pm I - A_i^d \frac{\Delta t}{h}) P_\pm \Delta^d W_i, \;\; U_{i,\pm,d} = U(W_{i,\pm,d}) + \frac{\Delta t}{2} S_i^n \qquad (13)$$

where $A_i^d = (\nabla_W U \nabla_W F^d)(W_i)$ and $P_\pm(W) = \sum_{\pm\lambda_k > 0}(l_k \cdot W) r_k$, and $\Delta^d W_i$ is the undivided but limited slope in the $d-th$ direction. The eigenvalues, and left and right eigenvectors of A_i^d are λ_k, l_k, and r_k, respectively with $k = 1 \cdots 8$ in the eight-wave formulation (see Powell et al. [6] for the left and right eigenvectors). We compute corrections to $U_{i,\pm,d}$ corresponding to one set of transverse derivatives appropriate to obtain $(1,1,1)$ diagonal coupling:

$$U_{i,\pm,d_1,d_2} = U_{i,\pm,d_1} - \frac{\Delta t}{3h}(F^{1\mathbf{D}}_{i+\frac{1}{2}e^{d_2}} - F^{1\mathbf{D}}_{i-\frac{1}{2}e^{d_2}}), \;\; d_1 \neq d_2, 0 \le d_1, d_2 < 3 \qquad (14)$$

where $F^{1\mathbf{D}}_{i\pm\frac{1}{2}e^d} = RP(U_{i,+,d}, U_{i+e^d,-,d})$. The notation $F = RP(U_L, U_R)$ implies that the flux F is evaluated by solving a linearized Riemann problem using U_L and U_R as

left and right states, respectively. We next compute final corrections to $U_{i,\pm,d}$ due to transverse derivatives:

$$U^{n+\frac{1}{2}}_{i,\pm,d} = U_{i,\pm,d} - \frac{\Delta t}{2h}(F_{i+\frac{1}{2}e^{d_1},d_2} - F_{i-\frac{1}{2}e^{d_1},d_2}), \tag{15}$$

where $F_{i+\frac{1}{2}e^{d_1},d_2} = RP(U_{i,+,d_1,d_2}, U_{i+e^{d_1},+,d_1,d_2})$, and $0 \le d < 3; d_1 \ne d_2 \ne d$. At this stage, we solve another Riemann problem at the cell faces using $U^{n+\frac{1}{2}}_{i,+,d}$ and $U^{n+\frac{1}{2}}_{i+e^d,-,d}$ as the left and right states, respectively. The magnetic field obtained from the solution to the Riemann problem at $n+\frac{1}{2}$ at the cell faces is not guaranteed to be divergence free. We enforce the solenoidal property of the magnetic field by a MAC projection, using **B** at the cell faces to obtain a cell-centered monopole charge density. A Poisson solver is used to find a scalar field satisfying $\nabla^2\chi = \nabla\cdot\mathbf{B}$ with Neumann boundary conditions in the radial/axial directions and periodic in the toroidal direction. The magnetic field at the cell faces is then corrected according to $B^{n+\frac{1}{2}}_{i+\frac{1}{2}e^d} = B^{n+\frac{1}{2}}_{i+\frac{1}{2}e^d} - \nabla\chi$. Finally the fluxes at cell faces are obtained as $F^{n+\frac{1}{2}}_{i+\frac{1}{2}e^d} = F(U^{n+\frac{1}{2}}_{i+\frac{1}{2}e^d})$ and the conserved quantities at the cell centers are updated using these fluxes. The Poisson equation in the projection step above is cast in a residual-correction form and solved using a multi-grid technique on each level in the AMR hierarchy. The residual smoothing is a Gauss-Seidel relaxation procedure with red-black ordering. When meshes cannot be coarsened any further, the Poisson solve is taken to convergence using a bottom-smoother which is a biconjugate gradient solver. We implemented the above method into the *Chombo* framework and have developed a second-order in space and time, adaptive parallel MHD code. Chombo is a collection of C++ libraries for implementing block-structured AMR finite difference calculations [13]. Particular care is taken in implementing coarse-fine interface interpolations of appropriate order to ensure second-order accuracy. Furthermore, conservation at coarse-fine interfaces is maintained by flux-refluxing. This leads to a non-zero cell-centered $\nabla\cdot\mathbf{B}$ in coarse cells which are adjacent to coarse-fine boundaries, which being a set of codimension one does not significantly affect the accuracy of the solution.

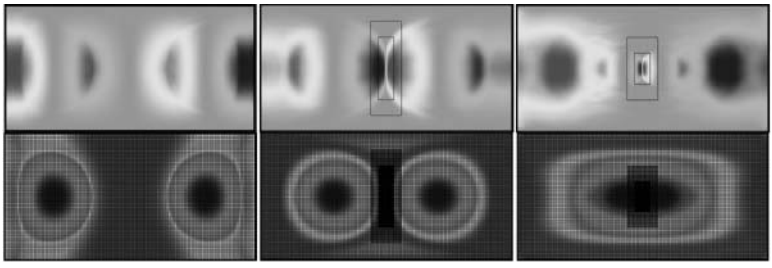


Fig. 4. Time sequence (left to right) of reconnection at $S = 10^3$. The top row is the vertical magnetic field in which black bounding boxes depict the meshes. The bottom row shows the perpendicular magnetic field with AMR meshes superimposed.

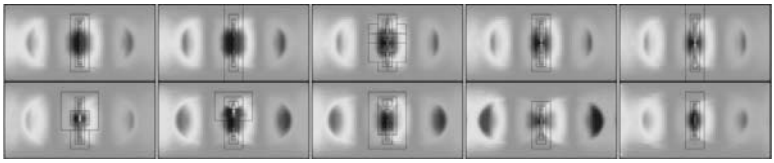


Fig. 5. Time sequence of the y-component of the magnetic field. The black bounding boxes depict the meshes in this AMR hierarchy. In this simulation 6 AMR levels were employed.

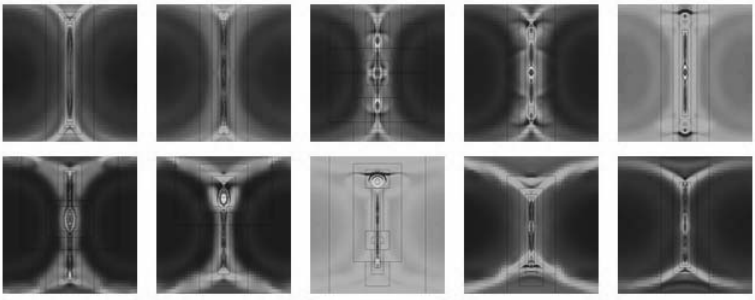


Fig. 6. Time sequence of the perpendicular current corresponding to the previous figure. The current layer is unstable which causes ejection of high pressure plasma from the reconnection layer.

5 Conclusion

In this paper, we presented a numerical method which is based on an unsplit upwinding method coupled with the eight-wave formulation. A MAC-projection scheme is implemented to enforce the solenoidal property of the magnetic field. This projection requires the solution of a Poisson equation which is solved using a multi-grid technique. A pellet injection model was implemented as a source term in the density equations and corresponding energy sources and sinks in the energy equation. AMR simulations of the pellet injection process were carried out for the inside and outside launch cases. Preliminary studies indicate that AMR provides a speed-up exceeding two orders of magnitude over corresponding uniform mesh simulations essential to accurately resolve the physical processes involved in pellet injection. AMR is an effective way of achieving computational efficiency in detailed and resolved simulations of the pellet injection process. It was observed that the pellet ablated mass is dominantly transported along magnetic field lines but that a $E \times B$ drift causes a significant outward radial motion of the pellet cloud in both the LFS and HFS cases. A high resolution numerical simulation is a viable method of computing the relative importance of these two competing phenomena for redistributing the pellet mass.

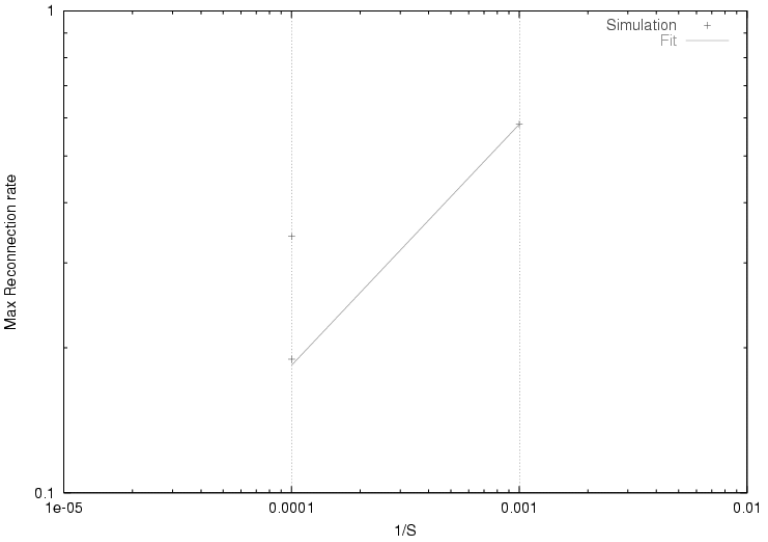


Fig. 7. Maximum reconnection rate measured at the center point. For $S = 10^4$ the maximum reconnection rate before the onset of the current layer instability falls on the Sweet-Parker scaling curve which lies above this after the current layer becomes unstable.

In magnetic reconnection, it was found that current layer becomes unstable at the higher Lundquist number investigated. The peak reconnection rate during this phenomenon is larger than the theoretical reconnection rates. It is still an open question whether such a mechanism is partially responsible for observed reconnection rates which are faster than those suggested by the Sweet-Parker scaling. In our MR studies, we observed speed-ups ranging from 6 to 32, due to AMR compared with uniform mesh simulations at the finest resolution.

Acknowledgment

The authors benefitted from useful discussions with Drs. W. Park, P. Parks, H. Strauss, and G. Schmidt. This work was supported by USDOE Contract no. DE-AC020-76-CH03073. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

1. L. Baylor et al., Phys. Plasmas, vol 32, pp:1878, 2000.
2. L. Baylor et al., J. Nucl. Matter, vol 313, pp:530, 2003.
3. P. B. Parks and R. J. Turnbull, Phys. Fluids, vol. 21, pp:1735, 1978.

4. A. K. Macaulay. Nuclear Fusion, vol. 34, pp:43, 1994.
5. S. K. Godunov. Numer. Methods Mech. Contin. Media, vol 1, pp:26, 1972.
6. K. G. Powell et al. J. Comput. Phys., vol. 154, pp:284-309, 1999.
7. B. V. Kuteev. Nuclear Fusion, vol. 35, pp:431, 1995.
8. P. T. Lang et al., Phys. Rev. Lett., vol 79, pp:1487, 1997.
9. P. B. Parks and W. D. Sessions and L. R. Baylor, Phys. Plasmas, vol. 7, pp:1968, 2000. pp:431, 1995.
10. J. A, Breslau, PhD Thesis, Princeton University, 2001.
11. P. Colella, J. Comput. Phys., vol 87, pp: 171-200, 1990.
12. R. Crockett et al. J. Comput. Phys. (submitted), 2003.
13. http://seesar.lbl.gov/ANAG/chombo.

AMR for global atmospheric modelling

N. Nikiforakis

Department of Applied Mathematics and Theoretical Physics,
University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, UK
N.Nikiforakis@damtp.cam.ac.uk

1 Motivation

The resolution of current operational global atmospheric models has been significantly increased thanks to the latest generation of supercomputers. Indicatively, the model used by ECMWF (European Centre for Medium-Range Weather Forecasts), which runs on two identical (but independent) IBM Cluster 1600 supercomputer systems, has a resolution of $T_L511L60$, which is equivalent to an (evenly geographically-distributed) horizontal resolution of about 40 km around the globe and 60 levels (computational cells) along the vertical. This amounts to 20,911,680 computational cells to discretise earth's atmosphere.

However, considering the hugely disparate length- and time-scales of atmospheric flows, this resolution is still not adequate to properly resolve many important phenomena which have an impact on short- to medium-term numerical weather prediction. The same can be said for orographic effects and their interaction with weather phenomena (eg. rain bands). In turn, there are weather events that impact on seasonal totals (eg. tropical cyclones, and jetstreams) and ultimately to climate prediction.

When chemically-active simulations are necessary, eg. to study phenomena like stratospheric ozone depletion, the lack of resolution has an even more detrimental effect on the results. To put this in context, consider the results of the two stratospheric polar vortex simulations shown in figure 1. These show snapshots of potential vorticity (PV), advected on a single isentropic layer at a high and a low grid-resolution grid (0.5×0.5 degrees and 3×3 degrees, respectively). The high resolution run reveals fine filamentary structures over North America and over the eastern Atlantic, while a thicker filament is prominent over Russia. Capturing accurately the behaviour of tracer transport in these regions is important because the chemical composition of the atmosphere is different inside and outside of the polar vortex, so reduced mixing (because of misrepresentation of the topology of the vortex) will have an effect on the production and depletion of chemical species. This is exactly what is happening in the coarse integration, which has given a much broader spatial extent to the

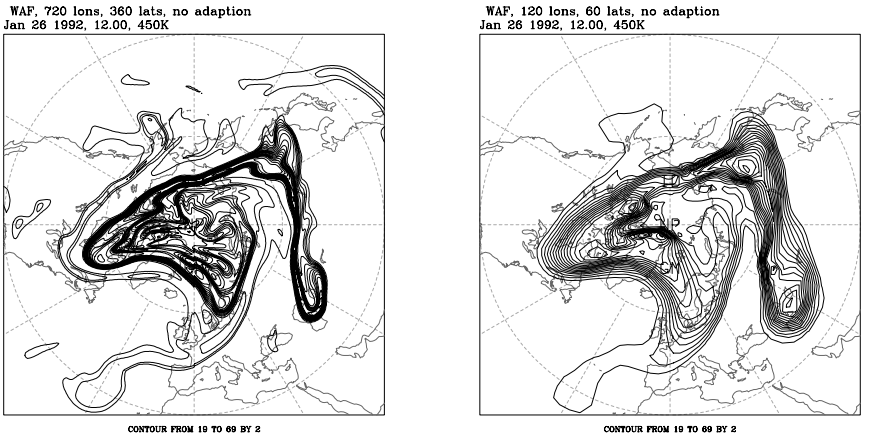


Fig. 1. Snapshot of the evolution of the stratospheric potential vorticity field on the 450K surface, at 0.5×0.5 degrees (left) and 3×3 degrees resolutions.

polar vortex over Russia and missed altogether the finer filaments, so any processes associated with these structures will not be accounted for properly.

The obvious solution of a uniform increase of the model resolution is not a viable option because of the associated computational cost in terms of memory as well as time of integration. Higher resolution operational computations are nevertheless performed by means of Nested Limited Area Modelling (NLAM). The UK Meteorological Office (UKMO) , for instance, has successfully adopted such an approach, where a global model is used to provide boundary conditions to a mesoscale regional model centered on the United Kingdom. The global model has an approximate horizontal resolution of 60km in mid-latitudes while the regional one has approximately 11km. Both the global and mesoscale models have 38 levels in the vertical. Another widely used example is the Penn State - NCAR mesoscale model (MM4 and MM5) (see also the review by Koch and McQueen 1987 and more recent implementations eg. Pielke *et al.* 1992). A disadvantage of this technique is that the boundary conditions of the finer-scale model are ill posed and the solution will be contaminated after a short period of time. A continuous update of the boundary conditions partly rectifies this problem. However this is essentially a one-way interaction approach, which is not well suited to problems where the impact of the small region on the global one is of importance.

This problem is by-passed in some models by integrating the governing equations on a single computational domain, while increasing the resolution locally by stretching the grid in such a way as to increase the cell population over the domain of interest. Spurious solutions will be generated at the coarse/fine boundaries; this can be easily corrected by varying the resolution smoothly. Examples are the operational French and Canadian models (see Courtier and Geleyn 1988 and Coté *et al.* 1993,

respectively) and the ones reported in the articles by Paegle (1989) and Hardiker (1997).

All of the above approaches are based on static grids. Current implementations of time-dependent mesh refinement include the OMEGA framework (Bacon *et al.* 1999) and the work by Behrens *et al.* (2000); both of these models use unstructured meshes. Continuous dynamic grid adaption in the form of node movement (by means of a coordinate transformation technique) within a global model has been recently presented by Prusa and Smolarkiewicz (2003). Structured adaptive mesh refinement has been used in limited-area models for mesoscale storm simulations (Skamarock and Klemp, 1993), but not in global atmospheric modelling.

In spite of these developments, discretisation along the third (vertical) dimension still remains an issue. Most of the current models attempt to capture the vertical structure of the atmosphere from the ground to its uppermost limit (including regions of significant variation like the boundary layer and the tropopause) using 30 to 60 computational cells (levels). Clearly this is not adequate, so subgrid-scale models (parametrisations) are widely employed, with various degrees of success. Models which are chemically-active use even fewer cells (12 levels to discretise 20 km), due to the expense of solving multi-species chemistry, especially for long integrations (Chipperfield 2003). If the processes along the vertical are to be studied by capturing them explicitly, then an efficient three-dimensional time-dependent adaptive methodology has to be used.

To this end, a structured adaptive mesh refinement algorithm for the simulation of three-dimensional chemically-active global atmospheric transport has been developed.

In the rest of this article some background on operational chemistry and transport modelling is given, followed by an overview of the numerical discretisation technique of the governing equations on planar and spherical geometries. Issues specific to AMR on spherical surfaces are discussed and finally the approach is evaluated by means of model problems which have an exact solution as well as real-world case studies.

2 Building an AMR Global Atmospheric Model

This work is concerned with a particular kind of global atmospheric models, namely off-line chemistry and transport models (CTMs), which are the main computational tools of the atmospheric chemistry modelling community. Widely publicised topics of research include the observed downward trend of stratospheric ozone over polar regions (the Antarctic ozone hole) and the aftermath of the 1991 eruption of Mt. Pinatubo.

As the term 'off-line' implies, CTMs differ from general circulation models (GCMs) in the respect that the velocity field (winds) and the rest of the dynamic and thermodynamic variables are not calculated by solving the primitive equations of meteorology. Rather, these and any other input required for the chemical source terms are provided by meteorological analyses or from the output of GCMs (running

in diagnostic or prognostic mode, respectively). Vertical advection rates are usually calculated by other means (eg. from heating rates using a radiation scheme - see for example Chipperfield *et al*, 1996). The chemical fields are initialised by assimilation of observations or measurements.

So in fact CTMs integrate systems of multidimensional, linear advection equations with variable coefficients, coupled through forcing terms. These are in the form:

$$\frac{\partial \psi_i}{\partial t} + \mathbf{v} \cdot \nabla \psi_i = Q_i + S, \qquad i = 1, \ldots, N_s \tag{1}$$

where ψ represents the advected scalar quantity which is usually, but not exclusively, a chemical mixing ratio, $\mathbf{v} = (u, v, w)^{\mathrm{T}}$ is the advection velocity vector, Q_i describes the chemical reactions of N_s chemical species and S denotes other parametrisation terms which may be present. Although the equations to be solved are linear, the variability of the velocity field in space and time means that the numerical scheme needs to address most of the issues related to nonlinear equations, eg. initially smoothly-varying parts of the flow which steepen to form discontinuities (albeit on discrete space).

Given that the grid management in our approach is separate from the numerical discretisation (and indeed from the governing equations), the validity of the grid-adaption approach is not affected, as long as the system remains hyperbolic/parabolic with source terms. If a mixed hyperbolic/elliptic system has to be integrated on the adapted grid, there are additional issues to be considered.

In the rest of this section we present the numerical methodology to solve the system (1) above, the implementation of AMR on concentric spherical surfaces and finally the add-ons necessary to run such a model operationally (i.e. initialising it with real-world data and forcing it with meteorological analyses).

2.1 Numerical formulation

Considering inert transport for a single species in the first instance, the governing equations for multidimensional scalar advection can be written in the form

$$\frac{\partial \psi}{\partial t} + \mathbf{v} \cdot \nabla \psi \; = \; 0 \,, \tag{2}$$

We are seeking to apply standard finite volume techniques for the integration of this equation. In the general case, when $\nabla \cdot \mathbf{u} \neq 0$, the conservation form of Eq. (2) is

$$\frac{\partial \psi}{\partial t} + \nabla \cdot \mathbf{f}(\psi) \; = \; \psi \nabla \cdot \mathbf{v} \tag{3}$$

where $\mathbf{f}(\psi) = \psi \mathbf{v}$ is the flux function. We chose to discretise the left hand side of equation (3) using flux-based, cell-centred finite volume schemes, whilst a discretisation of the right hand side (which is effectively a forcing term) can be included in the update formula, or may be treated by means of operator splitting. Multidimensional extensions are implemented by means of dimensional splitting, but unsplit schemes can also be used. The resulting scheme can be written as:

$$\psi_{ijk}^{(n+1)_x} = \psi_{ijk}^n - \frac{\Delta t}{V_{ijk}} \times \{ [A f(\psi)]_{i+1/2jk} - [A f(\psi)]_{i-1/2jk} + \psi_{ijk}^n [A u^n]_{i+1/2jk} - \psi_{ijk}^n [A u^n]_{i-1/2jk} \} \quad (4)$$

$$\psi_{ijk}^{(n+1)_{xy}} = \psi_{ijk}^{(n+1)_x} - \frac{\Delta t}{V_{ijk}} \times \{ [A f(\psi)]_{ij+1/2k} - [A f(\psi)]_{ij-1/2k} + \psi_{ijk}^n [A v^n]_{ij+1/2k} - \psi_{ijk}^n [A v^n]_{ij-1/2k} \} \quad (5)$$

$$\psi_{ijk}^{(n+1)_{xyz}} = \psi_{ijk}^{(n+1)_{xy}} - \frac{\Delta t}{V_{ijk}} \times \{ [A f(\psi)]_{ijk+1/2} - [A f(\psi)]_{ijk+1/2} + \psi_{ijk} [A w^n]_{ijk+1/2} - \psi_{ijk} [A w^n]_{ijk-1/2} \}, \quad (6)$$

where superscripts $()^{(n+1)_x}$, $()^{(n+1)_{xy}}$ and $()^{(n+1)_{xyz}}$ represent the two intermediate updates and the final solution for a given timestep, A, V are the cell face areas and volumes, respectively, (i, j, k) indexes the grid cells in the three coordinate dimensions and half indices indicate faces between cells. The grid faces have also been assumed to be perpendicular to the axes of the coordinate system being used.

If the final solution of the equation is $\psi_{ijk}^{(n+1)}$ and the solution operators for the partial update equations (4), (5) and (6) are $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively, then the suggested splitting is of the form

$$\psi_{ijk}^{n+1} = \mathcal{X}^{\Delta t} \mathcal{Y}^{\Delta t} \mathcal{Z}^{\Delta t} \psi_{ijk}^n \quad (7)$$

which is formally first order accurate in time (but still second order accurate in space); higher order accuracy may be achieved by employing other types of splitting, at the expense of computational cost. However, the validation case studies indicate that (for the meshes used) there is little, if any, detrimental effect on the quality of the solution.

To evaluate the numerical fluxes, a considerable number of finite volume schemes have been implemented and compared for accuracy, speed and the ability to maintain the interrelationships between the advected species (Walker 2003). For the purposes of this paper we present results using the second-order accurate Weighted Average Flux (WAF) method for hyperbolic conservation laws (Toro 1989); however, this can be replaced by most finite volume schemes that operate on structured, logically-rectangular meshes. The WAF flux along the x direction is simply:

$$f_{i+1/2}^{waf} = \frac{1}{2}(1 + \phi_{1+1/2}) u_{i+1/2} \psi_i + \frac{1}{2}(1 - \phi_{1+1/2}) u_{i+1/2} \psi_{i+1} \quad (8)$$

where $u_{i+1/2} = (u_i + u_{i+1})/2$ and ϕ is a parameter which includes the effect of flux limiter functions:

$$\phi = \mathrm{sgn}(\nu)[1 + (|\nu| - 1)b], \quad \nu = u\Delta t/\Delta x \quad (9)$$

where b implements the limiting procedure. Any limiter function can be used; for example, in the case of superbee:

$$b \;=\; \max[0, \min(2r, 1), \min(r, 2)] \tag{10}$$

where $r = \Delta\psi_{\text{upwind}}/\Delta\psi_{\text{local}}$, is an indication of the steepness of the solution. Similar expressions for the fluxes along the other two directions can be derived.

The basic algorithm described above does not as yet take into account the geometry of the problem. To discretise the thin (albeit three-dimensional) planetary atmospheric layer, we chose to use a regular longitude(λ)-latitude(θ)-height(r) structured grid. For reasons detailed below, this is not necessarily the optimum way to fit a grid in a thin region between two concentric spherical surfaces, but it has the advantage that existing operational ancillary algorithms (related to parametrisations and post-processing) can still be used with relatively minor modifications.

The information stored for cell geometry has to explicitly account for the change of lengths, areas and volumes associated with the spherical polar co-ordinate system. The volumes of the cells are given exactly by $V_{ijk} = \frac{1}{3}\Delta\lambda[\sin\theta_{j+1/2} - \sin\theta_{j-1/2}][3r_k^2\Delta r + (\Delta r)^3/4]$. The values of $\Delta\lambda$, $\Delta\theta$ and Δr are taken to be constant over the whole computational domain so they are fixed from the start of the calculation. Similarly, the face areas are given by $A_{i+1/2jk} = r_k\Delta\theta\Delta r$, $A_{ij+1/2k} = \cos\theta_{j+1/2}\, r_k\Delta\lambda\Delta r$ and $A_{ijk+1/2} = r_{k+1/2}^2\Delta\lambda[\sin\theta_{j+1/2} - \sin\theta_{j-1/2}]$. To compensate for the variation in the cell sizes a small modification is necessary to the WAF method, and in particular to the local dual cell CFL number used in the calculation of ϕ:

$$\nu_{i+1/2} \;=\; u_{i+1/2}\Delta t\,\frac{2A_{i+1/2}}{V_i + V_{i+1}}, \tag{11}$$

A different way to achieve the same result would be to apply the finite volume method to the transformed differential equations written in curvilinear (spherical polar) coordinates.

2.2 Implementation of AMR on the sphere

The basic structure of the AMR procedure as presented in Berger and Oliger (1984) and Berger and Colella (1989) has been retained, save for a number of application-dependent modifications. The computational domain is covered by a single regular longitude(λ)-latitude(θ)-height(r) coarse grid G_0, upon which the AMR hierarchical system of grid levels[1] G_1, G_2 etc can be built, see figure 2. The convergence of the grid lines towards the poles creates a smooth distortion of the grid and as a result there is a considerable difference in size between the cells near to the equator and the ones adjacent to the pole. As a result, a difference from a 3D orthogonal, cartesian AMR code is that the cell geometry has to be generalised to curve-faced hexahedra, which are allowed to vary in size as a function of latitude.

The variation of the cell volumes over the grid affects the transfer of solution information from fine meshes to coarse ones, which must be carried out through a volume weighted averaging procedure to ensure conservation. Since there are exact

[1] This should not to be confused with the terminology used in atmospheric models where 'levels' refers to the number of cells discretising height.

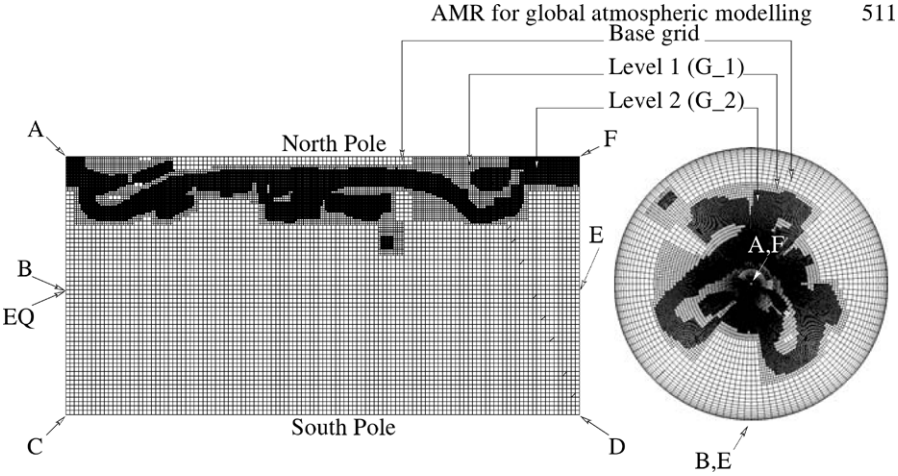


Fig. 2. Cylindrical (left) and satellite projections of a hierarchy of adaptive meshes (two levels of refinement) on a spherical surface (details about this particular integration are given in section 3.2).

expressions for the cell volumes, which additionally do not vary in the zonal direction, this procedure is simplified considerably. For the corresponding transfer of information from coarse meshes to fine ones (used to fill the dummy cells surrounding the mesh boundaries and the updated mesh structure) a MUSCL reconstruction is found to allow for the possibility that the integral-average of a linear reconstruction is not necessarily equal to the reconstructed solution value at the cell's centre. This is no longer guaranteed to be monotonic, but it will not change the sign of the solution. An interpolation which is inherently conservative and monotonic over this geometry is part of future work.

The distortion of the cells leads to singular points on the co-ordinate system and on the resulting grid at the poles. This poses a serious problem to many cell-vertex numerical schemes, but not so for the family of cell-centred finite volume schemes discussed above, where no information is kept at the vertices.

The boundary conditions can be described if we consider the cylindrical and satellite projections of the same single-layer AMR grid shown in figure 2. The grid boundary defined by points A,B,C merges with the corresponding boundary F,E,D, while the boundary A,F, which appears as a line on the cylindrical projection, collapses into a single point on the satellite projection, namely the north pole (ditto for line C,D which is the south pole). Periodic boundary conditions are enforced across both boundaries. The communication at the pole is facilitated by information transfer between diametrically opposite cells (which implies that an even number of longitudes has to be used), remembering that the velocity field has to be negated. For the inner and outermost shells reflective and transmissive boundary conditions have been used, but these can be modified to suit specific operational requirements.

For our implementation of AMR, the refinement factor between any two consecutive grid levels can take any positive integer value, which is directionally independent, i.e. the horizontal refinement factor can be different to the vertical. This is

particularly useful feature, considering the aspect ratio of the atmospheric layer. A simple monitor function (adaption criterion) used for tracking flow features of interest can be:

$$\xi_{ijk} = \max(|\psi_{i+1jk} - \psi_{i-1jk}|, |\psi_{ij+1j} - \psi_{ij-1k}|, |\psi_{ijk+1} - \psi_{ijk-1}|), \qquad (12)$$

which in effect considers local cell differences in tracer values; this is suitable for idealised advection studies. Alternative criteria can be used, depending on the complexity of the flow dynamics (or physics/chemistry), such as vorticity or rate of reaction.

2.3 Operational issues

Some additional modifications are essential in order to run such a model using meteorological analyses. The analyses are provided from a number of sources, eg. ECMWF. Initial conditions are interpolated trilinearly from this data on to the initial adapted grid. The winds are read at regular intervals (typically six hours) and their values at intermediate time-intervals have to be interpolated (trilinearly in space, linearly in time) to match the variable timestepping nature of the approach.

Two separate models have been build based on the approach described in this and in the previous sections, namely CTM-AMR-SL (Nikiforakis and Toro 1995, Walker and Nikiforakis 2000), which was based on a two-dimensional internal ballistics algorithm (Boden and Toro 1997), and CTM-AMR-3D (Hubbard and Nikiforakis 2003, Nikiforakis and Camaji 2003) which was based on a three-dimensional gas-dynamics agorithm (Berger 2000). The former operates on single spherical layers of the atmosphere and is more suitable for stratospheric flows (which are highly stratified), while the latter is a full three-dimensional code, albeit lacking explicit orography. By this we mean that although the effects of orography can be seen in the velocity, pressure and temperature fields as in any other CTMs (since these models are forced by meteorological analyses), the lower boundary is a smooth spherical surface, so there is a natural limit as to the location of its lowest level. In the next section some numerical results from these models are presented.

3 Results

The gains from AMR codes are obvious for flows where distinct gas-dynamical or chemically-active features exist (like shock waves or flames), which additionally cover a relatively limited part of the computational domain. It is not immediately apparent whether AMR can be of any use in the atmosphere where such features do not exist. The intention of this evaluation exercise was not so much to quantify the efficiency of the code, but to assess whether AMR would be of any use at all for global atmospheric calculations. Problems with exact solutions on spherical surfaces and real world case-studies were attempted. The numerical results are from CTM-AMR-SL as well as CTM-AMR-3D.

3.1 Model problems

An initial indication of the accuracy and performance of the codes is given by model problems which have an analytical solution, so some absolute statements can be made about the speed-up of the runs correlated to error estimates.

Solid body rotation

The first test case considers solid body rotation of a cosine bell-shaped profile on a single spherical surface traversing the pole at various angles to the equator. The velocity components of the advecting wind field (Williamson *et al.* (1992)) are given by

$$\begin{aligned} u &= u_0(\cos\alpha\cos\theta + \sin\alpha\cos\lambda\sin\theta) \\ v &= -u_0\sin\alpha\sin\lambda \end{aligned} \tag{13}$$

where α is the angle between the axis of solid body rotation and the polar axis of the spherical coordinate system. The period of the rotation is taken to be 12 days, and the radius of the earth to be 6371.22km, giving a value of $u_0 \approx 38.61\text{ms}^{-1}$. The initial scalar distribution is of the form

$$\psi = \begin{cases} (1+\cos(\pi r/R))/2 & \text{if } r < R \\ 0 & \text{if } r \geq R \end{cases} \tag{14}$$

where

$$r = \cos^{-1}[\sin\theta_c\sin\theta + \cos\theta_c\cos\theta\cos(\lambda-\lambda_c)] \tag{15}$$

is the great circle distance between (λ,θ) and the bell centre, initially taken as $(\lambda_c,\theta_c) = (\pi/2, 0.0)$. The bell radius R is set to $7\pi/64$, as proposed by Williamson and Rasch (1989). All numerical experiments were run with a maximum Courant number of 0.9.

This case study was used to evaluate both the single-layer and the 3D models (the latter in single-layer mode), representative results shown in figures 3 and 4. The bell is rotated at different values of α, and the error is measured using the standard l_1, l_2 and l_∞ norms, shown as bars in figure 3. These results compare favourably with the ones found in the open literature for the same case-study (eg. Williamson and Rasch, 1989; Lin and Rood, 1996; Nair *et al* 1999), inspite the lower order of accuracy of the underlying numerical scheme. The stalks show cpu times on a SUN ULTRA 10 workstation. While the errors of the AMR runs are of similar magnitude to the unadapted ones, the savings in cpu times (factors of 20 up to 50, depending on α) are significant. Unfortunately these are not representative on the performance of AMR when it comes to using meteorological analyses, as discussed below. The sequence of contour plots in figure 4 shows that the profile crosses the pole without distortion, so there is no apparent problem with the methodology at the pole singularity.

The message from this case-study is that the alterations on AMR necessary to operate on spherical domains had no detrimental effect on mesh management and communication, or on the overall code efficiency and accuracy, not least for the problematic regions of the grid singularities at the poles.

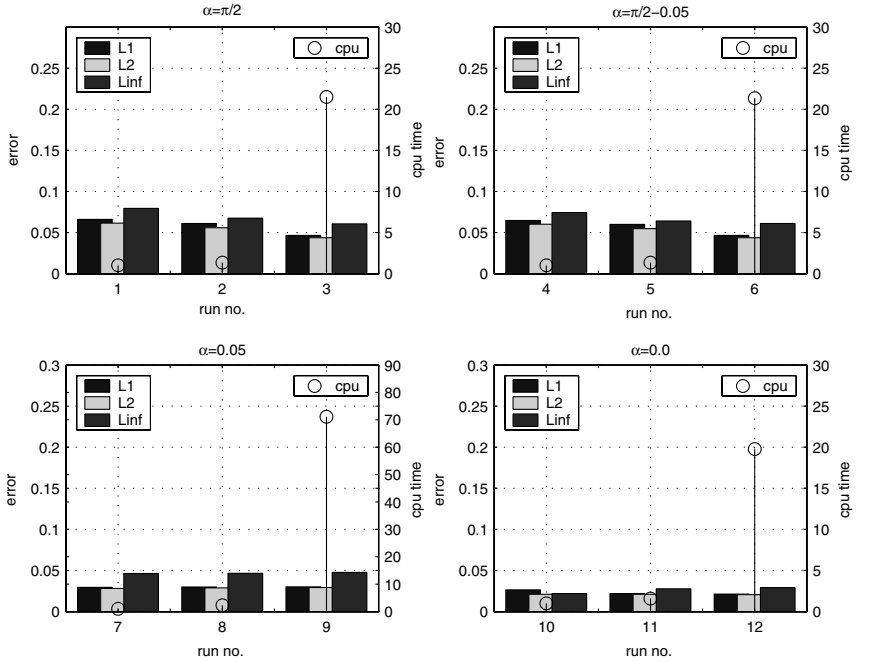


Fig. 3. Error emasures correlated to cpu time for the solid body rotation problem for various angles of rotation and refinement ratios. All runs are at an effective resolution of 384×192; in every subplot there are bars of l_1, l_2 and l_∞ norms for a constant cell-width run (run nos. 3,6,9,12), a run with two levels of refinement at $64\times32(\times2\times3)$ (run nos. 1,4,7,10) and a run with one level of refinement at $128\times64(\times3)$ (run nos. 2,5,8,11). The errors of the AMR runs are of similar magnitude to the unadapted ones, but the savings in cpu times are significant.

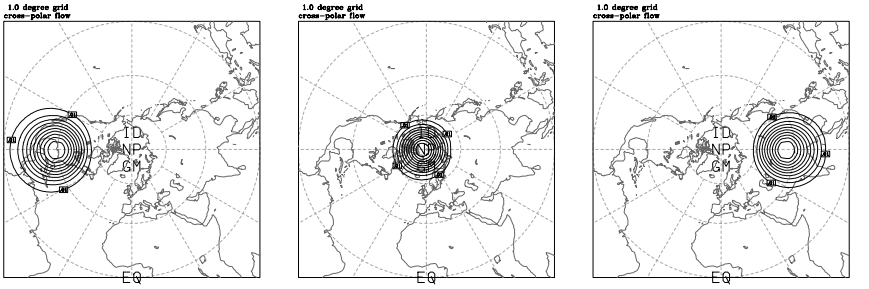


Fig. 4. Solid body rotation of a cosine bell as it traverses the North Pole; no distortion of the profile, due to the singular point at the pole, is apparent.

Kinematic cyclogenesis

A case study which is more relevant to atmospheric flows is a generalisation of the idealised cyclogenesis problem of Doswell (1984) to spherical geometry, as presented by Nair *et al.* (1999). Given a rotated coordinate system (λ', θ') with its north

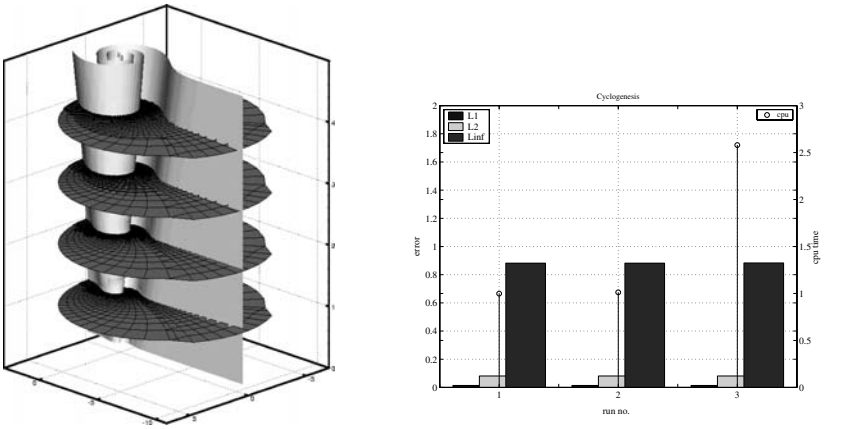


Fig. 5. Solution surface and superimposed meshes for the cyclogenesis case-study, from a global 3D run, using dynamically-adaptive grids for a spatially-varying vortex (left). The solution has been projected stereographically onto the planes tanget to the centre of rotation and only the region in the vicinity of the solution is shown. The errors and cpu times on the right refer to comparisons with the exact solution on a single spherical surface.

pole at (λ_0, θ_0) with respect to the regular spherical coordinate system (λ, θ), a steady circular vortex is defined by zero normal velocity $v' \equiv \mathrm{d}\theta'/\mathrm{d}t$ and tangential velocity

$$u'(\theta') \equiv \cos\theta' \frac{\mathrm{d}\lambda'}{\mathrm{d}t} = \frac{3\sqrt{3}}{2} a \,\mathrm{sech}^2(\gamma\rho)\tanh(\gamma\rho), \tag{16}$$

where a is the radius of the sphere,

$$\rho(\theta') = \frac{2\cos\theta'}{1+\sin\theta'}, \tag{17}$$

and $\gamma = 3/2$ is a stretching parameter that controls the length scale of the vortex. ρ can be interpreted as the distance from the north pole of the polar stereographic projection of the point (λ', θ'). The amplitude of the vortex has been normalised to have a maximum tangential velocity of unity, which occurs when

$$\theta' = 2\tan^{-1}\left(\frac{\gamma - c}{\gamma + c}\right), \quad c = \frac{1}{4}\ln\left(\frac{\sqrt{3}+1}{\sqrt{3}-1}\right) \approx 0.3292395 \tag{18}$$

The initial conditions are taken to be

$$\psi(\lambda', \theta', 0) \;=\; -\tanh\left[\frac{\rho}{\delta}\sin\lambda'\right], \tag{19}$$

in which $\delta = 0.01$ is the characteristic width of the frontal zone. This problem has an analytic solution, which is given by

$$\psi(\lambda', \theta', t) \;=\; -\tanh\left[\frac{\rho}{\delta}\sin(\lambda' - \omega t)\right], \quad \omega(\theta') \;=\; \frac{u'(\theta')}{a\cos\theta'} \tag{20}$$

where $\omega(\theta')$ is the angular velocity. Further details of the stereographic projection in the rotated coordinate system can be found in Nair *et al.* (1999), which also presents results with which these can be compared.

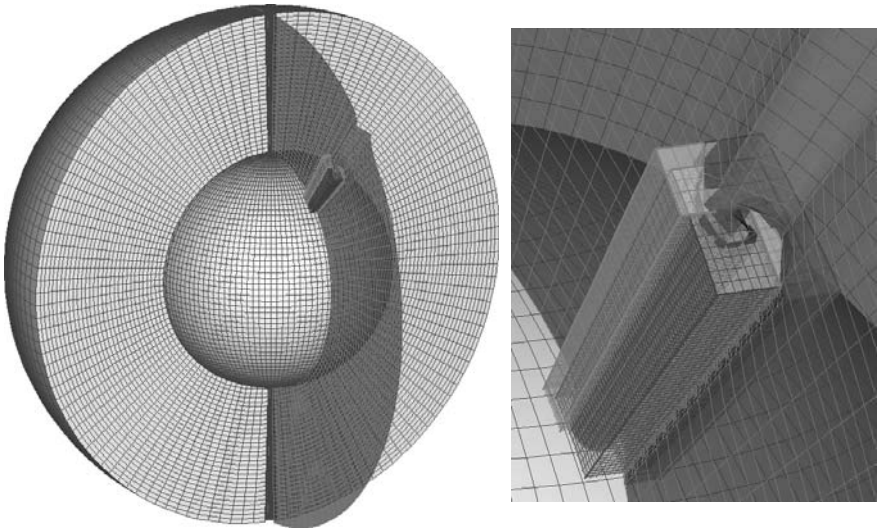


Fig. 6. Global mesh arrangement and the iso-surface of the solution for the kinematic cyclogenecis problem, for a static grid case, where the grid has been refined up to a specific height (left). Some of the surfaces have been removed in the zoom (right) to reveal detail. This mesh arrangement is suitable for mesoscale studies, as a two-way interaction replacement of nested grids.

The problem can be modified further to accommodate a variation on the strength of the vortex as a function of height. The solution surface and the adapted grid (two levels of refinement) are shown in figure 5, on a stereographic projection on the plane tangent to the plane of rotation and for an area limited at the vicinity of the vortex.

The error measures (figure 5) and cpu times refer to comparisons with the exact solution on a single spherical surface. The errors, as with the previous case-study,

are within reasonable limits and are of similar magnitude, irrespective of the mesh arrangements. Although there is a saving in cpu-time using AMR, the gains are less significant as compared to the solid-body rotation problem. In fact, a better arrangement for this kind of problem is the one shown in figure 6 where only static grids have been used. This grid arrangement is more suitable as a replacement for NLAMs, as discussed in the introduction, possibly in conjuction with dynamically-adaptive grids. While the static grids capture in detail the flowfield in the area of interest at all times, the dynamic grids can resolve and track the important features at a global scale, which in turn can influence the solution within the limited area. Further work on this topic will be reported in a separate article.

3.2 Meteorological flows

Time-dependent adaptive mesh refinement is cost-efficient only if there are distinct features in the flow and also if these features do not cover a large portion of the domain. Since AMR has not been used before in global atmospheric simulations, questions arise whether these prerequisites exist.

Some intuition in this matter is provided by considering that there is a notable variation of the magnitude of potential vorticity (PV) in certain regions of the atmosphere, namely at the edge of the polar vortex and also at the tropopause. At the same time, PV is less sensitive to other physical and chemical processes than most dynamic variables and chemical species, at least for short-term integrations (of the order of a few days, depending on the circumstances), making comparisons with observations feasible to some extent.

Since no orography exists in the current version of these models, we can only begin to evaluate them for the upper layers of the atmosphere. We therefore consider two case studies in this section, one from the evolution of the northern polar vortex and another from a case of stratosphere-troposphere exchange. The highly stratified nature of the former allows employment of CTM-AMR-SL, while the considerable variation of PV along the vertical in the latter, necessitates use of CTM-AMR-3D.

Stratospheric polar vortex flow

For this case-study we consider the period between 16 and 26 January 1992. This period was studied as part of the EASOE (European Arctic Stratospheric Ozone Experiment) campaign (November 1991 - April 1992). EASOE was conceived against a background of growing concern over stratospheric ozone decline over populated latitudes of the northern hemisphere, to improve our understanding of its behaviour. The high degree of distortion of northern polar vortex, and the filamentary structures related with wave breaking events, offer a good case-study for testing AMR. The evolution of the stratosphere during this period has been well documented, making it well-suited as an operational-case benchmark.

Unadapted as well as adapted integrations were performed using the CTM-AMR-SL on the 450K isentropic surface. This was forced by ECMWF analyses to advect PV (potential vorticity). The objective was to compare the cpu run-times between

a high resolution unadapted integration and an adaptive one at the same effective resolution. The results from the two runs have to be in good agreement with each other an also compare favourably with a third-party simulation, which has in turn been validated against analyses and observations. We are of course aware that it is not

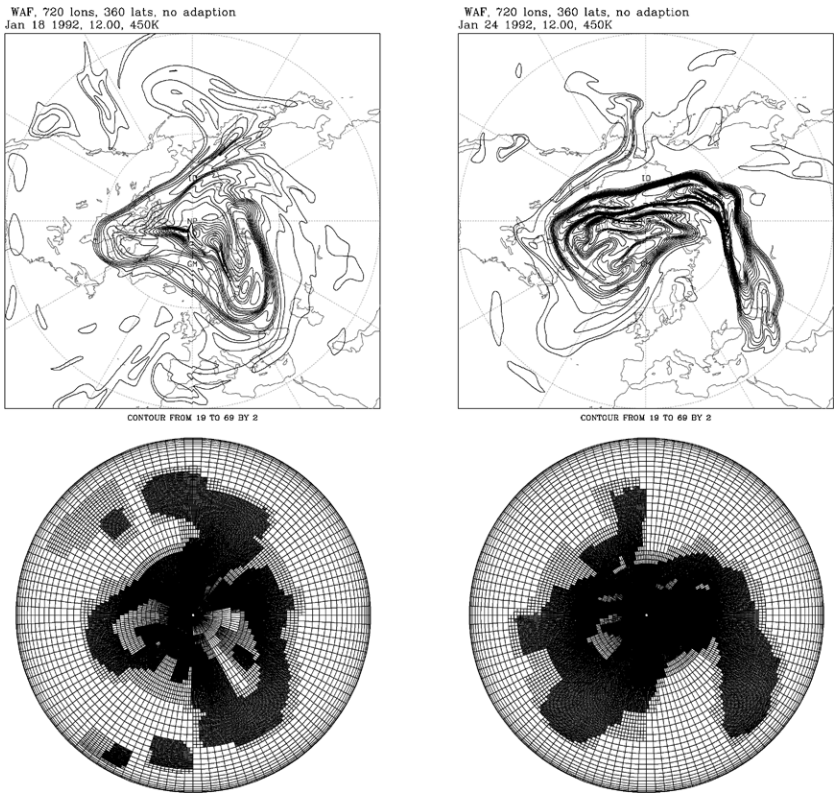


Fig. 7. Contours (polar stereographic projection) of the stratospheric potential vorticity field on the 450K surface on January 18 and 24 1992 from a 720×360 single-layer integration. The corresponding mesh distributions (satellite projection) are from an adaptive run, of the same effective resolution on a $120\times60(\times\ 2\ \times\ 3)$ grid. AMR accelerated the computation by a factor of five, in this case.

easy to make accurate assessment of performance correlated to accuracy for global models because although qualitative comparisons can be done with observations, the data is not reliable enough to do detailed comparisons on this scale. Also, since the models are initialised from sparse data which have been assimilated, the initial conditions for the calculations (and hence the results) are as good as the assimilation procedure.

Two snapshots of the results for the unadapted and AMR runs (showing contour plots and mesh arrangement respectively) are presented in figure 7. The fine-mesh unadapted integration discretised the 450K spherical surface on a 0.5×0.5 degrees, 720×360 (259200 computational cells) grid. The adaptive run was initialised in exactly the same way as the fine-mesh unadapted one and data were output in the same dates and times. The base mesh of this run was 120×60 cells (effective lowest-level grid-resolution of $3\,\text{deg} \times 3\,\text{deg}$) and there are two levels of refinement at factors of 2 and 3, giving an effective highest grid-resolution of 720×360 cells ($0.5\,\text{deg} \times 0.5\,\text{deg}$), i.e. the same as the unadapted run.

Plumb *et al* (1994), have used a contour advection technique to study the same period, initialised and forced in a similar way to ours. Their results show favourable qualitative agreement with observations and can be used as a valid numerical benchmark. The main features of the flow as described in that paper are captured in our integrations and there is very good qualitative agreement with their contour plots.

The results from the unadapted and AMR runs are nearly identical, but while the unadapted integration took about 10 hours real time (9.53 cpu units) on a desktop SUN workstation, the AMR one required just over two hours (1.59 cpu units), so there is a factor of five saving in cpu-time, even for a single tracer. It has to be kept in mind that this figure is not representative because in operational simulations there are typically more that 10-20 species, while the overhead associated with AMR is the same, so the efficiency of the algorithm increases significantly. For reference we mention that a lower resolution integration (3×3 degrees, $120 \times 60 = 7200$ computational cells, i.e. the same resolution as the base grid of the AMR run), took just over 4.5 minutes on the same machine.

The major advantage becomes obvious if we consider the total number of cells that need to be integrated. This is illustrated in figure 2, for the 20 January 1992, where polar stereographic and cylindrical projections are shown side by side. Because of the decreasing cell-size, as we move from the equator to the poles, the former gives a false impression of the number of cells that have to be integrated; the true number is shown in the cylindrical projection. If the integration was chemically-active, and keeping in mind that chemistry solvers are very expensive compared with advection ones, the savings are expected to be significantly higher.

Stratosphere/troposphere exchange

The ability of AMR to capture a highly temporal and spatial variation of the flow along all three space dimensions is evaluated using CTM-AMR-3D in a region of the tropopause. The case-study is a stratosphere-troposphere exchange event which took place over the North Atlantic during June 1996. The feature of interest is an isolated cyclonic vortex (cut-off low), which formed as a result of a meridional excursion of a jetstream. Within the vortex the tropopause is 2-3 km lower than in the surrounding atmosphere, which forms a pocket of isolated high PV, stratospheric air (Hoskins *et al* 1995, Gouget *et al* 2000). The cut-off low decayed over a period of few days, and the trapped air either returned to the polar stratosphere or mixed in the troposphere. The process is therefore of importance to stratosphere-troposphere exchange, which is of

interest because of its implications to the transport of chemical species between the two atmospheric regions. Given the highly three-dimensional nature of this process, which take place within a very narrow region of the atmosphere, AMR is well suited for its study.

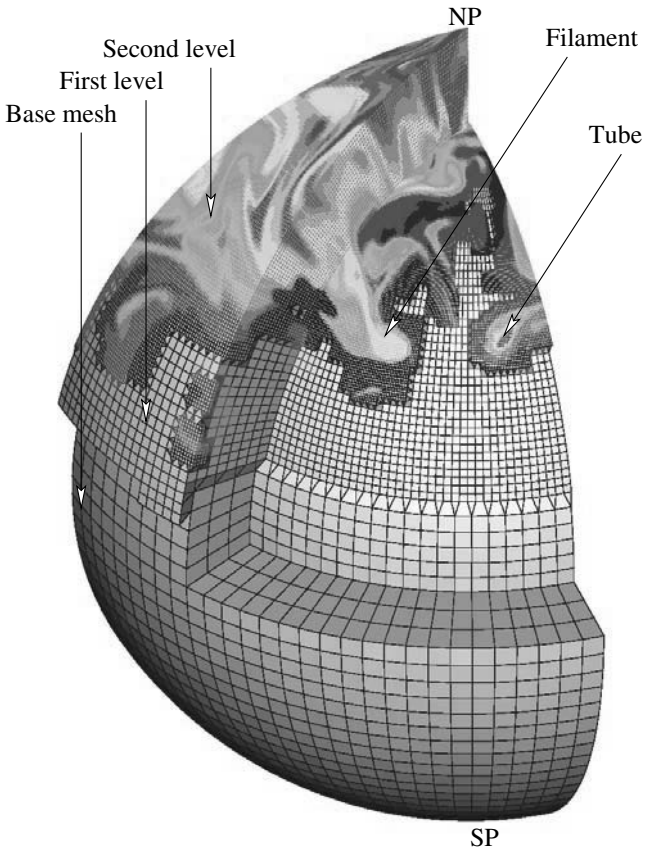


Fig. 8. Three dimensional simulation of a stratosphere-troposphere event over the Atlantic during June 1996. Note that although the grid appears to contain triangular cells, this is nothing but a graphical artefact (see main text).

Meteorological analyses from the ECMWF Re-Analysis datasets (the ERA project) were used to provide three-dimensional velocity fields and values of potential vorticity at intervals of 6 hours for the period between 06:00 17 June and 18:00 23 June (the numerical calculation only used the data between 300K and 360K). Snapshots of the results are presented in figures 8, 9 and 10. These integrations were performed on an underlying coarse $120 \times 60 \times 8$ mesh with two levels of refinement, the first by a factor of 2 in each dimension and the second by a factor of 3. The adaption criterion

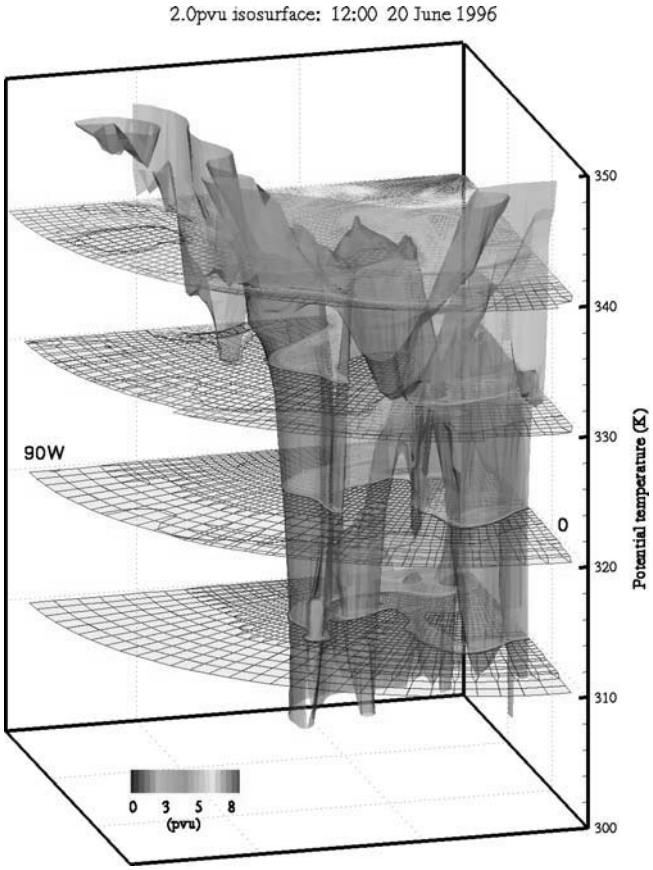


Fig. 9. Snapshot of the evolution of an isolated cyclonic vortex (cut-off low) over the Atlantic (90 degrees west to Greenwich Meridian), on 20 June 1996. The PV = 2.0 isosurface is shown; the computational meshes are shaded by PV magnitude. Global atmospheric AMR run on a $120 \times 60 \times 8(\times 2 \times 3)$ grid, between 290K and 370K.

$$\xi_{ijk} = \frac{\max(|\psi_{i+1jk} - \psi_{i-1jk}|, |\psi_{ij+1k} - \psi_{ij-1k}|, |\psi_{ijk+1} - \psi_{ijk-1}|)}{\max(1, \sqrt{\psi_{ijk}})}, \qquad (21)$$

was used for these runs, where ψ represents PV. The grid cells are flagged for adaption when this expression satisfies $\xi \geq 1.0$. As in the previous case-study, we are only interested in the evolution of PV in the northern hemisphere, and this is enforced by the conditional statement $1 \leq \psi \leq 10$. Because PV decreases in magnitude towards the equator, the denominator includes $\sqrt{\psi}$ to ensure that the monitor can track steep gradients of the solution without using an excessive number of fine mesh patches.

Comparisons with observations and between various computational runs are discussed in the paper by Hubbard and Nikiforakis (2003), where it was found that the

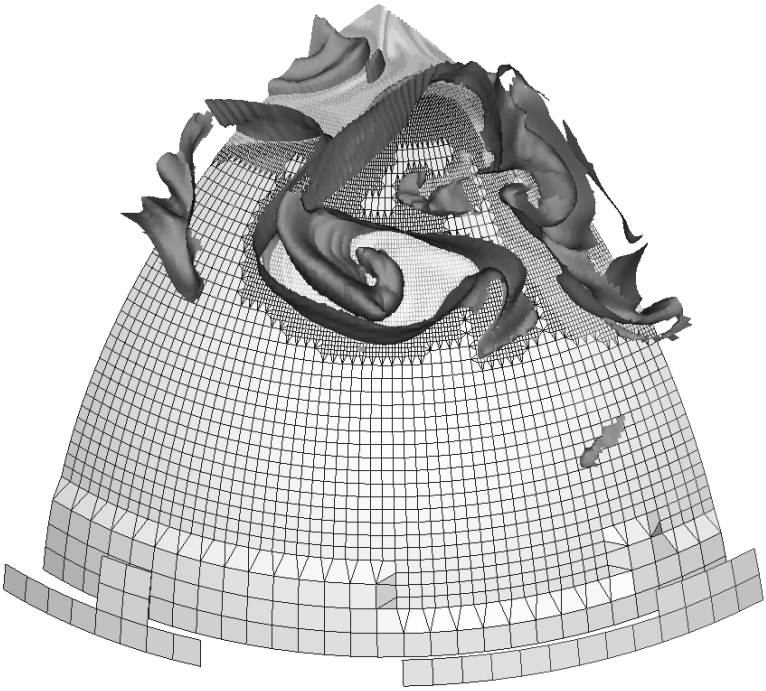


Fig. 10. Visualisation of a cut-off low over the Atlantic, 21 June 1996. The PV=2.0 isosurface is shown; the computational meshes are shaded by PV magnitude. For details of this integration see the caption in figure 9.

shape and size of these flow features agree closely with observations. Also, the unadapted and AMR results are in good agreement between them. The flow features that we were interested to capture included filaments and vertical tubes, across the boundaries of which there was abrupt change of PV magnitude. Figure 8 shows a section of the computational domain, where a coarse grid covers most of the south hemisphere, while the first level of refinement covers most of the northern one. The second level of refinement has picked the regions of high three-dimensional variation of PV. Note that although the grid appears to contain triangular cells, this is only a graphical artefact created by converting the hierarchy of structured rectangular meshes into a single unstructured grid whose vertices correspond to the original cell centres.

A detail of the domain on a stereographic projection (90 degrees west to the Greenwich meridian and 300K to 350K isentropic surfaces), is shown in figure 9. The steep variation of PV inside and out of the tubes and filaments is evident by plotting the 2.0 PV units isosurface (translucent blue). The isosurface intersects four planes of constant values of potential temperature (310K to 340K), where the meshes

are coloured by PV magnitude. Finally, figure 10 shows a PV=2.0 isosurface (which helps to visualise the synoptic vortical structure over the Atlantic) and the underlying computational meshes, which are coloured by PV magnitude. The criterion has flagged the second-level meshes efficiently, but it is not clear at this stage whether it would be more cost-efficient to have a lower mesh coverage at the first level of refinement. The effect of more involved adaption criteria is part of our current work.

The number of computational cells during this run varied between 2.7 and 3.1 million and it was obtained using one-sixth of the cpu time of the constant cell-width equivalent integration (12.4416 million cells) on a SUN-ULTRA 10 workstation.

4 Concluding remarks

In this article the development of AMR models for global atmospheric chemistry and transport (CTMs) at the University of Cambridge is reviewed. After a brief introduction of CTMs, the governing equations and the methodology for their numerical solution is presented. This is followed by the modifications of the standard AMR methodology necessary to operate on spherical layers and to be forced by meteorological analyses. Although atmospheric flows do not exhibit the same features of shocked gas dynamics (where AMR has been known to be very efficient), initial results based on idealised and real-world case-studies indicate that considerable savings can be achieved. Future work will concentrate on the implementation of different systems of equations and orography.

Acknowledgements The author would like to thank Dr S. Billet, Dr M. Hubbard and Miss A. Camaji who worked on the codes, Professor E.F. Toro and Professor M. Berger for making available the original software, Professor A. O'Neill, Professor J. Slingo and Professor J. Pyle who fosterd this work within UGAMP (UK Universities Global Atmospheric Modelling Programme) and NERC (Natural Environment Research Council) for providing part of the funding.

References

1. Bacon, D.P., Ahmad, N.N., Boybeyi, Z., Dunn, T.J., Hall, M.S., Lee, P.C.S., Sarma, R.A. Turner, M.D., Waight, K.T., Young, S.H., Zack, J.W.: A Dynamically Adapting Weather and Dispersion Model: The Operational Multiscale Environment Model with Grid Adaptivity (OMEGA). Monthly Weather Review: **128**, No. 7, pp. 2044-2076 (2000)
2. Behrens, J., Dethloff, K., Hiller, W., Rinke, A.: Evolution of Small-Scale Filaments in an Adaptive Advection Model for Idealized Tracer Transport. Monthly Weather Review: **128**, 2976-2982 (2000)
3. Berger, M.J. cited 2002: Adaptive mesh refinement software for hyperbolic conservation laws. Available online at `http://cs.nyu.edu/cs/faculty/berger/amrsoftware.html`
4. Berger, M.J., and Colella P.: Local adaptive mesh refinement for shock hydrodynamics. J. Comput. Phys. , **82**, 67-84 (1989)

5. Berger, M.J. and Oliger J.: Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys. , **53**, 482-512 (1984)
6. Boden E.P. and Toro E.F.: A Combined Chimera-AMR Technique for Computing Hyperbolic PDEs. Proc. Fifth Annual Conference of the CFD Society of Canada, 5.13-5.18 (1997)
7. Chipperfield, M.P.: A three-dimensional model study of long-term mid-high latitude lower stratosphere ozone changes, Atmos. Chem. Phys., **3**, 1-13 (2003)
8. Chipperfield, M. P., Santee, M. L., Froidevaux, L., Manney, G. L., Read, W. G., Waters, J. W., Roche, A. E. and Russell, J. M.: Isentropic 3D Chemical Transport Model and Comparison with UARS Data in Southern Polar Vortex September 1992. J. Geophys. Res., **101**, no. D13, 18861-18881 (1996)
9. Côte J., Roch M., Staniforth A. and Fillion L.: A variable-resolution semi-Lagrangian finite-element global model of the shallow-water equations. Mon. Wea. Rev., **121**, 231-243 (1993)
10. Courtier P. and Geleyn J.F.: A global numerical weather prediction model with variable resolution. Application of the shallow water equations. Techniques for horizontal discretisation in NWP models. ECMWF Workshop proceedings. (1987)
11. Doswell, C.A.: A kinematic analysis of frontogenesis associated with a nondivergent vortex. J. Atmos. Sci., **41**, 1242–1248 (1984)
12. Gouget H. , Vaughan G. , Marenco A. and Smit H.G.J.: Decay of a cut-off low and contribution to stratosphere-troposphere exchange. Quart. J. Roy. Met. Soc. **126**, 1117-142, (2000)
13. Hardiker, V.: A Global Numerical Weather Prediction Model with Variable Resolution. Monthly Weather Review: **125**, 1, pp. 59-73 (1997)
14. Hubbard, M.E., Nikiforakis, N.: A Three-Dimensional, Adaptive, Godunov-Type Model for Global Atmospheric Flows. Mon. Wea. Rev., **131**, 8, pp. 1848-1864 (2003)
15. Hoskins, B.J., McIntyre, M.E. and Robertson A.W.: On the use and significance of isentropic potential vorticity Boden E P and Toro E F (1997). A Combined Chimera-AMR Technique for Computing Hyperbolic PDEs. Proc. Fifth Annual Conference of the CFD Society of Canada, 1997, pp 5.13-5.18. maps. Quart. J. Royal Meteor. Soc., **111**, 877-946 (1985)
16. Lin, S., Rood, R.B.: Multidimensional Flux-Form Semi-Lagrangian Transport Schemes. Mon. Wea.r Rev. Vol. **124**, No. 9, pp. 2046-2070 (1996)
17. Nair, R., Côté J., and Staniforth A.: Cascade interpolation for semi-Lagrangian advection over the sphere. Q.J.R. Meteorol. Soc., **125**, 1445–1468 (1999)
18. Nikiforakis N. and Camaji A.: Combined dynamic and static mesh refinement for global atmospheric chemistry and transport modelling. In preparation (2004)
19. Nikiforakis N. and Toro E.F.: Evaluation of the WAF Method for Kinematic and Dynamic Atmospheric Modelling Problems. Proc. ICFD Conference on Numerical Methods for Fluid Dynamics, Oxford (1995)
20. Paegle, J.: A Variable Resolution Global Model Based upon Fourier and Finite Element Representation. Monthly Weather Review: Vol. **117**, No. 3, pp. 583-606 (1989)
21. Pielke, R.A., Cotton, W.R., Walko, R.L., Tremback, C.J., Lyons, W.A., Grasso, L.D., Nicholls, M.E., Moran, M.D., Wesley, D.A., Lee, T.J. and Copeland, J.H.: A comprehensive meteorological modeling system - RAMS. Meteor. Atmos. Phys., **49**, 69-91 (1992)
22. Plumb, R.A., Waugh,D.W., Atkinson,R.J., Newman, P.A., Lait, L.R., Schoeberl, M.R., Browell, E.B., Simmons, A.J. and Lowenstein, M.: Intrusions into the lower stratospheric Arctic vortex during the winter 1991-1992. J. Geophys. Res., **99**, (D1), 1089–1105 (1994)
23. Prusa and Smolarkiewicz P.: An all-scale anelastic model for geophysical flows: dynamic grid deformation Volume J. Comp. Phys. **190**, 2, 601-622 (2003)
24. Skamarock, W.C., and Klemp J.B.: Adaptive grid refinement for 2-dimensional and 3-dimensional nonhydrostatic atmospheric flow. Mon.Weather Rev, **121**(3), 788–804 (1993)

25. Toro, E.F.: A weighted average flux method for hyperbolic conservation laws. Proc. R. Soc. Lond., **423**, 401–418 (1989)
26. Walker, B.D.: Impact of Computational Methods on Tracer Inter-Relations in Atmospheric Chemistry and Transport Models. PhD Thesis, University of Cambridge, Cambridge (2003)
27. Walker B.D. and Nikiforakis N.: A Large Time Step Godunov Type Model of Global Atmospheric Chemistry and Transport, Published in Godunov: Theory and Applications (refereed), Edited Review, E.F, Toro (Editor), Kluwer Academics/Plenum Publishers, Proceedings of the International Conference on Godunov Methods. (2000)
28. Williamson, D.L., Drake, J.B., Hack, J.J., Jakob, R., Swartztrauber, P.N.: A standard test set for numerical approximations to the shallow water equations in spherical geometry. J. Comput. Phys., **102**, 211–224 (1992)
29. Williamson, D.L., and Rasch P.J.: Two-dimensional semi-Lagrangian transport with shape-preserving interpolation. Mon. Weather Rev., **117**, 102–129 (1989)

Simulation of Vortex–Dominated Flows Using the FLASH Code

Vikram Dwarkadas,[1] Tomek Plewa,[1] Greg Weirs,[1] Chris Tomkins,[2] and Mark Marr-Lyon[2]

[1] ASCI FLASH Center, University of Chicago vikram@flash.uchicago.edu, tomek@flash.uchicago.edu, weirs@flash.uchicago.edu
[2] Los Alamos National Laboratory ctomkins@lanl.gov, mmarr@lanl.gov

Summary. We compare the results of two–dimensional simulations to experimental data obtained at Los Alamos National Laboratory in order to validate the FLASH code. FLASH is a multi–physics, block–structured adaptive mesh refinement code for studying compressible, reactive flows in various astrophysical environments. The experiment involves the lateral interaction between a planar Ma=1.2 shock wave with a cylinder of gaseous sulfur hexafluoride (SF_6) in air. The development of primary and secondary flow instabilities after the passage of the shock, as observed in the experiments and numerical simulations, are reviewed and compared. The sensitivity of numerical results to several simulation parameters are examined. Computed and experimentally measured velocity fields are compared. Motivation for experimental data in planes parallel to the cylinder axis is provided by a speculative three–dimensional simulation.

1 Introduction

The impulsive acceleration of a material interface can lead to complex fluid motions due to the Richtmyer–Meshkov (RM) instability. Here, the misalignment of pressure and density gradients deposits vorticity along the interface, which drives the flow and distorts the interface. At later times the flow may be receptive to secondary instabilities, most prominently the Kelvin–Helmholtz instability, which further increase the flow complexity and may trigger transition to turbulence.

Experimental investigations of impulsively accelerated interfaces have focused mainly on interfaces with single–mode perturbations, and on the case we consider here, shock–accelerated cylindrical gas columns [1, 2, 3]. The experiments are relatively inexpensive and turnaround times are short. The challenges are diagnostics and repeatability: after the shock passes, the flowfield evolution is driven by flow instabilities and vortex dynamics, which are sensitive to the initial conditions and noise in the system. Specifically, for the present case of a single shock–accelerated gas cylinder, the flow is dominated by a counter–rotating vortex pair.

Verification and validation are critical in the development of any simulation code, without which one can have little confidence that the code's results are meaningful.

The sensitivity and the complex evolution of the vortex pair are desirable properties for our primary purpose, which is to use the experiments to validate our simulation code. A well–designed, well–characterized, and accurately diagnosed experiment is essential for validation.

FLASH is a multi–species, multi–dimensional, parallel, adaptive–mesh–refinement, fluid dynamics code for applications in astrophysics [4]. Calder et al. discuss initial validation tests of the FLASH code [5]. Herein we continue our validation effort by comparing FLASH simulations to an experiment performed at the Los Alamos National Laboratories [2, 3].

2 Two–Dimensional Simulations

2.1 Experimental Facility and Initial Conditions

The experimental apparatus is a shock–tube with a 7.5 cm square cross–section. Gaseous SF_6 flows from an 8 mm diameter nozzle in the top wall of the shock–tube, forming a cylinder of dense gas in the otherwise air–filled test section. A Ma=1.2 shockwave travels through the shock–tube and passes through the cylinder. Our interest is in the resulting evolution of the SF_6. The experiment is nominally two–dimensional, and the experimental data are taken in a plane normal to the cylinder axis.

The initial SF_6 distribution (before the shock impact) is visualized by Rayleigh–Scattering from the SF_6 molecules [2]. The image plane is 2 cm below the top wall of the test section. As the SF_6 flows downward, air diffuses into the SF_6 column, reducing the peak concentration of the heavy gas. One limitation of the visualization technique is that the pixel intensity in the images gives only the mole fraction of SF_6 relative to the peak mole fraction. The scaling between pixel intensity and mole fraction is linear, with the proportionality constant specified by the maximum initial mole fraction of SF_6, denoted X_{SF6}.

After shock passage, two–dimensional velocity vectors in a plane are obtained using particle image velocimetry (PIV) [6]. The technique yields high resolution (spacing between vectors is about 187 microns) and high accuracy (measurement error is 1.5% of the structure convection velocity). Raw images are interrogated using a two-frame cross-correlation technique [7], which produces approximately 3600 vectors per realization. For PIV both the air and the SF_6 must be seeded with water/glycol droplets, nominally 0.5 μm in diameter, the displacement of which is used to obtain velocity data—hence, simultaneous velocity and composition images cannot be obtained.

The 608×468 pixel image of the initial SF_6 distribution is shown in Fig. 1. The pixel size is 38 microns when projected into the measurement plane. The pixel intensity is plotted, with 20 contours equally spaced between values of 5 and 165. The deformation of the contours indicates that the distribution of SF_6, as revealed by the diagnostics, is only approximately radially symmetric. Also, the signal is completely dominated by noise at the level of about 5–10% (the two lowest density contours

in Fig. 1). Since the asymmetries are likely to vary from one experimental shot to another and the flowfield evolution is highly sensitive to noise, smooth initial conditions for our simulations are obtained by fitting a radially–symmetric function to the experimental data.

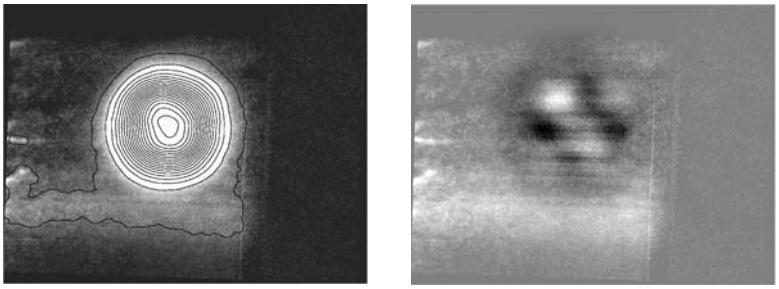


Fig. 1. (a) The initial conditions for the single–cylinder experiment with an 8mm nozzle. (b) Residuals between the experimental image and the composite Gaussian fit. The scale varies from -10 (black) to +10 (white) in intensity units. The maximum signal in the image has an intensity of about 165. Note a semi–regular non–radial m=4,l=4 component with maximum signal reaching about 10% of local intensity.

To obtain the smooth initial conditions we used the MINGA minimization package [8]. The fit extended out to a radius of 150 pixels from the center. After examining many trial functions, we selected the form:

$$C(r) = u_1 e^{(-r^2/u_2^2)} + u_3 e^{(-r^2/u_4^2)} + u_5 e^{(-(r-u_6)^2/u_7^2)} + u_8 e^{(-(r-u_9)^2/u_{10}^2)}$$

where $u_1 = 144.0725, u_2 = 69.45422, u_3 = 9.221577, u_4 = 20.10299, u_5 = 32.47960$, $u_6 = 42.59238, u_7 = 32.10067, u_8 = -1.559247, u_9 = 98.27106$, and $u_{10} = 15.51837$. (The length units are pixels, and the maximum intensity will be rescaled to X_{SF6}.) Residuals are shown in Fig. 1. The experimental data appears to contain a significant non–radial signal which can be characterized by an m=4, l=4 perturbation with an amplitude of about 10%. Our fit does not account for this additional component.

2.2 Overview of the Simulations

As the shock travels, it accelerates the medium through which it propagates. As the shock traverses the cylinder, vorticity is deposited baroclinically along the interface, i.e., due to the misalignment of the pressure gradient (normal to the shock) and the density gradient (normal to the interface.) The vorticity deposition is not uniform: it is maximum when the gradients are perpendicular, and since the shock is slowed in the SF_6, the maximum is shifted to the downstream portion of the cylinder edge. Once the shock has passed through the SF_6, vorticity generation due to the primary instability is complete. The existing vorticity drives the flow: a counter–rotating vortex pair forms, then rolls up. The vortex Reynolds number of the flow, as measured

experimentally, is Re$= \Gamma/\nu \approx 5 \times 10^4$. The development and evolution of the vortex pair and subsequent instabilities at the interface proceed in a weakly compressible regime. More precise descriptions can be found in the references [1, 9, 3].

Figure 2 shows a sequence of images from our baseline simulation. The minimum grid resolution is 78 microns, the initial peak mole fraction of SF_6 is 0.6, and the Courant (CFL) number is 0.8. (The CFL number is a nondimensional measure of the timestep size.) For all simulations, the streamwise and spanwise extent of the domain were 64 cm and 8 cm, respectively. Overall the flow features in the simulation results are similar to those in the experimentally obtained images. Next we will describe the effects of several simulation parameters on the computed results. The amount and location of small–scale structure, relative to the experimental data, is used as a qualitative metric.

Fig. 2. Evolution of the SF_6, with time elapsed after shock impact listed in μs. The mass fraction of SF_6 is shown, with X_{SF6}=0.6 and CFL=0.8.

2.3 Effects of Simulation Parameters

Effect of Maximum Initial Mole Fraction

Because the experimental images provide only information about the relative mole fractions, the maximum mole fraction of SF_6 at the start of the simulation is a free parameter in our initial conditions. We have focused on two values, X_{SF6}=0.8, motivated by experimental estimates, and X_{SF6}=0.6, better supported by comparison of simulation and experimental results. Simulation results are presented in Fig. 3. For X_{SF6}=0.8, the numerical solution shows excessive small–scale structure compared to the experimental data.

Effect of Grid Resolution

Using FLASH's adaptive mesh refinement capability, we have run simulations at three grid resolutions. Results at 750 μs are presented in Fig. 4, with minimum grid spacings of 156 microns, 78 microns and 39 microns. We observe that the amount of small–scale structure increases on finer grids. This can be understood since the numerical dissipation in FLASH's shock–capturing scheme (PPM) is resolution dependent, and no physical viscosity model was used for these simulations. (Estimates

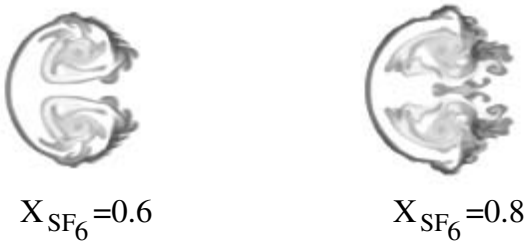


Fig. 3. SF_6 mass fraction at 750μs after shock impact. Initial maximum SF_6 mole fraction X_{SF6}=0.6 on the left, and on the right X_{SF6}=0.8.

of the length scale of molecular diffusion at the flow conditions of the experiment are below 10 microns.) At a grid resolution of 39 microns, the primary vortex cores are not easily identified among the diffuse, turbulent structures. At the lower resolutions, the two primary vortex cores are unambiguous.

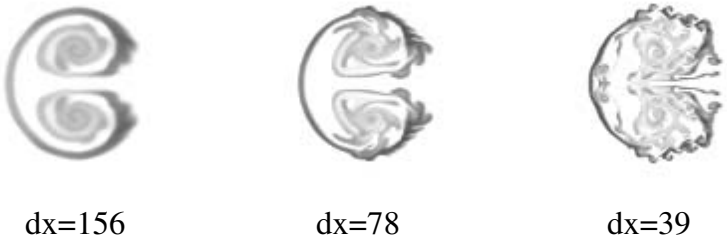


Fig. 4. SF_6 mass fraction at 750 μs with increasing grid resolution, labeled in microns. At the highest level of resolution, the vortex cores appear diffuse.

Flow–Mesh Interaction

It is known that unavoidable interpolation errors near discontinuous jumps in grid resolution can act as sources of spurious small–scale structure [10]. To test this possibility we have run simulations in which a predetermined area around the vortices is uniformly refined to the highest resolution. Compared to fully adaptive refinement (the default) this approach significantly reduces the amount of perturbations introduced by the grid adaption process.

In Fig. 5 we compare the results from a fully adaptive grid and grids with maximally refined rectangles of 3×3 cm, 4×4 cm, and 4×8 cm. The vortex structure is always less than 2 cm across. The results are shown at 750 μs after shock impact. For all grids, the minimum grid spacing is 78 microns and the CFL number is 0.8. For the different grids the large scale morphology, such as size of the cylinder cross–section and the basic vortex structure, remains the same. However, the shape of the

cross–section visibly differs depending on the grid, as does the amount and location of small–scale structures. In particular, differences are noticeable in the small–scale instabilities present on the vortex rolls. Since all other simulation aspects are the same, the differences must originate with perturbations at jumps in refinement.

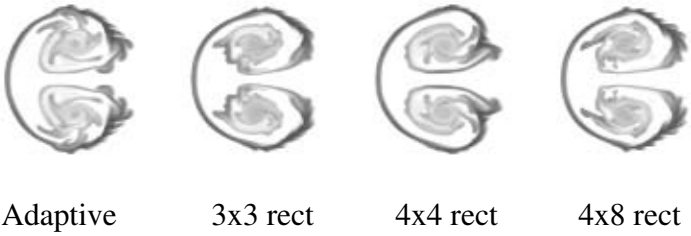


Fig. 5. Solutions on different grids, 750 μs after shock impact at CFL=0.8. Left to right: fully adaptive grid; 3×3 cm refined rectangle; 4×4 cm refined rectangle; 4×8 cm refined rectangle. In the rightmost image, the refined rectangle covers the entire spanwise extent of the test section.

Effect of Courant Number

The timestep in an explicit hydrodynamic code is limited by the CFL number. In general we use a value of CFL=0.8, but we have also performed simulations with the time step size limited by CFL=0.2. Reducing the timestep generally reduces the temporal truncation error; however, it might have an adverse effect on the spatial error. Additionally the mesh can adapt more often per unit of simulation time, and if errors are committed every time the mesh adapts, this could lead to a less accurate solution.

We repeated the simulations described above, but at CFL=0.2. The results are shown in Fig. 6. The simulations at CFL=0.2 are much less sensitive to grid adaption: there is much less variation between solutions on adaptive and locally uniform grids at CFL=0.2 than at CFL=0.8. One explanation for these results is that the errors at the fine–coarse boundaries are larger and lead to stronger perturbations at higher CFL numbers. An alternative explanation is that at higher Courant numbers, PPM does not adequately compute solutions at these conditions. Our simulations indicate that for FLASH, a lower CFL number is preferred in this regime.

Our fully adaptive grid simulations provides a speed–up factor of about 10, compared to the grids with the 4×8 cm maximally refined rectangle. Such large savings demonstrate the advantages of adaptive mesh refinement. At the same time, caution is warranted: our results also demonstrate that AMR generates perturbations which, depending on other simulation parameters and the flow regime, can give rise to spurious small–scale structures.

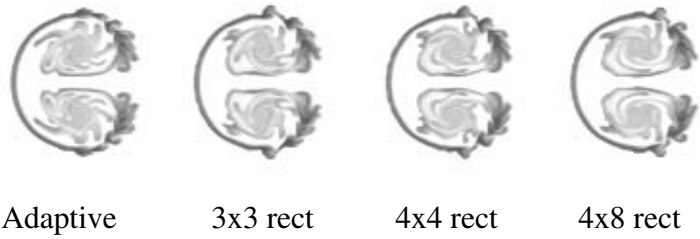


Fig. 6. Solutions on different grids for CFL=0.2; for other details see Fig. 5.

2.4 Metrics for Comparison to Experimental Data

Evolution of Cylinder Size

In Fig. 7 we present integral scale measures – the streamwise and spanwise extent of the SF_6 – over time. The contour of SF_6 mass fraction equal to 0.1 was used to define the edge of the SF_6. We plot results for our baseline simulation and simulations where a single parameter is varied relative to the baseline. For a given maximum initial SF_6 mole fraction, the cylinder height and width are essentially insensitive to the parameters varied. These integral measures provide a basis for comparison of our simulation results to those of others (e.g. [3, 11]), as well as to the experimental data.

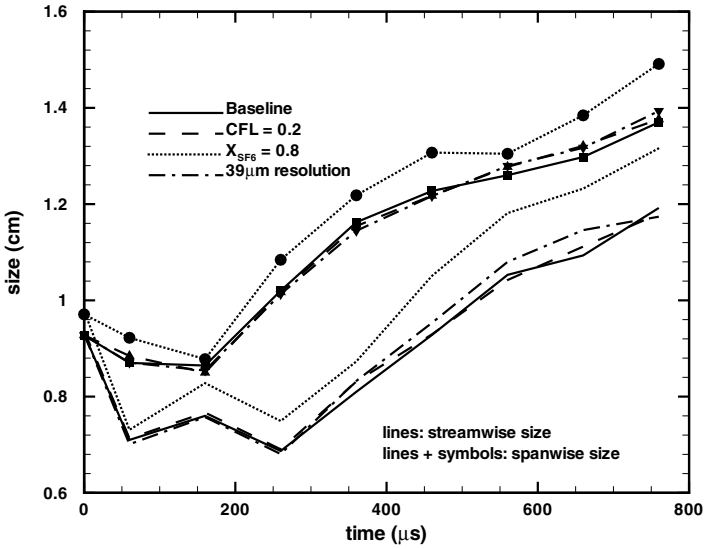


Fig. 7. Integral scale measures plotted at nine simulation times. For the baseline case, X_{SF6}=0.6, CFL=0.8, and 78 micron resolution. For other cases a single parameter varies from the baseline case.

Velocity Comparisons

We compare the magnitude of the velocity fluctuations from the experimental data to that from the simulation results. The velocity fluctuation is defined as the velocity in the frame of reference of the vortices. To find the velocity fluctuation we subtract the convective velocity, which we define as the streamwise velocity component in the lab frame at the point of the maximum vorticity.

The experimental data are the velocity components and the corresponding vorticity on 60×60 points uniformly covering a 12×12 mm region around one of the primary vortices. We find the convective velocity to be 101.25 m/s. The magnitude of the velocity fluctuations are plotted as flooded contours in the left plot of Fig. 8, together with streamlines of the velocity fluctuations. The plot is oriented such that the shock has passed through the image from top to bottom, and the centerline (between the primary vortices) is near the right edge of the plot.

The right plot of Fig. 8 shows the same quantities but from the simulation data (78 micron resolution, X_{SF6}=0.6, CFL=0.2), for which the convective velocity is 92.90 m/s. The velocity fluctuation field in the simulations compares reasonably well to the experimental data. The maximum and minimum fluctuations occur in approximately the same places. The velocities in the simulation are up to 10% higher than in the experiment. The simulation data shows more structure, and there is some (visual, at least) correlation between the structure in the velocity fluctuations and the SF_6 distribution.

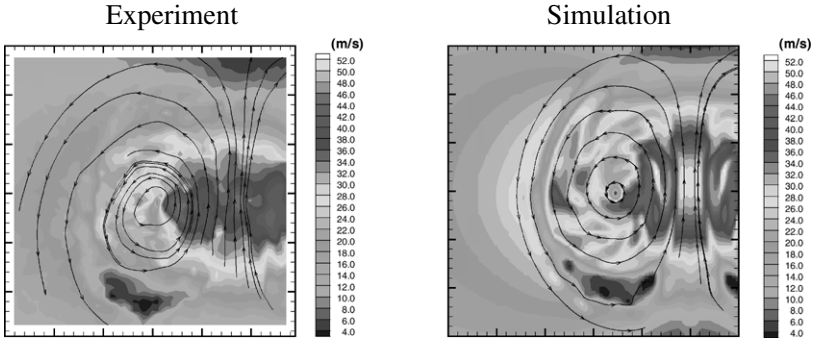


Fig. 8. Velocity fluctuation magnitude and streamlines around one primary vortex 750 μs after shock impact. The shock has passed through the image from the top to bottom. Left: experimental data. Right: simulation data.

3 Three–Dimensional Simulation

While the experiment is nominally two–dimensional, several factors might contribute to non–negligible three–dimensional effects. Probably the most significant experimental deviations from two–dimensionality are due to the diffusion of the SF_6 as it flows from the top of the test section to the bottom. As a result, the mixture composition and density vary with height. (This is why we assume the maximum initial mole fraction of SF_6 is less than 1.0 in the two–dimensional simulations.) Even if the cylinder were invariant with height, the flowfield after the shock passes the cylinder is dominated by vorticity dynamics, and small scale structures arising from flow instabilities eventually develop at the vortex edges. The effects of such inherently three–dimensional phenomena cannot be captured in two–dimensional simulations.

We executed a speculative three–dimensional simulation which, though it cannot definitively establish that three–dimensional effects are essential for the shock–cylinder interaction, compels further experimental and computational investigation. Our z-direction extension of the initial conditions is purely ad hoc, because we have no corresponding experimental data. Of course, this simulation cannot be used as a validation test for the FLASH code, but we hope it will open a new line of investigation and discussion.

For this simulation the z-direction is parallel to the cylinder axis, with z=0.0 cm at the bottom wall and z=8.0cm at the top wall. To initialize the flowfield, we begin with the "raw" image in Fig. 1. Only after this simulation was completed did we learn that the raw images have spurious high frequency noise, and smooth approximations to the raw data are believed to better represent the true SF_6 field [2]. The streamwise (x) and spanwise (y) dimensions of the image are rescaled linearly with z, so the SF_6 covers a smaller area at the top wall and a larger area at the bottom. Consequently the maximum mole fraction of SF_6 in each plane varies as z^2, and is 0.64 at the top wall and 0.47 at the z=6.0 cm plane, at which the "raw" image was obtained. Otherwise the initialization is the same as for the two–dimensional simulations. The simulation was run at CFL=0.8 using fully adaptive mesh refinement, and the minimum grid spacing was 156 microns in all three spatial dimensions.

Near the top wall in our simulation, the vortices have rolled up more than at the bottom, and more small scale structure has developed. This observation appears to hold throughout the course of the simulation, and can be understood as follows. The rescaling of the initial SF_6 distribution results in larger composition gradients, and correspondingly larger density gradients, near the top and smaller gradients near the bottom. The primary source of vorticity generation in this problem is through baroclinic torque, so more vorticity is deposited near the upper wall, where the density gradients are largest.

The circulation provides a quantitative measure of the vortex development. The z-component of circulation was calculated for the lower-y half of each xy-plane of the simulation, bounded by the centerline, y=0.0 cm, and the inflow and outflow boundaries. Figure 9 shows the circulation at z=0.0, 2.0, 4.0, 6.0 and 8.0 cm as functions of time. The profiles are essentially the same up to 300 μs after the shock impact, with the magnitude in each plane increasing from the bottom wall to the top wall;

this suggests that there are no significant three–dimensional effects through this time other than the diffusion of the SF_6 as it flows vertically. However, after 300 μs, differences between the profiles for each height appear, beginning with the profile near the top wall, and eventually spreading to lower heights.

We find that by the end of the simulation, the z-component of velocity has reached a maximum of 17 m/s, which is more than half of the maximum of the spanwise component. Positive z-velocity is maximum in the vortex cores, away from the upper and lower walls. Apparently, the stronger vortices near the upper wall have correspondingly lower core pressures compared to the weaker vortices near the lower wall; this pressure difference accelerates gas toward the upper wall through the vortex cores. The acceleration is compounded because the cores of the vortices are filled with air, and the heavier SF_6 is wrapped around the outside. The lighter air in the core of each vortex is inside a tube of SF_6, and is preferentially accelerated toward the upper wall. The different circulation profiles and large z-component of velocity suggest that three–dimensional effects are not negligible for the initial conditions we assumed. Only with experimental data can we test those assumptions.

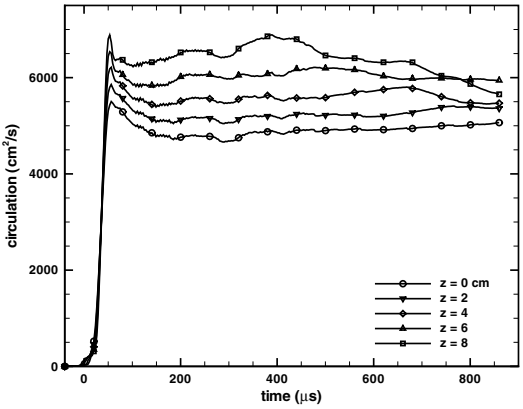


Fig. 9. Z–component of circulation in the lower-y half of the xy-plane at five vertical locations in the tunnel, as a function of time.

4 Concluding Remarks

To date we have made a large number of two–dimensional simulations to validate the FLASH code for problems dominated by vortex dynamics. While this is work in progress, we can make the following remarks:

- The overall morphology of the flowfield is captured by the simulations, but differences exist in the location and extent of small–scale structure.
- Simulation velocity magnitudes lie within 10% of experimental values.

- For vortex–dominated subsonic flows, FLASH users should be cautious regarding the choice of CFL number, mesh refinement criteria, and if PPM is used, contact steepening.

Our three–dimensional simulation, despite issues with the initial conditions, suggests that three–dimensional effects might be important for this experiment, and measurements should be made parallel to the cylinder axis to examine the issue.

References

1. J. W. Jacobs. The dynamics of shock accelerated light and heavy gas cylinders. *Physics of Fluids A*, 5(9):2239–2247, September 1993.
2. C. Tomkins et al. A quantitive study of the interaction of two Richtmyer-Meshkov-unstable gas cylinders. *Physics of Fluids*, 15(4):986–1004, April 2003.
3. C. A. Zoldi. *A Numerical and Experimental Study of a Shock-Accelerated Heavy Gas Cylinder*. PhD thesis, State University of New York at Stony Brook, 2002.
4. B. Fryxell et al. FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *ApJS*, 131:273–334, November 2000.
5. A. C. Calder et al. On validating an astrophysical simulation code. *ApJ Supplement Series*, 143:201–229, November 2002.
6. K. Prestridge et al. Validation of an instability growth model using particle image velocimetry measurements. *Phys. Rev. Lett.*, 84(19):4353–4356, May 2000.
7. K. T. Christensen et al. PIV sleuth: Integrated particle image velocimetry (PIV) interrogation/validation software. TAM report 943, UIUC, 2000.
8. T. Plewa. Parameter fitting problem for contact binaries. *Acta Astronomica*, 38:415–430, 1988.
9. J. J. Quirk and S. Karni. On the dynamics of a shock-bubble interaction. *Journal of Fluid Mechanics*, 318:129–163, 1996.
10. J. J. Quirk. *An Adaptive Mesh Refinement Algorithm for Computational Shock Hydrodynamics*. PhD thesis, Cranfield Institute of Technology, UK, 1991.
11. W. J. Rider et al. Statistical comparison between experiments and numerical simulations of shock-accelerated gas cylinders. In H. A. Mang et al., editor, *Proceedings of the Fifth World Congress on Computational Mechanics*, 2002.

Color Plates

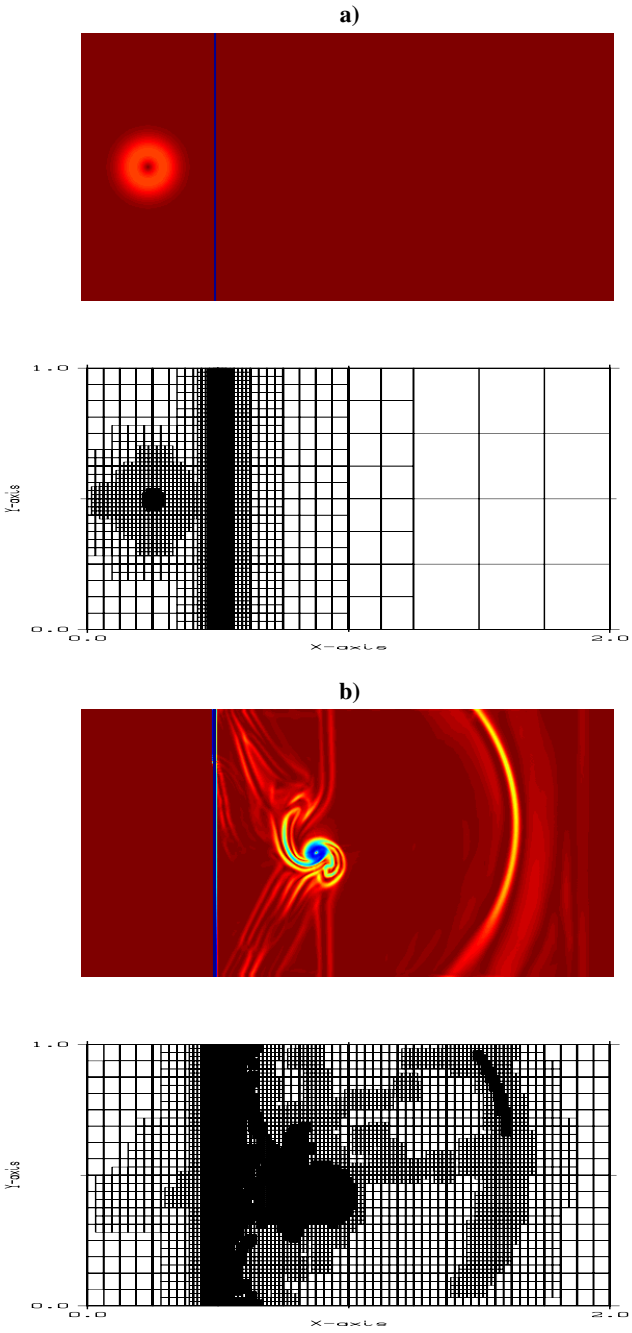


Plate 1. [Fig. 8 on p. 155] Schlieren pictures of the density field of a Mach 4 shock-vortex interaction and associated multilevel grids. a) time $t = 0$, b) time $t = 0.4$.

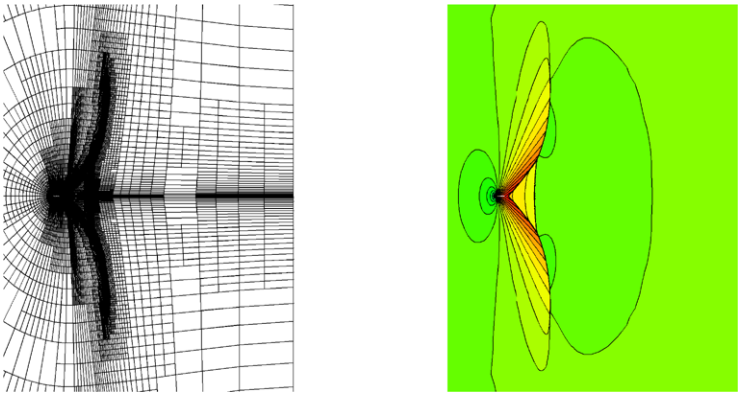


Plate 2. [Fig. 9 on p. 156] Total view of NACA0012 airfoil, $M = 0.95$, $\alpha = 0.0°$. *Left*: Computational grid. *Right*: Pressure distribution, $M_{min} = 0.0$, $M_{max} = 1.45$, $\Delta M = 0.05$.

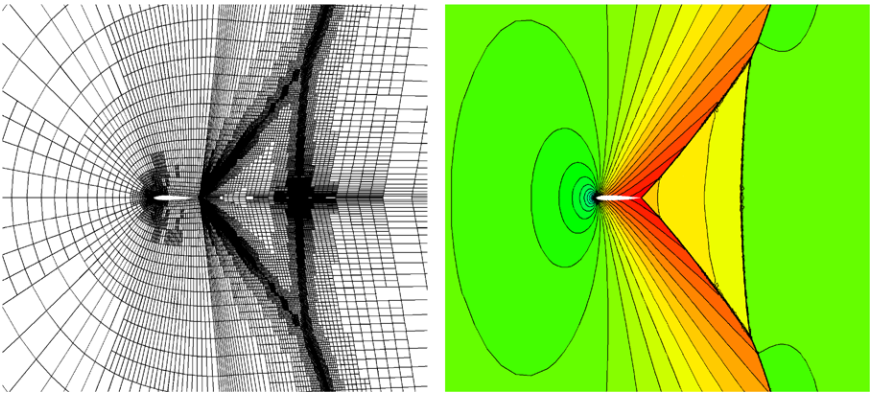


Plate 3. [Fig. 10 on p. 156] Detail of Fig. 9.

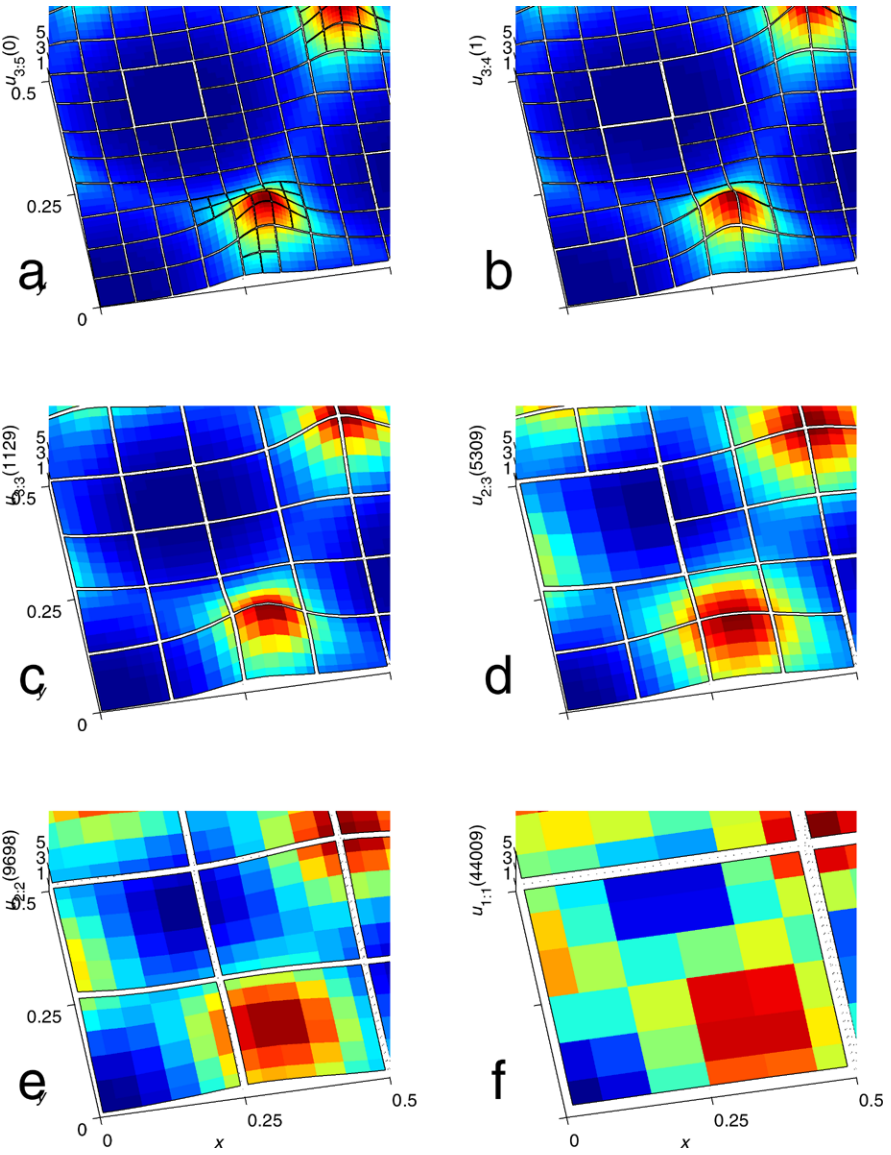


Plate 4. [Fig. 1 on p. 168] Surface plots of $u(t,x,y)$ for $(x,y) \in [0,\frac{1}{2}]^2$, from solving the HeatEq. (18-19), on adaptive piecewise-polynomial spaces $\mathbb{V}_k^2$ (15) corresponding to elements $\mathbb{E}_k$ (13) that cover the biperiodic domain $[0,1]^2$. Each $\mathbb{E}_k$ contains 7^2 Gauss quadrature nodes $\vec{x}_{\vec{j},k}$ (16). The plots (a-f) correspond to times t in the rows of Table 1. Each vertical plot axis is labeled $u_{\ell_{\min}:\ell_{\max}}(n)$ to indicate the smallest and largest element edges $2^{-\ell_{\max}}$ and $2^{-\ell_{\min}}$ at time step n. In (19) were set $e^{-a_1} = e^{-a_2} = \frac{2}{5}$, $\vec{l} = 2\pi(1,2)$ and $\vec{x}_o = (\frac{1}{2},\frac{1}{2})$. Observe that in (a) the peaks (red) generate locally small-scale representations, that evolve with time to locally larger scales.

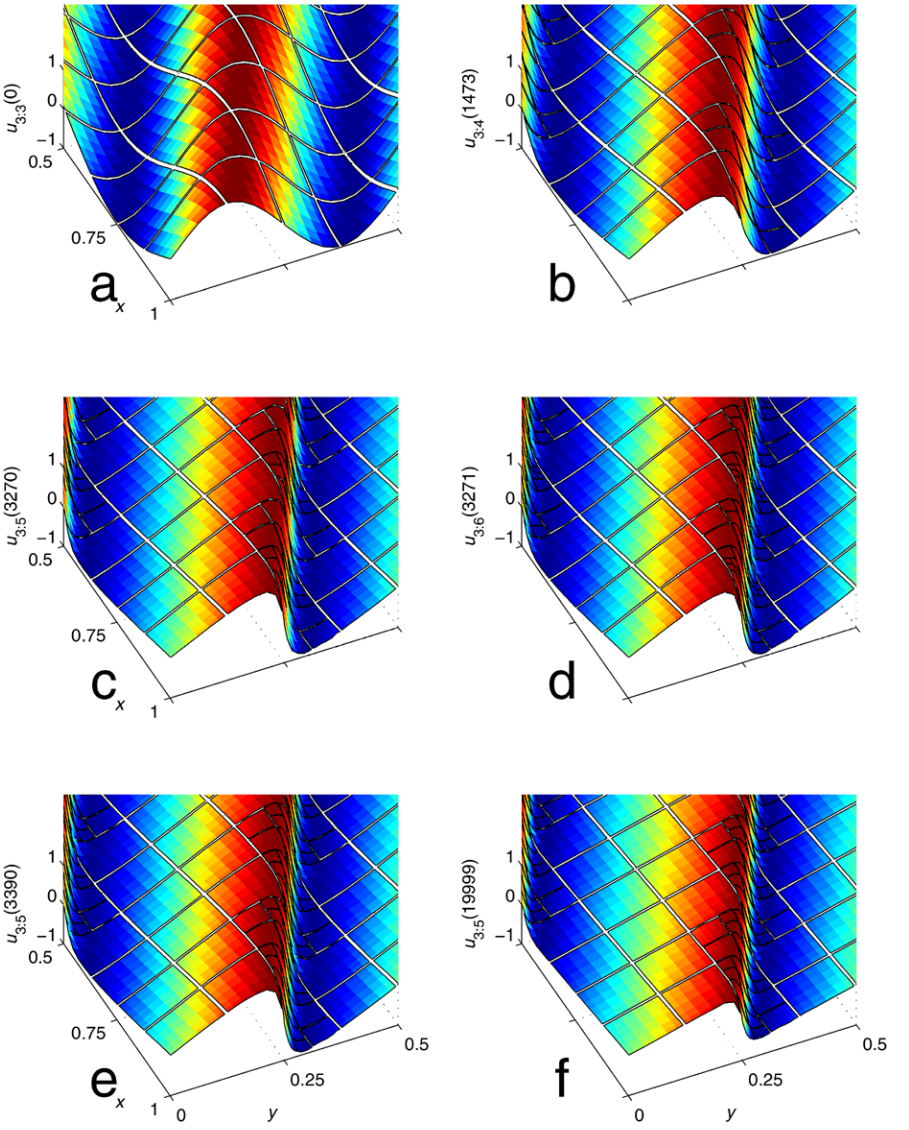
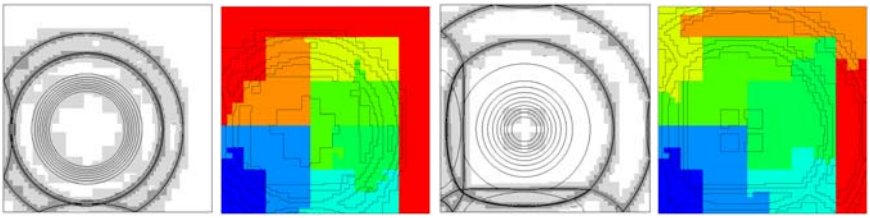
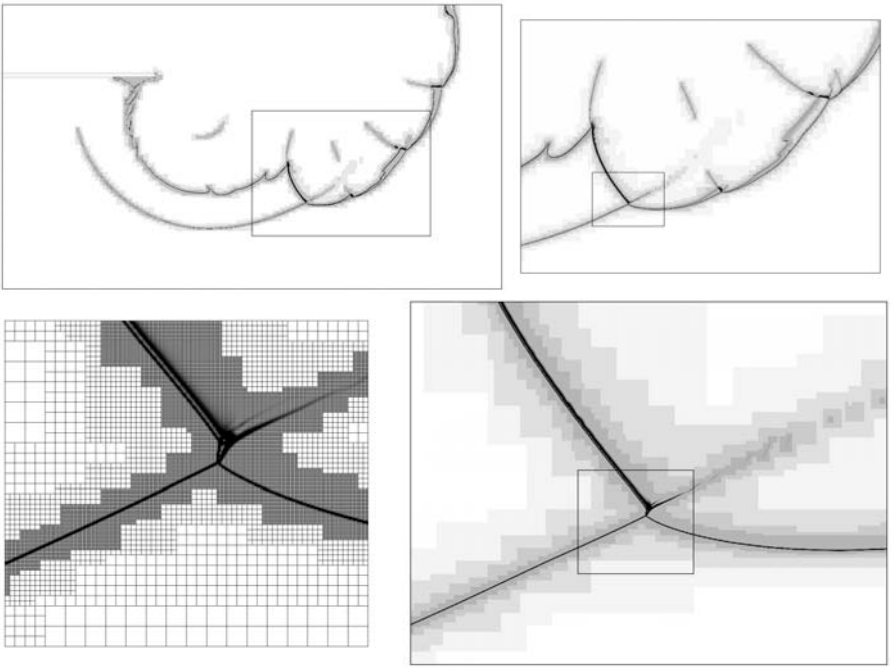


Plate 5. [Fig. 2 on p. 169] As in Fig. 1 but for $u_x = \partial_x \chi$ from the Burgers Eq. (21-22), zoomed to view $(x,y) \in [\frac{1}{2},1] \times [0,\frac{1}{2}]$. Observe the formation over time of steep fronts along the line family $\vec{l}\cdot\vec{x} \in (2\mathbb{Z}+1)\pi$, e.g., the line $1-x = 2y - \frac{1}{2}$ is prominent in this view. This behavior is no more than the rotation of 1D Burgers dynamics into 2D, since if $U(t,x)$ solves the 1D Burgers Eq., then $\vec{u}(t,\vec{x}) = \vec{l}U(l^2 t, \vec{l}\cdot\vec{x})$ solves (20).



After 38 time steps After 79 time steps

Plate 6. [Fig. 4 on p. 370] Circular Riemann problem in an enclosed box. Isolines of density on two refinement levels (indicated by gray scales) and distribution to eight nodes (indicated by different colors).



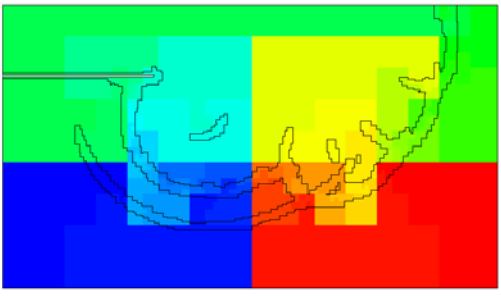


Plate 7. [Fig. 5 on p. 370] (Upper four graphics.) Planar detonation diffraction. Density distribution on four refinement levels 240μs after the detonation has left the tube. Multiple zooms are necessary to display the finite volume cells.

Plate 8. [Fig. 6 on p. 370] Planar detonation diffraction. Distribution of computational domain to 48 nodes.

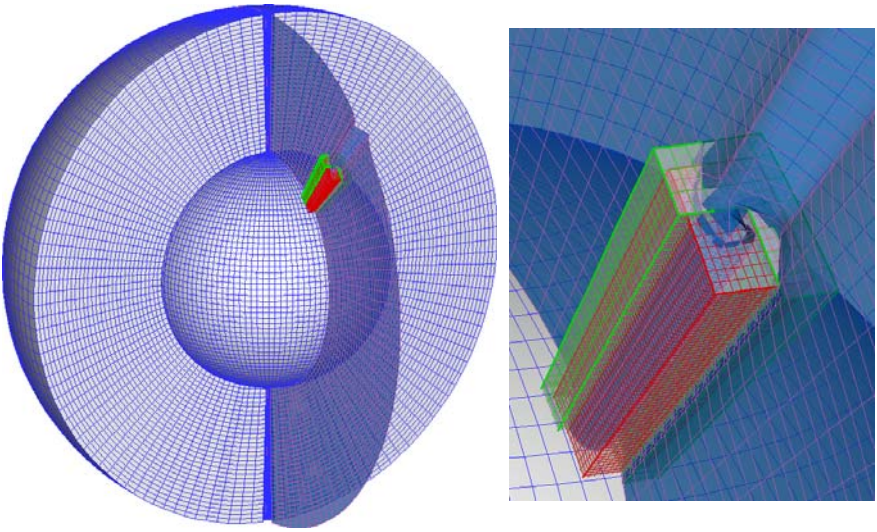


Plate 9. [Fig. 6 on p. 516] Global mesh arrangement and the iso-surface of the solution for the kinematic cyclogenecis problem, for a static grid case, where the grid has been refined up to a specific height (left). Some of the surfaces have been removed in the zoom (right) to reveal detail. This mesh arrangement is suitable for mesoscale studies, as a two-way interaction replacement of nested grids.

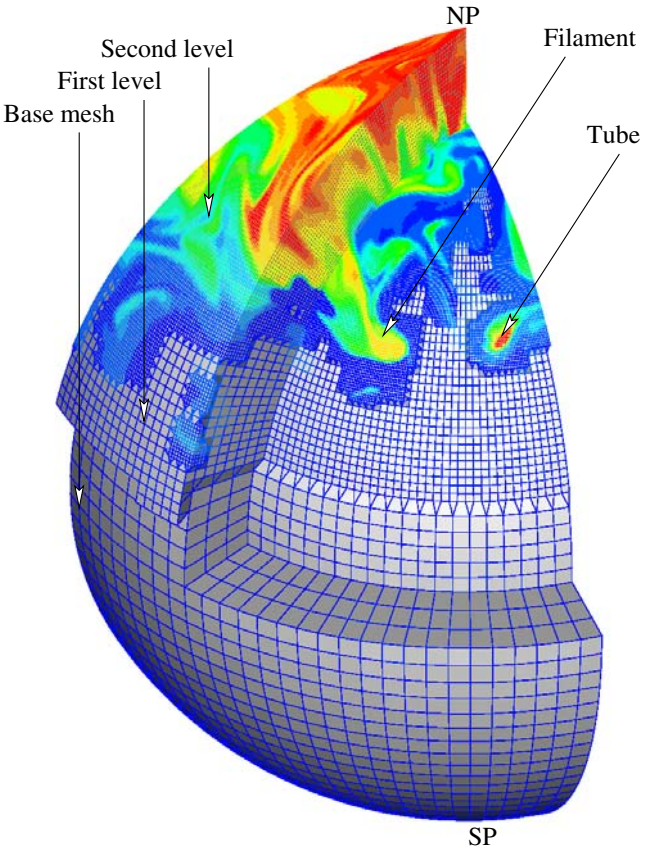


Plate 10. [Fig. 8 on p. 520] Three dimensional simulation of a stratosphere-troposphere event over the Atlantic during June 1996. Note that although the grid appears to contain triangular cells, this is nothing but a graphical artefact (see main text).

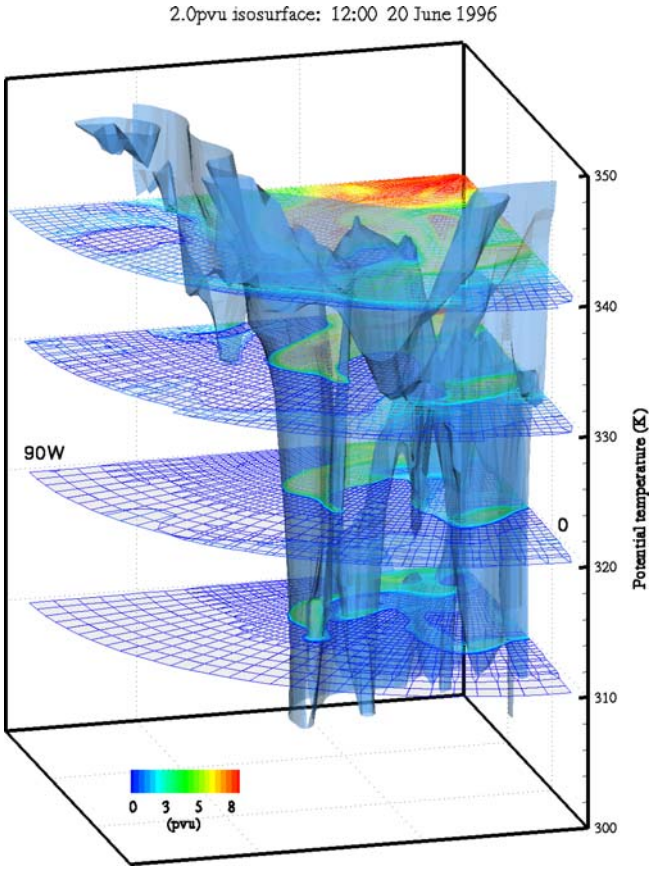


Plate 11. [Fig. 9 on p. 521] Snapshot of the evolution of an isolated cyclonic vortex (cut-off low) over the Atlantic (90 degrees west to Greenwich Meridian), on 20 June 1996. The PV = 2.0 isosurface is shown; the computational meshes are shaded by PV magnitude. Global atmospheric AMR run on a $120 \times 60 \times 8(\times 2 \times 3)$ grid, between 290K and 370K.

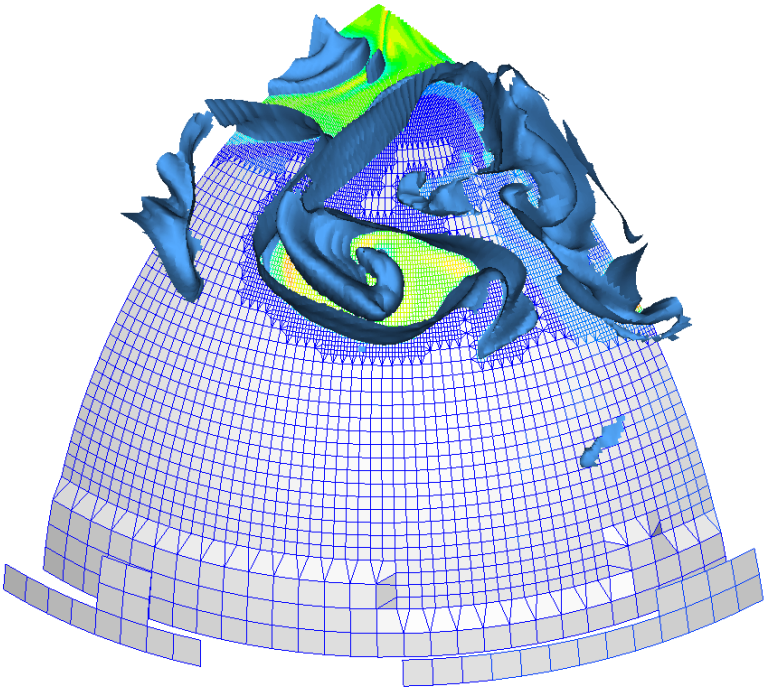


Plate 12. [Fig. 10 on p. 522] Visualisation of a cut-off low over the Atlantic, 21 June 1996. The PV=2.0 isosurface is shown; the computational meshes are shaded by PV magnitude. For details of this integration see the caption in figure 9.

Index

Editorial Policy

§1. Volumes in the following three categories will be published in LNCSE:

i) Research monographs
ii) Lecture and seminar notes
iii) Conference proceedings

Those considering a book which might be suitable for the series are strongly advised to contact the publisher or the series editors at an early stage.

§2. Categories i) and ii). These categories will be emphasized by Lecture Notes in Computational Science and Engineering. **Submissions by interdisciplinary teams of authors are encouraged.** The goal is to report new developments – quickly, informally, and in a way that will make them accessible to non-specialists. In the evaluation of submissions timeliness of the work is an important criterion. Texts should be well-rounded, well-written and reasonably self-contained. In most cases the work will contain results of others as well as those of the author(s). In each case the author(s) should provide sufficient motivation, examples, and applications. In this respect, Ph.D. theses will usually be deemed unsuitable for the Lecture Notes series. Proposals for volumes in these categories should be submitted either to one of the series editors or to Springer-Verlag, Heidelberg, and will be refereed. A provisional judgment on the acceptability of a project can be based on partial information about the work: a detailed outline describing the contents of each chapter, the estimated length, a bibliography, and one or two sample chapters – or a first draft. A final decision whether to accept will rest on an evaluation of the completed work which should include

– at least 100 pages of text;
– a table of contents;
– an informative introduction perhaps with some historical remarks which should be accessible to readers unfamiliar with the topic treated;
– a subject index.

§3. Category iii). Conference proceedings will be considered for publication provided that they are both of exceptional interest and devoted to a single topic. One (or more) expert participants will act as the scientific editor(s) of the volume. They select the papers which are suitable for inclusion and have them individually refereed as for a journal. Papers not closely related to the central topic are to be excluded. Organizers should contact Lecture Notes in Computational Science and Engineering at the planning stage.

In exceptional cases some other multi-author-volumes may be considered in this category.

§4. Format. Only works in English are considered. They should be submitted in camera-ready form according to Springer-Verlag's specifications.
Electronic material can be included if appropriate. Please contact the publisher.
Technical instructions and/or TEX macros are available via
http://www.springeronline.com/sgw/cda/frontpage/0,10735,5-111-2-71391-0,00.html
The macros can also be sent on request.

General Remarks

Lecture Notes are printed by photo-offset from the master-copy delivered in camera-ready form by the authors. For this purpose Springer-Verlag provides technical instructions for the preparation of manuscripts. See also *Editorial Policy*.

Careful preparation of manuscripts will help keep production time short and ensure a satisfactory appearance of the finished book.

The following terms and conditions hold:

Categories i), ii), and iii):
Authors receive 50 free copies of their book. No royalty is paid. Commitment to publish is made by letter of intent rather than by signing a formal contract. Springer-Verlag secures the copyright for each volume.

For conference proceedings, editors receive a total of 50 free copies of their volume for distribution to the contributing authors.

All categories:
Authors are entitled to purchase further copies of their book and other Springer mathematics books for their personal use, at a discount of 33,3 % directly from Springer-Verlag.

Addresses:

Timothy J. Barth
NASA Ames Research Center
NAS Division
Moffett Field, CA 94035, USA
e-mail: barth@nas.nasa.gov

Michael Griebel
Institut für Angewandte Mathematik
der Universität Bonn
Wegelerstr. 6
53115 Bonn, Germany
e-mail: griebel@iam.uni-bonn.de

David E. Keyes
Department of Applied Physics
and Applied Mathematics
Columbia University
200 S. W. Mudd Building
500 W. 120th Street
New York, NY 10027, USA
e-mail: david.keyes@columbia.edu

Risto M. Nieminen
Laboratory of Physics
Helsinki University of Technology
02150 Espoo, Finland
e-mail: rni@fyslab.hut.fi

Dirk Roose
Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven-Heverlee, Belgium
e-mail: dirk.roose@cs.kuleuven.ac.be

Tamar Schlick
Department of Chemistry
Courant Institute of Mathematical
Sciences
New York University
and Howard Hughes Medical Institute
251 Mercer Street
New York, NY 10012, USA
e-mail: schlick@nyu.edu

Springer-Verlag, Mathematics Editorial IV
Tiergartenstrasse 17
D-69121 Heidelberg, Germany
Tel.: *49 (6221) 487-8185
Fax: *49 (6221) 487-8355
e-mail: Martin.Peters@springer-sbm.com

Lecture Notes in Computational Science and Engineering

Vol. 1 D. Funaro, *Spectral Elements for Transport-Dominated Equations.* 1997. X, 211 pp. Softcover. ISBN 3-540-62649-2

Vol. 2 H. P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming. 1999. XXIII, 682 pp. Hardcover. ISBN 3-540-65274-4

Vol. 3 W. Hackbusch, G. Wittum (eds.), *Multigrid Methods V.* Proceedings of the Fifth European Multigrid Conference held in Stuttgart, Germany, October 1-4, 1996. 1998. VIII, 334 pp. Softcover. ISBN 3-540-63133-X

Vol. 4 P. Deuflhard, J. Hermans, B. Leimkuhler, A. E. Mark, S. Reich, R. D. Skeel (eds.), *Computational Molecular Dynamics: Challenges, Methods, Ideas.* Proceedings of the 2nd International Symposium on Algorithms for Macromolecular Modelling, Berlin, May 21-24, 1997. 1998. XI, 489 pp. Softcover. ISBN 3-540-63242-5

Vol. 5 D. Kröner, M. Ohlberger, C. Rohde (eds.), *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws.* Proceedings of the International School on Theory and Numerics for Conservation Laws, Freiburg / Littenweiler, October 20-24, 1997. 1998. VII, 285 pp. Softcover. ISBN 3-540-65081-4

Vol. 6 S. Turek, *Efficient Solvers for Incompressible Flow Problems.* An Algorithmic and Computational Approach. 1999. XVII, 352 pp, with CD-ROM. Hardcover. ISBN 3-540-65433-X

Vol. 7 R. von Schwerin, ***M**ulti **B**ody **S**ystem **SIM**ulation.* Numerical Methods, Algorithms, and Software. 1999. XX, 338 pp. Softcover. ISBN 3-540-65662-6

Vol. 8 H.-J. Bungartz, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.* Proceedings of the International FORTWIHR Conference on HPSEC, Munich, March 16-18, 1998. 1999. X, 471 pp. Softcover. 3-540-65730-4

Vol. 9 T. J. Barth, H. Deconinck (eds.), *High-Order Methods for Computational Physics.* 1999. VII, 582 pp. Hardcover. 3-540-65893-9

Vol. 10 H. P. Langtangen, A. M. Bruaset, E. Quak (eds.), *Advances in Software Tools for Scientific Computing.* 2000. X, 357 pp. Softcover. 3-540-66557-9

Vol. 11 B. Cockburn, G. E. Karniadakis, C.-W. Shu (eds.), *Discontinuous Galerkin Methods.* Theory, Computation and Applications. 2000. XI, 470 pp. Hardcover. 3-540-66787-3

Vol. 12 U. van Rienen, *Numerical Methods in Computational Electrodynamics.* Linear Systems in Practical Applications. 2000. XIII, 375 pp. Softcover. 3-540-67629-5

Vol. 13 B. Engquist, L. Johnsson, M. Hammill, F. Short (eds.), *Simulation and Visualization on the Grid.* Parallelldatorcentrum Seventh Annual Conference, Stockholm, December 1999, Proceedings. 2000. XIII, 301 pp. Softcover. 3-540-67264-8

Vol. 14 E. Dick, K. Riemslagh, J. Vierendeels (eds.), *Multigrid Methods VI.* Proceedings of the Sixth European Multigrid Conference Held in Gent, Belgium, September 27-30, 1999. 2000. IX, 293 pp. Softcover. 3-540-67157-9

Vol. 15 A. Frommer, T. Lippert, B. Medeke, K. Schilling (eds.), *Numerical Challenges in Lattice Quantum Chromodynamics.* Joint Interdisciplinary Workshop of John von Neumann Institute for Computing, Jülich and Institute of Applied Computer Science, Wuppertal University, August 1999. 2000. VIII, 184 pp. Softcover. 3-540-67732-1

Vol. 16 J. Lang, *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems.* Theory, Algorithm, and Applications. 2001. XII, 157 pp. Softcover. 3-540-67900-6

Vol. 17 B. I. Wohlmuth, *Discretization Methods and Iterative Solvers Based on Domain Decomposition.* 2001. X, 197 pp. Softcover. 3-540-41083-X

Vol. 18 U. van Rienen, M. Günther, D. Hecht (eds.), *Scientific Computing in Electrical Engineering.* Proceedings of the 3rd International Workshop, August 20-23, 2000, Warnemünde, Germany. 2001. XII, 428 pp. Softcover. 3-540-42173-4

Vol. 19 I. Babuška, P. G. Ciarlet, T. Miyoshi (eds.), *Mathematical Modeling and Numerical Simulation in Continuum Mechanics.* Proceedings of the International Symposium on Mathematical Modeling and Numerical Simulation in Continuum Mechanics, September 29 - October 3, 2000, Yamaguchi, Japan. 2002. VIII, 301 pp. Softcover. 3-540-42399-0

Vol. 20 T. J. Barth, T. Chan, R. Haimes (eds.), *Multiscale and Multiresolution Methods.* Theory and Applications. 2002. X, 389 pp. Softcover. 3-540-42420-2

Vol. 21 M. Breuer, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.* Proceedings of the 3rd International FORTWIHR Conference on HPSEC, Erlangen, March 12-14, 2001. 2002. XIII, 408 pp. Softcover. 3-540-42946-8

Vol. 22 K. Urban, *Wavelets in Numerical Simulation.* Problem Adapted Construction and Applications. 2002. XV, 181 pp. Softcover. 3-540-43055-5

Vol. 23 L. F. Pavarino, A. Toselli (eds.), *Recent Developments in Domain Decomposition Methods.* 2002. XII, 243 pp. Softcover. 3-540-43413-5

Vol. 24 T. Schlick, H. H. Gan (eds.), *Computational Methods for Macromolecules: Challenges and Applications.* Proceedings of the 3rd International Workshop on Algorithms for Macromolecular Modeling, New York, October 12-14, 2000. 2002. IX, 504 pp. Softcover. 3-540-43756-8

Vol. 25 T. J. Barth, H. Deconinck (eds.), *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics.* 2003. VII, 344 pp. Hardcover. 3-540-43758-4

Vol. 26 M. Griebel, M. A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations*. 2003. IX, 466 pp. Softcover. 3-540-43891-2

Vol. 27 S. Müller, *Adaptive Multiscale Schemes for Conservation Laws*. 2003. XIV, 181 pp. Softcover. 3-540-44325-8

Vol. 28 C. Carstensen, S. Funken, W. Hackbusch, R. H. W. Hoppe, P. Monk (eds.), *Computational Electromagnetics*. Proceedings of the GAMM Workshop on "Computational Electromagnetics", Kiel, Germany, January 26-28, 2001. 2003. X, 209 pp. Softcover. 3-540-44392-4

Vol. 29 M. A. Schweitzer, *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*. 2003. V, 194 pp. Softcover. 3-540-00351-7

Vol. 30 T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders (eds.), *Large-Scale PDE-Constrained Optimization*. 2003. VI, 349 pp. Softcover. 3-540-05045-0

Vol. 31 M. Ainsworth, P. Davies, D. Duncan, P. Martin, B. Rynne (eds.), *Topics in Computational Wave Propagation*. Direct and Inverse Problems. 2003. VIII, 399 pp. Softcover. 3-540-00744-X

Vol. 32 H. Emmerich, B. Nestler, M. Schreckenberg (eds.), *Interface and Transport Dynamics*. Computational Modelling. 2003. XV, 432 pp. Hardcover. 3-540-40367-1

Vol. 33 H. P. Langtangen, A. Tveito (eds.), *Advanced Topics in Computational Partial Differential Equations*. Numerical Methods and Diffpack Programming. 2003. XIX, 658 pp. Softcover. 3-540-01438-1

Vol. 34 V. John, *Large Eddy Simulation of Turbulent Incompressible Flows*. Analytical and Numerical Results for a Class of LES Models. 2004. XII, 261 pp. Softcover. 3-540-40643-3

Vol. 35 E. Bänsch (ed.), *Challenges in Scientific Computing - CISC 2002*. Proceedings of the Conference *Challenges in Scientific Computing*, Berlin, October 2-5, 2002. 2003. VIII, 287 pp. Hardcover. 3-540-40887-8

Vol. 36 B. N. Khoromskij, G. Wittum, *Numerical Solution of Elliptic Differential Equations by Reduction to the Interface*. 2004. XI, 293 pp. Softcover. 3-540-20406-7

Vol. 37 A. Iske, *Multiresolution Methods in Scattered Data Modelling*. 2004. XII, 182 pp. Softcover. 3-540-20479-2

Vol. 38 S.-I. Niculescu, K. Gu (eds.), *Advances in Time-Delay Systems*. 2004. XIV, 446 pp. Softcover. 3-540-20890-9

Vol. 39 S. Attinger, P. Koumoutsakos (eds.), *Multiscale Modelling and Simulation*. 2004. VIII, 277 pp. Softcover. 3-540-21180-2

Vol. 40 R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Wildlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering*. 2005. XVIII, 690 pp. Softcover. 3-540-22523-4

Vol. 41 T. Plewa, T. Linde, V.G. Weirs (eds.), *Adaptive Mesh Refinement – Theory and Applications.* 2005. XIV, 552 pp. Softcover. 3-540-21147-0

Vol. 42 A. Schmidt, K.G. Siebert, *Design of Adaptive Finite Element Software.* The Finite Element Toolbox ALBERTA. 2005. XII, 322 pp. Hardcover. 3-540-22842-X

Vol. 43 M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations II.* 2005. XIII, 303 pp. Softcover. 3-540-23026-2

For further information on these books please have a look at our mathematics catalogue at the following URL: `www.springeronline.com/series/3527`

Texts in Computational Science and Engineering

Vol. 1 H. P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming. 2nd Edition 2003. XXVI, 855 pp. Hardcover. ISBN 3-540-43416-X

Vol. 2 A. Quarteroni, F. Saleri, *Scientific Computing with MATLAB.* 2003. IX, 257 pp. Hardcover. ISBN 3-540-44363-0

Vol. 3 H. P. Langtangen, *Python Scripting for Computational Science.* 2004. XXII, 724 pp. Hardcover. ISBN 3-540-43508-5

For further information on these books please have a look at our mathematics catalogue at the following URL: `www.springeronline.com/series/5151`