PIECEWISE - PARABOLIC METHODS FOR PARALLEL COMPUTATION
WITH APPLICATIONS TO UNSTABLE FLUID FLOW
IN 2 AND 3 DIMENSIONS

FINAL REPORT
DEPARTMENT OF ENERGY GRANT NO. DE-FG02-87ER25035

Principal Investigator, Paul R. Woodward

University of Minnesota

March 26, 2003

DOE Patent Clearance Granted

MP Dvorscak

Mark P. Dvorscak
(630) 252-2393

E-mail: mark.dvorscak@ch.doe.gov
Office of Intellectual Property Law
DOE Chicago Operations Office

June 6 2003
Date

Introduction

This project covers a span of many years, beginning in early 1987. It has provided over that considerable period the core funding to my research activities in scientific computation at the University of Minnesota. It has supported numerical algorithm development, application of those algorithms to fundamental fluid dynamics problems in order to demonstrate their effectiveness, and the development of scientific visualization software and systems to extract scientific understanding from those applications. Over the years these activities have attracted complementary funding which has made possible expansion of this core effort under the aegis of a small lab, the Laboratory for Computational Science & Engineering (LCSE), which I direct. The LCSE has during the last year moved into the University of Minnesota's new Digital Technology Center in the renovated Walter Library building at the center of the campus (see figure). This project has support from the University of Minnesota in the form of prime space in this beautiful renovated building, including roughly 4500 sq. ft. of lab space on the ground floor, plus offices on the 4th floor within the Digital Technology Center. The Minnesota Supercomputing Institute, of which I am a Fellow, is on the 5th floor of this same building and its primary computing equipment is in the basement. The LCSE's space on the ground floor includes a 2500 sq. ft. PowerWall visualization room (see figure). The PowerWall visualization system represents a technology that grew more or less directly out of work supported by this project at its outset on scientific visualization using high resolution movie generation and display. The PowerWall technology has been broadly adopted at the DoE labs, including especially Livermore, Los Alamos, Sandia, and Argonne. Our own PowerWall, shown in the figure, was installed two years running in DoE exhibit booths at the Supercomputing conferences. Most recently,

*New LCSE location in the Digital Technology Center (left),
with a special PowerWall visualization room (below).*

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

work on PowerWall visualization has been supported primarily through the DoE ASCI and NSF Supercomputer Centers (PACI) programs.

In the following sections, we will very briefly survey results of this project up to the last 3-year grant period, and then spend more time on the most recent results from that last continuation, with its two no-cost extensions, identifying current continuing directions for this research.

Improvements in the basic single-fluid PPM compressible gas dynamics algorithm

When this project began, in the 1980s, our Piecewise-Parabolic Method (PPM) had just been published (see reference list) along with tests of its effectiveness on 1- and 2-D flow problems with strong shocks. This grant funded my team's initial work at the University of Minnesota in which we exploited very large amounts of computer time on the Minnesota Supercomputer Institute's Cray-2 to demonstrate the high resolving power of PPM for complex 2-D compressible flow problems. Those demonstrations led to many improvements in the PPM algorithm, continually increasing its resolving power and decreasing the incidence of known numerical artifacts in the solutions. The most recent versions of PPM, incorporating all these improvements, are made available through a library of code modules and associated documentation, which we call PPMLIB and which was supported through Dan Hitchcock's program in the Office of Science. To appreciate these improvements, see the results in the articles of the reference list, many of which can be found on the LCSE Web site at www.lcse.umn.edu. Principal improvements to the original PPM algorithm for single fluid problems over this period involve the elimination of the original algorithm's grid jiggling mechanism in favor of a very smart numerical viscosity, the interpolation of Riemann invariants and fundamental fluid dynamic variables in a consistent manner, the elimination of false sound waves generated at moving contact discontinuities in the original algorithm, and a test for smoothness of the local flow that is used to decide whether or not to apply monotonicity constraints to the interpolations (with these monotonicity constraints kicking in in a smooth and continuous manner). In a recent comparison of many different fluid dynamics algorithms carried out by Burton Wendroff at Los Alamos, it is at least my own opinion that PPM came out on top. Through this process of continual improvement, supported by this grant, PPM has therefore maintained its status as a state-of-the-art numerical method for compressible flow simulation.

Before discussing the various extensions of the PPM algorithm to multifluid hydrodynamics, MHD, and implicit implementations, I should first point out that PPM remains at the state of the art not only because of its accuracy and computational efficiency but also because of the enormous amount of work that our team has performed over the years, with support from this grant, in modifying the way that this algorithm is implemented on parallel computing systems. It is frequently noted that roughly equal gains have historically been achieved by improving algorithms as by improving computing systems. Although one can argue this point, it is clear that one must participate in both improvements in order to remain at the state of the art. No method that ignores the development of the large parallel computers found at the major DoE labs can remain useful for large-scale scientific simulation. At the outset of this project, PPM had already been completely vectorized and parallelized to use all 4 CPUs of a Cray machine of the day. When the Cray-2 was introduced in 1985, PPM achieved over 100 Mflop/s on each CPU despite this machine's use of slow DRAM memory. This unusually high performance (roughly one useful flop per CPU clock cycle) was the result of the formulation of the PPM algorithm in 1-D passes, each implemented in a series of vectorizable loops using 1-D local arrays. For roughly the same reason, PPM in its latest form achieves 961 Mflop/s in 32-bit mode or 775 Mflop/s in 64-bit mode on a 2.53 GHz Intel Pentium-4 CPU installed in a \$1000 desktop PC. Although this present performance on the \$1000 machine is roughly double that on the University of Minnesota's \$17,000,000 Cray-2 of 1985, the trick to getting real performance today is finding an effective way to utilize thousands of such microprocessors simultaneously and efficiently. This is a formidable task. Our team at Minnesota has put enormous effort over many years, with support from this project, to achieve this goal. We have met with much success.

Parallelizing PPM

This project has supported a wide variety of pioneering experiments with PPM in the art of parallel computation. In the late 80s, Cray Research made available to us 3 multiprocessor Crays interconnected by HiPPI links so that we could demonstrate computation of a single problem on a cluster of such machines. Unfortunately, only a single hour of such triply dedicated time was made available, and our code got working on this unusual cluster only in the final minutes. Later, we were able to get two 4-processor Crays at the Minnesota Supercomputer Center running on a single problem at a parallel efficiency of 88%. This turned out not to be a very practical demonstration; although, apparently, there was considerable interest in this experiment among Cray's customers in the oil industry. For us it was a first experience with what is now commonly called an SMP cluster, the most popular supercomputer architecture at the present moment. To do this demonstration, we decomposed our problem domain into grid bricks and sent information about halos of ghost cells back and forth over the HiPPI connections between the machines. This strategy resulted in a minimum of data communication between the machines, but it demanded good load balancing, which was only possible on dedicated equipment.

In the early 90s, first Los Alamos and then the University of Minnesota purchased Connection Machines. Working with Karl-Heinz Winkler's group at Los Alamos to implement PPM on the LANL Connection Machine, I determined that the proper way to do this was to execute the equivalent of vector instructions on each CPU of the machine. This insight led to intense discussions with the Thinking Machines representatives. Ultimately it led to a new compiler for the machine, the so-called "slice-wise" compiler. For PPM, this gave an overall factor of 3 increase in performance, which, I was told, was achieved by many other applications as well. The result was that for PPM the price/performance ratio for this machine was roughly 10 times better than for a Cray of the day. Both Los Alamos and the University of Minnesota obtained new Connection Machine CM-5s almost simultaneously, although the machine at LANL was twice as large. Our team continued to work with the compiler team at Thinking Machines, and we were instrumental in getting many further improvements into the compiler shortly before the company went bankrupt. On our university's 512-node (2048 CPU) CM-5, we were able to achieve 11 Gflop/s performance with PPM, which was about as well as anyone was able to do on that machine for a real application. The SIMD computing techniques we developed proved useful for several more years on the Cray T3D and T3E machines, on which our codes were also very good performers (we participated in a NASA program for which our project payments hinged on achieved T3E performance).

When the CM-5 was described to me, I thought it would be an SMP cluster, with each SMP consisting of 4 of its CPUs located at a single node of its communication network. Although the design of the machine seemed to indicate an SMP cluster, the design of the compiler made use of the machine as such a cluster impossible. In the fall of 1993, we collaborated with Silicon Graphics to test a true SMP cluster architecture. SGI had just come out with its 36-processor "Challenge Servers" (the name was influenced by our collaboration, the goal of which was to do true "Grand Challenge" computation on a cluster of such workstations). In its manufacturing building in Mountain View, SGI built for us a cluster of sixteen 20-processor workstations (Challenge servers), which SGI later labeled the "Challenge Array," since they thought that workstations did not sound sufficiently serious and a cluster did not, apparently, have the dignity of an array. During the previous year or two we had collaborated with Karl-Heinz Winkler's group at Los Alamos both to implement PPM on true workstation clusters (1 CPU per machine) and also to think about more promising parallel architectures which IBM might build. Our experience in that collaboration was that clusters of single-CPU machines achieved at best the performance of a single Cray CPU, with very much more programming trouble. We also found that getting a vendor like IBM to build a cool new machine was far more difficult than we had thought. My team's work with SGI in the summer and fall of 1993 therefore was based on tying together whatever the vendor had seen fit to build, rather than trying to get them to build something else. Because this vendor had chosen to integrate many CPUs with a shared memory in a single box, we were able to get way past the performance of a single Cray CPU. A Cray CPU at that time offered roughly 450 Mflop/s perform-

ance for PPM, while we were able to achieve 5 Gflop/s on our 16-node SGI SMP cluster. We decomposed our problem domain into equal-sized grid bricks augmented by halos of ghost cells. We used thread-based shared memory programming to update each grid brick with the 20 CPUs, and we then performed synchronous communications over 20 FDDI network links in a 3-D torus arrangement. Over several days, we used this experimental machine, which was subsequently dismantled and sold in parts to SGI customers, to perform the first simulation of homogeneous, compressible turbulence with PPM on a grid of 1024^3 cells. This record-breaking size of the computation was made possible by the use of inexpensive DRAM memory in the air-cooled SGI machines. The 28 GB of aggregate memory on this system, with a list price of merely 11 M\$, was the key enabling feature for this unprecedented PPM turbulence calculation. We had been promised a month of dedicated time on this SMP cluster, but that month turned out to include the building of the machine and the debugging of our application. With only 4 solid days of computation, we got this run started, but were unable to access any machine that could complete it until Los Alamos purchased an SMP cluster from SGI as part of the ASCI program, and we were able to run on early hardware in 1997, in collaboration with Winkler's group. In 1995, partially in recognition of all this work on parallel computation, the IEEE presented me with the Sidney Fernbach Award. Also, as a result of this work the Army Research Lab purchased from SGI an 8-node Power Challenge Array and bought a ninth node to install in our lab.

The SMP cluster experiment with SGI was cited as a proof of concept by the ASCI program in its white paper laying out the basis for its "blue" platform procurement. Our team, with support from this project, produced a test code, a simplified version of our SMP cluster PPM code that we called sPPM. Designated performance levels for this sPPM benchmark code were "mandatory requirements" of the original ASCI blue platform procurements. Working with LLNL and IBM, we were able to achieve 1.03 Tflop/s sustained performance on the LLNL machine, so long as IBM (legally) counted the flops the right way. After SGI built an aggressive form of loop fusion into its compiler at our suggestion, sPPM actually ran in about 10% less time on the LANL machine than on the LLNL machine. However it was not possible, according to the contract language, to legally count the flops on that system so favorably. With sPPM on the Livermore machine, we performed in December, 1998, a simulation of the Richtmyer-Meshkov instability on a grid of 8 billion cells. For this simulation, we and our collaborators at LLNL and IBM were presented the 1999 Gordon Bell Award in the performance category. In this simulation, we went beyond the sPPM benchmark implementation to thoroughly overlap communication and computation, but the strategy still demanded perfect load balancing and dedicated equipment. The considerable scientific insight that came out of this calculation will be discussed in a later section.

During the final grant period of this project, and with support also from Dan Hitchcock's program in the Office of Science, we have developed a more flexible and powerful method of SMP cluster computing that we call SHMON, for SHared Memory On Network. The initial impetus for this work was to find an effective means for dynamic load balancing over a cluster. Our threads-based shared memory programming techniques for an SMP at a network node already exploited self-scheduled loops of parallel tasks, so that dynamic load balancing was automatic and highly effective (see our article from the IEEE High Performance Distributed Computing conference in 1998 on the reference list). In SHMON, we extended to the cluster network this shared memory coding style using self-scheduled task loops. This model requires that any SMP on the network be able to update any grid brick, not just grid bricks whose data reside in that node's memory. The key to this development, set out in a 1996 article in *IEEE Computer* magazine (see reference list), was to realize that for dynamic load balancing to be effective, there must be many more grid brick subdomains to update than there are network nodes. This requirement generally forces one to set the size of the grid brick that constitutes a single task for an SMP to the smallest size that can be reasonably efficient. For such small grid bricks, the surface to volume ratio is so large that we require only about 3 times the bandwidth to read from across the network an entire grid brick rather than just the ghost cell regions. Because SMP cluster networks on all major systems have always offered far more than this bandwidth, it is practical to implement a shared memory computing model over the cluster.

To see that typical cluster networks offer sufficient bandwidth for our SHMON parallel computing model, it is most useful to consider a typical example. Consider a computation of a fluid flow with our PPM code on a grid of 2048^3 cells (we plan to carry out such a run over this coming summer). If we update all the cells of a grid brick subdomain for a full time step in a single SMP task, then the smallest efficient grid brick turns out to have roughly ten times the number of grid cells on a side as the number of ghost cells needed along each face of the grid brick. For our present PPM algorithm, this number of ghost cells is 10. Thus we tend to use grid bricks of 128^3 cells, which demand that 18% of the computational labor be performed in the ghost cell regions. Since this ghost cell work is redundant with work done for neighboring grid bricks, it constitutes an overhead for the parallel implementation. Happily, this turns out to be nearly the only parallel overhead, and performance scales linearly with aggregate problem size and number of network nodes. Now for our example problem, we will have 4096 grid bricks of 128^3 cells each. Whether or not we choose to use 64-bit arithmetic, we always store the grid brick data as 32-bit floating point numbers. With 5 fluid variables per grid cell, we then have 20 bytes per grid cell to store. Therefore the grid brick interior constitutes a “file” (we will call it a file whether we store it in memory or on disk) of 40 MB. This same grid brick, with all its adjacent ghost cell regions becomes a brick of 148^3 cells, which is larger by a factor of 1.55, making it 62 MB. Running on the same Intel Pentium-4 microprocessor mentioned earlier, and performing 64-bit arithmetic (for which this CPU gives about the same performance on PPM as the very latest and much more expensive Intel 1 GHz Itanium-2), we can update the grid brick (at 984 flops per cell per 1-D pass) in roughly 3.27 sec. With two of these CPUs at each node (a typical configuration today), this grid brick update would take 1.64 sec. During this time, we would need to read and write back 62 MB if we were to completely overlap computation and communication. Today, a cluster of such dual processor Pentium machines would typically be interconnected via Myrinet (as is the 1024-node cluster at Los Alamos), and this interconnect delivers to our application 150 MB/sec sustained bandwidth. At this rate, we can bring an entire grid brick, with its ghost cells, and write one back across the network in 0.83 sec. This means that the network offers twice the bandwidth needed to allow any SMP to update any grid brick. Even if the interconnect is implemented with Gigabit Ethernet, which delivers to our application 90 MB/sec sustained bandwidth, we could still bring the grid brick data across the network without slowing down the computation at all.

The reason we have chosen to implement in SHMON a shared memory programming style for the entire SMP cluster is that it delivers tremendous benefits. We launch the 4096 grid brick update tasks in a specified order on each time step, although it is not necessary that these tasks be completed in this same order. Each grid brick data context consists of a grid brick interior, 6 ghost cell faces, 12 edges, and 8 corners. We can make sure that all 27 of these files are located at the same network node. We can then lay out the data contexts across the network nodes in such a way that the task launch order proceeds from one node to the next, so that there will not be contention on the network. A participating network node consults a global database (now implemented using MySQL) in order to select its next grid brick update task. It fetches the data context for this next task while it is updating its previous grid brick. Once the data context has been fetched, the SMP can update it without the need for any additional network communication. Upon completion of the update, and during the computation of the next update, it writes new data back across the network to files in 27 different task data contexts for the next time step’s round of updates. Once this data has been written, the node accesses the global database to record this fact and to select its next task.

It should be clear from the above description that, in our example with 4096 grid bricks, each node should always be able to find a task to perform, so that it is never idle, even if we have as many as 1024 nodes participating. There is no requirement that the grid brick updates all take the same amount of time. Load balancing over the entire cluster is automatic and effortless. For the same reason that the grid brick updates need not represent equal amounts of work, we need not require that the SMPs at the network nodes be equally powerful. Nor do we require that the nodes be dedicated to our use. If a node is shared with other users, it will simply take longer to update its grid brick, but this is of no concern, so long as we are paying only for the fraction of the node that we receive. If a node takes too long to do its work, we

can kill its process and give up on it, so long as we have been careful to use the excess network bandwidth that most SMP clusters offer today to place duplicate copies of the data contexts at alternate network locations. In fact, if a node fails to respond for any reason, we can eliminate it from the active group and proceed with the computation, immediately reassigning its work to a more responsive node. Thus, SHMON computation can be made fault tolerant. This is a considerable boon on very large systems. Particularly when we store the task data contexts on disks (either at the network nodes or on special storage nodes) SHMON offers yet another large benefit. It allows us to start up a computation of any size so long as we can get access to only a very small number of nodes (perhaps even as small a number as just 1). As additional nodes become available, we can add them to the participating set, allowing them to select tasks from the global database and to jump right into the team without any fuss. This capability allows us to bring to modern parallel computation the benefits of time sharing, with perhaps the greatest benefits delivered to runs aimed at identifying bugs that occur only at scale.

Our first demonstration of SHMON computation was performed at NCSA, since that center allowed us to interconnect two of their 128-processor SGI Origin-2000 machines using 4 Fibre Channel Arbitrated Loops, on which we placed a large number of striped, dual ported Fibre Channel disks. The only other connection between these machines was a Fast Ethernet link. We placed a raw file system on these disks, which we owned, and used them to hold all the data contexts for a 1024^3 PPM simulation of turbulent convection in a red giant star. We found that only 150 MB/sec sustained bandwidth was needed to support our SHMON computation with PPM on this system, so that our 4 Fibre Channels turned out to be overkill. We used this set up for a two-week demonstration period to run our large PPM simulation as a background job, relinquishing CPUs whenever anyone else desired them and grabbing CPUs whenever they became available, with a time granularity of 10 seconds. We demonstrated this capability at Supercomputing 98 in the NCSA exhibit booth. Unfortunately, it then required manual requests to be sent to the various job queues, and NCSA offered to redesign its set of Origin-2000s only if we could help several other users to compute in this mode (an effort that is now finally underway, with the NSF TeraGrid as the target computing platform).

We performed a more recent demonstration of SHMON computation using NCSA's Itanium cluster, which NCSA agreed to augment for SHMON with striped dual SCSI disks at each 2-processor network node and with 8 special storage nodes dedicated to our use, offering 800 MB/sec sustained aggregate bandwidth and about a terabyte of scratch storage managed solely by our team. Placing the task data contexts on these storage nodes has allowed us to demonstrate completely resilient fault-tolerant computing on this cluster. Through a recent donation to our LCSE lab from Unisys of a 32-processor Itanium-2 machine with 64 GB memory which we will interconnect on a Gigabit Ethernet fabric to 66 Dell PCs in a student lab on the same floor as our machine room, we now have a local development system for SHMON computing (each PC will have 400 GB of 100 MB/sec disk storage dedicated to our use and accessible to us over the network whether or not a student is using the machine from its console).

The concept of Grid computing arose near the end of this DoE project, but our work on SHMON parallel computation was one of the fluid dynamics applications that initially seemed promising for Grid architectures. In a sense, a Grid is just a network of SMP clusters. Just as the data transport bandwidth per delivered Gflop/s steps down sharply when one goes out onto a cluster network from within an SMP, this bandwidth steps down yet another notch in going out onto the Grid from within a cluster. The Grid then merely introduces one more step in the memory hierarchy of latency and bandwidth (latency is rarely a limiting consideration for SHMON). Although we considered experimentation with SHMON Grid computation while this project funding was still in force, it was not until more recently that we have been able to locate systems with the proper system software infrastructure and Grid network bandwidth. Using this SHMON approach that grew out of our work on this project, we intend over the next year to perform Grid computing experiments. A recent NSF CISE Research Resources grant is enabling us to outfit a cluster of PCs in a student lab in our building, and this cluster will be interconnected with the 32-CPU Unisys machine that has recently been donated to the LCSE. Together these two systems represent for us

a Grid computing testbed where we have sufficient control over the system and network configurations as well as the usage policies to make effective Grid computing tests. We are also planning to test this approach on NSF's new TeraGrid (PPM has been selected by the NSF as one of just 2 early test applications for the TeraGrid). Although neither of these test systems is at a DoE lab, the labs do offer similar systems in the form of the large ASCI computing platforms, each of which now looks like a small cluster of SMP clusters. It is simply more difficult to obtain access to these systems, because of security concerns, and it can prove impossible to publicly show results computed on them even if access is granted. Nevertheless, I believe that the SHMON paradigm will prove very useful on DoE systems. It is worth noting that, partly through interaction of our LCSE team with the DoE labs through an ASCI contract now completed, the new large computing systems there offer special storage nodes with shared file systems. These storage nodes are just what one needs for fault tolerant and flexible SHMON computation.

Extending PPM to MHD

With support from this project, Wenlong Dai and I extended our PPM gas dynamics algorithm to MHD. This work was accomplished mostly before the last grant period of the project, so I will only briefly mention it here. Several other teams were working on this problem, including those of Phil Roe at Michigan and Phil Colella at Berkeley, but our paper on this subject was submitted for publication first. I also believe that we made special contributions to this research area by discovering a method by which the cell-centered Godunov-type formulation of our schemes for hyperbolic conservation laws could be adapted so that, in Cartesian coordinates, the constraint that the divergence of the magnetic field vanishes could be maintained exactly (see article on reference list). This same technique, I am assured by Phil Colella, also works for handling material strength. In addition to this discovery, we also sorted out some peculiarities of 2-D MHD and were able to deal with them much more simply by realizing that they represented very special, degenerate cases of more normal 3-D phenomena.

Our extension of PPM from gas dynamics to MHD was an important learning experience. It forced us to realize just how special gas dynamics is as a representative of a hyperbolic system of conservation laws. In particular, the ability to relatively easily solve nonlinear Riemann problems (although it had never seemed easy before) is a very special property of gas dynamics. Even though we developed a Riemann solver that very closely matched for MHD the properties of our standard gas dynamics Riemann solver, we quickly found that instead of this computation taking about 10% or 15% of the overall time, in MHD it took 95% of the time. This fact forced us to simplify the Riemann solver along lines that I had already experimented with in our gas dynamics code. It also forced us to seriously consider what, precisely, the Riemann solver is good for in such a numerical method. In gas dynamics, because accurate nonlinear solutions to Riemann's problem are easy to achieve, it is tempting to think that they are more useful than they actually turn out to be. The nonlinearity of the Riemann solution acts like a smart viscosity in helping to generate entropy (as well as the other sound wave Riemann invariant whose characteristics cross the shock) in the proper amount in the numerical representation a gas dynamic shock. Peter Lax's early work proved that this function can be replaced by any other convenient form of numerical viscosity, although I believe it is important to make the replacement numerical viscosity "smart" in that it turns on only for strong shocks, and then only when it is really needed. The key to when it is needed was already laid out in my 1984 articles with Phil Colella, in our discussion of grid jiggling and "Cray instabilities." In replacing grid jiggling in PPM with a smart viscosity in the mid 80s, I had already come up with a formulation for a variable numerical diffusion coefficient which turned out to generalize easily to the MHD case and to work without a hitch (see reference list). This work taught us that the one aspect of a Riemann solver that is critical to numerical computation is its selection of the upstream Riemann invariant data for each characteristic field. It turns out to be this feature, rather than the nonlinearity, that yields such sharp shock structures. No matter how carefully one solves the Riemann problem, we realized, it is never the "right" Riemann problem unless one is using shock fitting in one

dimension. Therefore one always must introduce a smart viscosity in any event. With the simplified Riemann solver we developed for MHD, we just had to make that smart viscosity a tiny bit larger. As a result, our work extending PPM to MHD taught us what we really need from a Riemann solver for any hyperbolic system and what we can do without. This lesson has been helpful in other contexts (see below).

Extending PPM to handle slow flows implicitly

Wenlong Dai and I worked over the course of several years on this project on the problem of building implicit versions of numerical algorithms like PPM. We started with my work with Fryxell, Winkler, and Colella, published in 1986 (see reference list), which had been limited to 1-D Lagrangian calculations. Most of the advances of importance in this area occurred toward the end of this project, and therefore I will discuss them more thoroughly here. There are many difficulties in formulating an effective implicit version of a numerical method as complex as PPM. One of these difficulties, encountered already in the earliest work, was finding the Jacobian derivative of the nonlinear Riemann solver. As already discussed earlier for MHD, Wenlong Dai and I had discovered that we did not need so complex a Riemann solver, and this insight is especially valid for flows that can be treated implicitly. However, even with the Jacobian of the Riemann solver in hand, many difficulties remain. In our 1-D Lagrangian implicit PPM, we had given up on the possibility of developing an iterative approach to the equations that result, and we had used a direct solution technique for a linearized system, with an iteration to capture the nonlinearity after the fact, as it were. This seemed not only inefficient, but very unwieldy when applied in multiple dimensions. Also, our work parallelizing PPM had shown that the formulation of the scheme as a sequence of 1-D passes was critically important to its efficient implementation on microprocessor CPUs, and especially on SMPs built with such CPUs. These problems provided stimulation for this project over a course of several years, with most of the work concentrated in the last grant period. By the end of the project, Wenlong Dai and I had come up with solutions, we believe, for all of these problems. It remains to implement and test our resulting approach and to quantitatively assess its efficiency on modern parallel computing platforms, namely SMP clusters. Renewed support from this program could provide the necessary resources to undertake that work.

The solutions to the problems listed above are laid out in a series of articles by Dai and Woodward on the reference list. Tentative solutions for multidimensional iterative schemes that are consistent with the SHMON parallel programming style have not yet been published, since they must first be tested more thoroughly and made more computationally efficient. Rather than review these developments in a time ordering, it is better to instead briefly lay out the design of an implicit PPM hydrodynamics scheme that they imply. These ideas have been applied (see reference list), at least in 1-D, to MHD and radiative transfer as well as to hydrodynamics.

Design of the 1-D Pass:

The 1-D pass of the method for the hydrodynamic case provides the simplest context in which to see some of the special properties of our implicit-explicit approach. To date, we have multidimensional results only for hydrodynamics.

Consider the 1-D coupled set of hyperbolic conservation laws $\partial U / \partial t + \partial F / \partial x + S = 0$, where U is a column matrix of conserved quantities, F is a column matrix of fluxes of those quantities, and S is a column matrix of source terms. For 1-D inviscid gas dynamics in Eulerian coordinates, the elements of the matrix U are of course: $\rho, \rho u_x, \rho E$. Here ρ is the gas density, u_x is the x-velocity, and $E = \epsilon + \frac{1}{2} u_x^2$ is the total energy per unit mass. The equation of state relates the pressure, p , to the density and the internal energy per unit mass, ϵ , and relates the sound speed c_s to both of these quantities. For inviscid flow in 1-D, S might represent a gravitational body force. For multi-D flow, we will use S to represent the effects of fluxes of U in transverse grid dimensions. In this case U will also include transverse momenta. In the diagram above we show a space-time diagram for 3 cells of our 1-D grid.

In this diagram we show 3 cells along the horizontal axis and a single time step along the vertical axis. We denote time levels by the superscripts H and N for time levels half way through the time step and at the new time level, respectively. The values of the conserved quantities, U , represent averages over the cell volumes, even though the symbols are placed at the cell centers in the diagram. The fluxes, F , of the conserved quantities are computed at time levels H and N at the left- and right-hand cell interfaces. No fluxes are computed at the original, or “old,” time level, a feature that allows the scheme to produce an approximation to the correct long-time solution in the limit of infinite Courant number, as is discussed in our first (1986) paper on this sort of method – Fryxell, Woodward, Colella, and Winkler – given in the reference list. By approximating the fluxes as functions of time at each cell interface using a straight line joining the values computed at time levels H and N , this scheme achieves second-order formal time accuracy while maintaining the robustness of the backward Euler time differencing scheme (which also employs only new and no old flux values). This linear time dependence of the approximated fluxes is used to compute the time averaged flux for the first half time step, which in turn is used to determine the values of the conserved quantities, U , averaged over the cells at time level H . Clearly, in this approximation the time-averaged fluxes needed to obtain U at time level N are just the fluxes at time level H . This fact allows the implicit scheme to be joined continuously onto the explicit PPM scheme, in which only an approximation to the fluxes at time level H is computed and in which no values for U at time level H are needed. When a characteristic speed is so slow that it can be treated explicitly, then we simply do so according to the PPM scheme, and there is no glitch in passing back and forth from explicit to implicit treatments.

It was shown in Fryxell et al. that our introduction of the time level H in this scheme makes it well behaved in the limit of infinite Courant number while preserving its second-order accuracy in time. This does, however, increase the computational labor associated with a time step update. We take the view that this labor is justified by the accuracy and robustness of the scheme. We also note that on SMP clusters computationally intensive methods tend to extract significantly greater fractions of the available CPU power than simpler schemes. Fryxell et al. showed that for linear advection the extra labor associated with time level H allowed the same accuracy to be obtained with a roughly 8 times larger time step, so that in that simple case there is no issue in terms of efficiency.

Flux Computations:

It is important to point out that the flux values noted on the space-time diagram are computed in a special way that may not strike some readers as intuitive. The dark diagonal lines in the diagram represent characteristic paths. First consider the computation of the fluxes at the half time step, H . The left- and

rightward moving sound wave characteristics as well as the streamline characteristic, shown for the case of a positive fluid velocity, meet at the right-hand cell interface at time level H in the diagram. The fluxes $F_{i+1/2}^H$ are computed by solving an approximate Riemann problem using the Riemann invariant data transported along these characteristics. In the case of fast sound waves (or of slow flow) that is shown, both sound wave characteristics from time level N would extend more than all the way across the grid cells i and $i+1$ if we were to continue them on down until they met the x -axis. This means that we cannot use our PPM interpolation of the subgrid structure of the fluid state within cell i or cell $i+1$ to find the Riemann invariant information carried along these characteristics. Instead, we use an interpolation in time for the average fluid state in each cell to obtain this information.

The implicit scheme defines three time levels – the original time, the new time (level N), and the time half way between (level H) – at which the cell averages of conserved quantities (U) are either specified or solved for. The values of U at these 3 time levels are used to define interpolation parabolae that provide estimates for U at any time level within the time step. The sound wave characteristics are then traced from the cell interface at either time level H or N to the middles of the adjacent cells, where their ending points determine times at which the fluid state, U , is interpolated. An approximate Riemann solver is then used to combine the rightward propagating Riemann invariant information, R_+ , from this fluid state in cell i with the leftward propagating Riemann invariant information, R_- , from cell $i+1$ to produce, in the case of hydrodynamics, the pressure and velocity at the interface at either time level H or N . The spatial domains from which these Riemann invariant signals emanate are indicated by the brackets drawn on the diagram. These domains extend across the width of the grid cells in question; they are not points in space-time but spatial regions. This means that the pressures and velocities obtained at the cell interface are actually best understood as estimates of temporal averages there, as explained in Fryxell et al. for the simpler case of linear advection. This fact has a special consequence. As the time step is reduced toward a Courant number of unity, this temporal average for the interface flux at level H approaches an average over the entire time step. As the Courant number falls below unity, it is therefore natural to form the spatial average for the transported Riemann invariant data from the spatial domain of dependence of the cell interface over the time step. This is precisely the method used to obtain interface fluxes in the explicit PPM scheme. Therefore the implicit and explicit schemes are entirely compatible, and they are joined continuously onto each other here to form a method that treats each characteristic individually as either implicit or explicit as the value selected for the time step demands.

We may also observe that our special construction allows for a smooth transition, in going from implicit to explicit treatments, from a simple second-order accuracy in space and time to a treatment like that of explicit PPM in which higher-order features are incorporated into the spatial interpolations. This point is not a detail. The special features of the PPM spatial interpolation are designed to recognize and accommodate sharp fronts. Here we assume that if the propagation of any such fronts is of importance, the time step will be adjusted so that no such front moves more than one grid cell width per time step, making an explicit treatment possible. In the example discussed below, the propagation of a contact discontinuity is treated via explicit PPM spatial interpolation of the entropy, while sound wave Riemann invariants involving pressure and velocity disturbances are treated implicitly. This feature of our method saves a potentially enormous amount of labor in the implicit portion of our calculations. It also makes possible incorporation of moment conserving explicit advection at the fluid velocity (discussed in a later section) in which subgrid structures are specified and updated by the advection scheme rather than interpolated.

From our approximate Riemann solver at the time level H , we obtain an estimate of the fluid pressure and velocity in a time interval centered on H at the cell interface. We call this velocity $\bar{u}_{i+1/2}^H$, and we have used an overbar to emphasize that this should be understood to be a temporal average. We use this velocity as the slope of the streamline characteristics from the interface at both levels H and N . This allows us to trace these streamline characteristics, as shown in the diagram, and to obtain the spatial domain indicated by the bracket along the x -axis in the diagram, which contains the entropy information,

R_0 , that propagates to the cell interface during the time step. We use the average fluid state over this domain to extract this entropy information, denoted by $R_{0,i+1/2}^H$ in the diagram, and use it to determine the time-averaged density, $\bar{\rho}_{i+1/2}^H$, centered on time level H at the right-hand cell interface. In a 1-D pass of a multidimensional calculation, we obtain, along with the average entropy in the bracketed domain, average values of the transverse velocities, which we assign to the time averaged values at the right-hand cell interface centered on the time level H . We use these to construct fluxes of the transverse momenta. We also solve an approximate Riemann problem at time level N at each cell interface. As with the Riemann problem at level H just described, this gives us estimates of the interface fluxes of conserved quantities, U . By performing a linear interpolation between these interface fluxes, denoted on the diagram as $F_{i+1/2}^H$ and $F_{i+1/2}^N$, we estimate a linear time dependence of the fluxes over the entire time step. Along with any source terms, the difference of these fluxes at the two cell interfaces gives us the time rate of change, which we approximate as a linear function, of the conserved quantities U over the time step. Integrating this time derivative, itself linear in time, gives us a parabola for U as a function of time. From this parabola we extract the values of U at time levels H and N .

Implicit Iteration:

Finally, the implicit aspect of our scheme is fully apparent. An initial guess for U_i^H and U_i^N leads us, through a variety of interpolations and Riemann solvers, to a set of new values for these quantities. Given U_i and fixed values for U at all three time levels in both adjacent grid cells, one can formulate a nonlinear solver for U_i^H and U_i^N based upon all of these relationships. Based upon this solver for a single cell, we have built an iterative solver for the U values at levels H and N across an entire strip of our mesh. Our characteristic formulation provides for us an exceptional opportunity to achieve global information propagation upon every pair of revisions of all the cells. This is a very important feature of our method. It derives from our formulation of implicit problems using the structure that comes with hyperbolic problems rather than elliptic ones. If we begin at the left-hand end of a grid strip and revise grid cells one-by-one, working from left to right across the grid strip, rightward propagating Riemann invariant information is propagated globally. If we now pick up information from the right-hand boundary condition and work our way back from right to left, one cell revision at a time, leftward-propagating Riemann invariant information is globally communicated. Those characteristics which are explicit, of course, are handled in the first such sweep, and need no further revision. If the problem were linear, which of course it is not, we would have the answer in just two sweeps. In this case, left- and right-moving waves, like small ripples on a pond, would not interact, and there would be nothing more that we could do to improve our solution.

For a purely linear problem, our pair of sweeps along the grid strip gives a direct solution, and in this respect (alone) it resembles a tridiagonal linear system solver. However, in general our sweeps are not linear, since we solve a fully coupled nonlinear problem in revising each cell of the grid strip. If the boundary values in “ghost” or boundary cells are held fixed (a very important condition), just one or at most two more pairs of sweeps serves to converge on the full nonlinear solution. However, this solution will generally not satisfy the imposed boundary conditions if they depend upon the solution itself. One could revise the boundary values upon each sweep, but this approach leads to huge numbers of iterations before convergence is achieved. Instead, we choose to revise the boundary values with a Newton method. This brings convergence very rapidly, with only a couple of Newton iterations producing a satisfactory overall result in a time-dependent calculation (see examples in the references). To obtain the Jacobian matrix expressing the dependence of the solution values of U at levels H and N in the two grid cells, one at each end of the grid strip, upon the values of U at these same time levels in the two ghost cells, one just beyond each end of the grid strip, we must perform some significant computation. Essentially, we must perform a pair of sweeps across the grid strip, just as in the original method, in order to find each such derivative value. Here the 1-D nature of the pass is critical in keeping the computational labor under control by keeping the number of such derivatives to be obtained very small. For hydrodynamics, we need derivatives with respect to the pressure and the x -velocity only, even for 3-D problems. For

radiation hydrodynamics, we must also obtain derivatives with respect to the radiation pressure and the x -component of the radiation flux. The other variables in these cases are treated explicitly in the x -pass. We get these derivatives evaluated at the state, at our 3 time levels, that we have determined for the entire grid strip from our converged solution for the present estimates of the boundary values at levels H and N . In obtaining each derivative, we sweep across the grid and back solving for the derivatives of U^H and U^N in each cell with respect to the boundary cell value rather than for U itself. The equations we solve come from simply applying the chain rule all the way across the grid. Thus each derivative determination looks like the solution iteration process itself, and involves a comparable amount of labor.

Multiprocessing Issues:

One might argue that we obtain from the labor described above derivatives in interior cells that we proceed to completely ignore, but it is hard to see how one might eliminate this work. It serves to incorporate into the derivatives at the far end all the necessary detail about the internal structure of the grid strip. If we decompose the grid strip into several smaller sub-strips, then each such smaller strip can be processed in an identical fashion. Now the boundary conditions for interior strips depend not on the solution within those strips but instead on the solution in neighboring strips. If we consider a single macro-iteration to consist of obtaining a converged estimate of the solution and of the Jacobian derivatives which are based upon a particular set of boundary values, then the labor in such an iteration for a strip scales linearly with the number of cells in the strip. However, information propagates only across each small strip, not across the entire grid. Nevertheless, we have found (see reference list) that the Newton iteration converges equally rapidly if the set of all Jacobian derivative values for all the small strips is used to obtain estimates for new values for the boundaries of all the strips. Thus, global information propagation is achieved through this small linear problem that is solved for the new boundary values for all small strips. It is in fact not surprising that this Newton iteration for the set of all these boundary values converges as rapidly as the original Newton problem for the entire strip, since both are linearizations, even if the original one is about a potentially slightly closer approximation to the global state. This division of a grid strip into sub-strips for parallel processing is reminiscent of the multigrid approach, even though no coarsened grids are used. It is tempting to introduce multigrid acceleration of some sort here, although we have not seen as yet how it might further reduce the labor, since very few iterations are involved in our method as it stands. It is most natural to use some multigrid adaptation to accelerate the computation of the Jacobian derivatives. However, this would probably reduce their accuracy and thus result in a larger number of macro-iterations each involving somewhat less work. On SMP clusters, this is not obviously a useful direction for algorithm design; benefits will depend upon factors such as network bandwidth that are platform dependent. Therefore, at least for the moment, we have left multigrid concepts aside.

Multi-Dimensional Iteration:

It should be clear from the above discussion that if we could only build up accurate 2- and 3-D methods from just a few 1-D passes like that just described, we would have a very efficient method, with a number of interesting special properties, for implicit SMP cluster computing. It turns out that this cannot be done without introducing some level of multidimensionality into each 1-D pass. The reason for this is that the 1-D pass strongly damps short wavelength disturbances that it detects in its implicit characteristic fields. For 1-D problems, this is a desirable trait (it is related to the scheme's robust behavior in the limit of infinite Courant number). However, in 2- or 3-D problems our 1-D pass can mistakenly assign variations in U arising from resolved waves in the transverse dimensions to the fast characteristic fields of its simplified 1-D problem. This process does not affect robustness of the method; it affects accuracy. This sort of mistake can be avoided by including in each 1-D pass source terms that represent the fluxes of U in the transverse dimensions. These source terms are not computed or modified in the 1-D pass; they are obtained from the application of 1-D passes in the transverse dimensions. They serve to eliminate the effects of the wave misidentification problem just mentioned, as our experiments with implicit hydrodynamics demonstrate.

To see how these source terms address the problem mentioned above, consider the example of a stagnation flow involving inward flow in the x -direction and outward flow in the y -direction. Projection of this velocity field in an x -pass will detect (false) sound wave Riemann invariant disturbances, even if initially all pressures are equal. These false sound waves, if short in wavelength as measured in grid cells, will be strongly damped when the Courant number is large. This will tend, incorrectly, to impede the stagnation flow in this region. In our tests, this process acts incorrectly to damp surface instability growth, which naturally tends to appear first at short wavelengths. The introduction of source terms representing transverse fluxes directly addresses this problem. In the case just described, those source terms create a sink of material in the cell, which maintains the x -velocity distribution that feeds it. In addition, the transverse expansion, described through these source terms, tends to prevent unphysical increases of the pressure in the cell from these horizontal motions in the x -pass.

We have experimented at the end of this project with implicit problems in hydrodynamics. We have found that a sequence of passes symmetrized for 3 successive steps as $xyzxyzx$, then $yzxyzxy$, and then $zxyzxyz$, or perhaps a 50% more costly sequence like $xyzxyzxyzx$, then $yzxyzxyzxy$, and then $zxyzxyzxyz$, serves to produce results that compare very well with much more expensive runs carried out with our explicit PPM code. In these sequences, we have found that convergence is more rapid if we use results, such as the transverse fluxes, from the previous time step. It may be even more beneficial, although this leads to larger memory requirements for the code, to use the H and N time level values from the previous step to guess a linear time dependence of both the U and F quantities to start off the next time step's iterations. For accuracy, there is, as with implicit methods for ordinary differential equations, a trade-off between number of iterations and time step size. We have not yet had time to sort this out, and it may prove to be application dependent. So far, we have performed our tests using the problem, in 2- and 3-D, of a buoyant cylinder or sphere of gas in an isothermal, initially hydrostatic atmosphere. We have so far made the initial sphere 25% hotter than its surroundings (appropriate to a forest fire, perhaps), so that the explicit computation we use for comparison is affordable. As this initial temperature contrast is reduced, the flow becomes slower, and the simulation becomes more expensive with the explicit code. With our present formulation, we beat the cost of the explicit PPM code at a Courant number of about 20, and we feel that we can improve on this performance by at least a factor of 2 by simplifying the computation of Jacobian derivatives for our Newton iteration. I note that the expensive interpolations of the explicit scheme are performed only once per time step on the very first iteration. Our tests, with Courant numbers around 16, as determined by our constraint that the fluid advection remain explicit, show very close agreement between the explicit and implicit results, even with the Rayleigh-Taylor and Kelvin-Helmholtz instabilities that develop at the surface of the warm bubble. This comparison is easiest to see in 2-D, where we have used finer meshes. We do not see any evidence of new issues for the algorithm design in 3-D. Our 3-D results, although coarse, compare well and are robust and stable.

In the images at the top of this page, explicit (PPM) and implicit (PPMimp) results for this test problem of the rising thermal are compared for a grid resolution of 512^2 cells. The left-hand half of each

image shows the PPM result, while the right-hand half shows that of PPMimp for this symmetrical problem. The uninteresting portions of these images near the bottom and top of the domain have been cropped to save space. These grayscale plots show the density variations, with the original ten-fold increase in density from top to bottom in the initially isothermal atmosphere subtracted out, and with white representing the highest relative density. Boundary conditions at the left and right are periodic, and at the top and bottom are hard walls. These boundary conditions confine long wavelength sound waves in the problem domain. These sound waves can be seen as the large-scale patterns of light and dark regions in these images. At the left, the flow is shown after the warm bubble has mostly risen to its equilibrium elevation (the level in this atmosphere where its entropy is the same as that of the surrounding gas). Initial Rayleigh-Taylor fingers, dependent of course upon the grid resolution in these inviscid Euler simulations, have formed in nearly an identical fashion with both codes, despite the high Courant number of the implicit calculation. Of course, the Rayleigh-Taylor instability exists in incompressible flow, so this is not really a surprise. The largest differences between implicit and explicit simulations are visible in the detailed development of the Kelvin-Helmholtz instabilities, although these occur in the same places, as physically they must. Since these are unstable flows, it is unreasonable to expect detailed correspondence between them, except in the location and intensity of unstable regions and in the statistical properties, for fine grids, of the fluid mixing that the instabilities cause. The comparison at a later time is shown in the image at the right.

Enhancing the accuracy of PPM

During the final grant period of this project, we made a concerted effort to increase the accuracy of PPM. This was a challenging task, since for flows with strong shocks PPM is arguably the most accurate explicit hydrodynamics scheme presently in use. The idea was to leave the treatment of strong shocks by PPM unchanged, since we could not see how to improve upon it, and to apply it on adaptively refined grids, with a local refinement factor of 4. In the remainder of the flow, on a 4 times coarser grid, we intended to use a piecewise-parabolic scheme that could potentially enhance the accuracy of PPM in smooth flows dramatically. Much work was performed based on this idea. It did not result, as originally planned, on a hybrid AMR code, but it should be clear from the discussion below that this investigation has been very fruitful.

Cell-by-cell AMR for PPM:

The first element of this work was the development of a cell-by-cell AMR version of PPM, with refinement taking place only at strong shocks, contact discontinuities, and at reflecting walls. This work was described at the MICS program PI meeting in Chicago, and it has recently been accepted for publication (see Dingo and Woodward on the reference list). The idea here was certainly not to re-invent AMR, but to build an AMR code with the special (and limited) properties required by this project. We decided to use a single time step for all grid cell sizes, since there is little computational efficiency to be gained from multiple time step values, as the bulk of the grid cells are always on the finest meshes. Colella has pointed out that this design choice results in small Courant numbers on the coarser meshes. For his piecewise-linear scheme, this would reduce accuracy, but PPM has no such difficulty. For PPM the order of the scheme actually increases as the Courant number tends toward zero, with the result that overall accuracy shows only very minor dependence upon Courant number. We chose a cell-by-cell grid refinement approach, as is done in the RAGE code at Los Alamos, since we intended to refine the grid only at fronts (shocks and contact discontinuities). Block based grid refinement to capture front propagation accurately is very wasteful in fine grid cells. For example, the FLASH code at Chicago, which uses this approach to capture combustion fronts, has been forced to run problems using grid blocks only 8 cells on a side. These are very inefficient for difference schemes like PPM (which is used in FLASH) with large numerical stencils. Our AMR implementation also used a new technique to reduce the wasted computation at the edges of refined regions. Rather than constructing the equivalent of “ghost” cells for this refined grid, we used the considerable information from intermediate results of the coarse grid

computation instead. This cell-by-cell PPM AMR code is now being rewritten with only a single level of refinement at shocks and contact discontinuities and with an improved data structure that we believe will help it to fit into our SHMON parallel computing framework.

Our PPM AMR code has now taken on a new role. After the end of this project I began a close collaboration with the team of Mike Gittings and Bob Weaver at Los Alamos. The idea of this collaboration is to feed technology from the PPM codes into Mike Gittings' cell-by-cell AMR code RAGE. This collaboration allows full-blown tests and demonstrations of numerical techniques in AMR contexts to be made with the RAGE code, which is a full-featured production code. This means that our own PPM AMR code need only be used for simple testing, and it need not offer large numbers of refinement levels. Our team's experience with AMR showed that this technology adds a very large degree of complexity to numerical algorithm development. For a small university-based team like ours, I have come to believe that full-blown AMR is not practical. This decision is partly based upon the fact that most of the problems that my team has been addressing (see later sections) do not require AMR. Without our applications driving a requirement for AMR in our own codes, I have concluded that it is best to collaborate with others for whom we may develop modules for use in AMR codes. We still need a simplified testing capability for these modules locally, but full-blown testing is best done in the collaborators' codes. Our collaboration with Weaver and Gittings works well in this respect, since their codes share the cell-by-cell refinement approach and they permit immediate testing of new modules at full scale on today's very largest machines. Using our team's PPMLIB mechanism, we will continue to make the modules we develop available to others. The regular mesh versions of these modules should also prove suitable for incorporation into block-based AMR codes.

Better Piecewise-Parabolic Advection, PPB:

The main engine for increased accuracy of PPM calculations that we envisaged was not AMR, although it was seen to play an important role, but instead an algorithmic change. This idea was motivated by my experience with a piecewise-parabolic advection scheme invented by Bram van Leer (his Scheme VI in his "Towards IV" paper from the 1970s). Van Leer developed this scheme only in 1-D and did not come up with a mechanism to constrain it to preserve monotonicity. In my 1986 article (see reference list), I extended this scheme to 2-D and demonstrated its use on a gravitational two-stream instability problem. I also renamed the extended scheme PPB, for Piecewise-Parabolic Boltzmann scheme, since I thought at the time that its most useful application would be in simulations of advection in phase space. Over the years, I had found ways to simplify this scheme by discarding some of its highest-order components and to implement it in 1-D sweeps with or without monotonicity constraints. Many tests show that, as a rule of thumb, the simplest version of this scheme has roughly the resolving power of PPM on a mesh that is refined in every spatial dimension and time by a factor of 3. This fact is made somewhat less surprising when we consider that although PPB uses parabolas to describe subcell structures just like PPM, PPB uses 3 moments prescribed in each cell to determine these parabola. In 1-D, it therefore works with as much information as a PPM computation on a mesh refined by a factor of three. To see the advantages of PPB, it is best to consider an advection test problem. We will use the two-stream instability problem, since its winding flow is encountered in many physical situations (such as turbulence) and the winding continually introduces ever smaller features in the flow. Later, when we come to discuss present applications of this method within our PPM codes, we will produce more familiar test problems to illustrate the properties of the PPB method.

In order to test and demonstrate the performance and accuracy of advection schemes, I have embedded these in a gravitational two-stream instability program that I wrote for use in a computational methods course at the University of Minnesota. Each of the advection implementations discussed here has been compiled into a Fortran DLL to run inside this two-stream instability application (written in Visual Basic). This allows visualization of the results of the advection calculations and also measurements, through this user interface, of performance. In order to measure performance from within this

larger application, an option is selected through the user interface that causes the time spent in the Fortran DLL implementing the advection scheme to be measured and converted to a number of Mflop/s.

The two-stream instability application simulates the advection of a fluid in a 2-D phase space. The horizontal dimension, x , is a spatial dimension, while the vertical dimension, u , is a velocity dimension. The location of a point in the phase space plane gives its spatial location, x , and its velocity in the x -direction, u . If we plot the phase-space density, f , as a function of u at any particular location x , then this plot gives us what physicists commonly call the velocity distribution function. The integral of the phase-space distribution function f over velocity is the density ρ in configuration space. This density is interpreted in this problem as a density of stars, each of which is a point mass far too small to collide with any other star during the time of our simulation. These stars interact through the gravitational force, which is computed by finding the Fourier transform of the density distribution ρ .

In our two-stream instability experiment, we assume periodic boundary conditions in the spatial dimension, x . We initially prescribe the spatial density ρ as a constant

plus a cosine function that makes this density 50% larger in the center of our problem domain than it is at the edges. This large initial perturbation gets the problem moving interestingly right away. We also initially set up the phase space density f so that the velocity distribution function at each value of x consists of the superposition of two Gaussians, centered at velocities of $\pm u_0$ and with standard deviations of c_0 . Each Gaussian thus corresponds to a group of stars moving either to the left or to the right. For the cases we will present here, we choose $u_0 = 4$ and $c_0 = 2$, so that these groups of stars are initially very well separated in velocity. Each group is 50% denser at the center of our problem domain, and therefore all stars are accelerated to one mild degree or another toward this point. This acceleration makes the phase space distribution move upward in the left half-plane and downward in the

Figure 1a-c. The phase space distribution function at times 0.2, 0.4, and 0.6 in the gravitational two-stream instability problem described in the text. This advected distribution was computed using the PPBshear advection scheme on a grid of 512x512 cells.

right half-plane of our problem. By symmetry, accelerations vanish both at the center of the domain in x and at its edges.

The initial phase space distribution function is

$$f(x,u)=\left(\frac{1+0.5\cos(\pi x/L)}{2\sqrt{2\pi}c_0}\right)\left\{\exp\left[-\frac{(u-u_0)^2}{2c_0^2}\right]+\exp\left[-\frac{(u+u_0)^2}{2c_0^2}\right]\right\}$$

The problem is further specified by setting L , half the periodic length in the spatial dimension, to 6 and by setting the value of the gravitational constant to 2π .

The time development of this gravitational two-stream instability, which is very similar to that first presented in Woodward 1982 and 1986, is shown in the figures here using the Visual Basic program running the PPBshear scheme, the most accurate of various alternative efficient versions of PPB, all of which are simpler than that described in Woodward 1986. These results, computed on a very fine grid of 512×512 cells, may be regarded as showing the correct time development of the phase space density f for this problem. The first of the images from this run already shows that the two Gaussians have moved upward in the left half-plane and downward in the right half-plane. As this problem proceeds, a circulation about the center of the problem domain develops. This circular motion in phase space corresponds, of course, to a back-and-forth oscillatory motion in configuration space for the two groups of stars. The Gaussian strands, or streams, in phase space thus wind about each other ever more tightly as the evolution proceeds. Ultimately, at any given resolution we can find a time so late that the strands can no longer be distinguished. The system thus relaxes to a much hotter system of stars in which two separate velocity groups of stars can no longer be distinguished. Diffusion, which is inevitable on scales near that of our grid cells, must then ultimately bring this system into a steady, fully relaxed, equilibrium state.

Figure 1d-e. The phase space distribution function at times 0.8 and 1.4 in the gravitational two-stream instability problem described in the text. This advected distribution was computed using the PPBshear advection scheme on a grid of 512×512 cells.

This two-stream instability problem is well known in plasma physics, where the force law has the opposite sign. Although it is less familiar in astrophysics, it is in fact responsible for the process called dynamical friction that causes randomization of stellar velocities in galaxy mergers near the centers of rich clusters of galaxies. For this problem, the advection in phase space is particularly simple. Along each horizontal strip of grid cells, the fluid all moves to the left or right at the identical speed. Similarly, along each vertical column of grid cells, the fluid all moves upward or downward at the same speed as well. Nevertheless, because the total amount of phase space fluid is strictly conserved, the swirling flow

in phase space stretches the original Gaussian streams out into ever thinner bands, which become progressively more and more tightly spun around the center of the problem domain, like a watch spring. As time advances, the advection problem therefore becomes progressively more difficult, and any advection scheme must ultimately break down no matter how fine a grid we choose at the outset. We can see this behavior in the snap shots shown here from the PPBshear run on the 512×512 grid. The developing flow is easily resolved on this grid in the first snap shot, but in the second, at problem time 0.4, the Gaussian streams have become extremely thin near the edges of the problem domain, so that their separate identities can no longer be maintained by the advection scheme. Nevertheless, on this fine grid and with this very accurate advection scheme, these two streams are still well resolved near the center of the problem domain for the entire duration of this experiment.

The behavior of the PPBshear scheme as it becomes unable to correctly describe the evolving phase

space distribution function is quite benign. It is worth noting that in this problem, when individual streams become so thin that they can no longer be accurately tracked, they tend to be located near other such streams, and the streams tend to be merged together by the advection scheme. This behavior can be seen in the snap shots above at problem times 0.4 and 0.6. Although the two separate streams can no longer be distinguished along some of their lengths, a merged stream has been produced that is being quite accurately tracked as it moves through the grid. This PPB advection scheme conserves 9 moments of the phase space distribution function to machine round-off accuracy. As a result, the single stream created by merging the two separate, unresolved streams must preserve all those 9 moments of the more complex, unresolved structure. In our two-stream instability problem, this is true except insofar as errors in the phase space distribution function result in errors in the density ρ in configuration space, which in turn create errors in the gravitational accelerations, which are the advection speeds in the vertical direction in our phase space.

We can consider the PPBshear results of the two-stream instability problem on the 512×512 grid that are presented above to be representations of the exact solution of this sample problem, since they are demonstrably far more accurate than any others we will present. At the same time, it is clear from the results shown here that there is a convergence of the results of this and the other advection schemes as the grid is progressively refined. Evidence of that assertion for the PPBshear scheme is given in the final panel, Figure 1f, where results from running the same problem on a much coarser grid of only 128×128 cells are shown below those obtained on the 512×512 grid. The flow is displayed at a problem time of 3.0, when the phase space distribution function has become rather thoroughly wound up.

Figure 1f. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. In the upper panel, this advected distribution was computed using the PPBshear advection scheme on a grid of 512×512 cells, while in the lower panel a grid of only 128×128 cells was used. The extent to which the solution has converged on the finer grid is evident.

In the figure at the right (Figure 2), a comparison of the results for this problem are shown for the PPBshear scheme and for two variants of the PPM advection scheme, one with and one without the scheme's contact discontinuity steepener activated. The PPM contact discontinuity steepener, first introduced in 1984 (see reference list), is a physically motivated yet nevertheless somewhat ad hoc procedure intended to keep sharp jumps in an advected variable (the entropy, for standard, single-fluid PPM gas dynamics) sharp despite the continued action of numerical diffusion. The superiority of the PPBshear results is clearly evident. The relative costs, both in computational labor and in computation time, for these three runs are given in the figure caption. The PPBshear scheme produces a higher quality result in half the time. A variant of the PPB scheme using only 6 moments delivers roughly the same accuracy as PPM on a grid of one third the spatial resolution in just 7.8% of the time. This result is shown on the next page. This scheme, which we call PPB6, uses only 215 flops per grid cell update in 2-D and runs at 1428 Mflop/s on a 2.53 GHz Pentium-

Figure 2a-c. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. In the upper panel, this advected distribution was computed using the PPBshear advection scheme on a grid of 128x128 cells, while in the lower 2 panels the PPM advection scheme was used on a 384x384 grid. PPBshear requires 928 flops per cell update, while PPM without contact steepening (middle panel) requires only 109 and PPM with steepening requires only 149. Also, PPBshear runs at 998 Mflop/s on a 2.53 GHz Pentium-4 PC, while PPM without steepening runs at 1521 Mflop/s and with steepening runs at 1270 Mflop/s on that machine. Thus, the results of the PPBshear calculation are definitely superior, and they are obtained in less than half the computer time required for the PPM computation without steepening. The cost advantage for PPBshear is substantially larger than the factor of 2 in computer time, since the PPBshear results are more accurate. In 3-D this cost advantage would be the same, since there would be an additional factor of 3 fewer cell updates relative to PPM but there would also be 3 times as many moments to update for each cell. In the lower panel, the advected distribution was computed using the PPMstpn advection scheme on a grid of 384x384 cells. The contact discontinuity steepening algorithm has introduced some stair-step features by enhancing sharp edges detected in the distribution when those edges were advected in directions near the principal grid diagonals. Although tweaking of the dimensionless constants that control the aggressiveness of this steepening is possible, the generation of such false signals is difficult to avoid entirely without eliminating the benefits of the technique. Especially in situations where these false signals can stimulate fluid instabilities, such as Kelvin-Helmholtz instabilities, one is far better off using one of the PPB methods, such as PPBshear, to make sure that features that should stay sharp do so.

4 CPU. In 3-D, we would expect the PPB6 advection scheme to be 23 times more efficient than PPM, since it would gain an additional factor of 3 advantage but have to update 10 rather than just 6 moments.

The PPM scheme with contact discontinuity steepening produces some stair stepping on this problem, as can be seen in the figure. For this problem, the contact discontinuity steepener in the PPM scheme produces some stair-stepping of artificially steepened gradients, a flaw of this sort of scheme for transport at nearly 45° to the mesh that is hard to remove without eliminating all benefits of the discontinuity steepening.

Using PPB to advect Riemann invariants in a hydrodynamics computation:

During the last grant period Wenlong Dai and I developed a new piecewise-parabolic gas dynamics scheme that incorporated the PPB advection scheme for all Riemann invariant fields. In addition, it used the advected Riemann invariant data to obtain approximate solutions to

nonlinear Riemann problems which were used to construct fluxes of the conserved quantities. The cell averages of the Riemann invariants were constructed from applying these fluxes to the conservation laws, so that strong shocks could be properly treated by the scheme. The higher moments of the Riemann invariant distributions were updated using the PPB technique. Because of the need to go back and forth between Riemann invariants and conserved quantities, which involves the solution of a nonlinear system, the resulting scheme was very computationally complex. In one dimension, it fully delivered the increased accuracy we were led to expect relative to PPM based upon the results of advection tests like those just discussed. However, in multiple dimensions, an unanticipated feature of the method came to light.

The primary benefit of the PPB method is increased resolving power while still using parabolae for interpolation. The use of parabolae, rather than still higher order polynomials, keeps the arithmetical complexity under control and makes the application of monotonicity constraints relatively straightforward. The increased resolving power makes possible the accurate treatment of disturbances that have shorter wavelengths measured in grid cell widths. A key to our approach in implementing PPB schemes is their formulation in terms of 1-D passes, just as we have always used with the PPM schemes. The 1-D passes greatly simplify the schemes, and they make very high performance implementations possible for today's microprocessor CPUs. In principle, there are special errors that are introduced by directional operator splitting, although these are to some extent compensated for by the method's natural handling of corner transport. With the PPM schemes, we were never able to identify specific shortcomings of simulations that resulted from directional splitting, although we knew that splitting errors limited PPM to second

Figure 3. The phase space distribution function at time 3.0 in the gravitational two-stream instability problem described in the text. In the upper panel, this advected distribution was computed using the PPB6 advection scheme on a grid of 128x128 cells, while in the lower panel the PPM scheme without discontinuity steepening was used on a 3 times finer grid of 384x384 cells. The results are of comparable accuracy.

order formal accuracy. Our PPB-based gas dynamics scheme, however, revealed the nature of these splitting errors to us for the first time. For the very shortest wavelength sound waves, propagation speeds were artificially slowed, slightly, when the waves traveled obliquely to the mesh. Waves that would be so affected by the PPM scheme have such short wavelengths that they cannot propagate at all, since the PPM numerical dissipation strongly damps their amplitudes. However, the PPB advection scheme is so accurate that it does not damp these very short wavelength sound waves. To produce this phase error in any noticeable degree, the wavelength of the sound wave must be comparable to the grid cell width (such as a wavelength of just 4 cells).

The phase error of the PPB gas dynamical scheme for very short wavelength sound waves arises from the misidentification of waves that results in a 1-D pass of a multidimensional computation. Consider for example the propagation of a small amplitude sound wave in the diagonal direction on a 2-D Cartesian grid with uniform, square grid cells. In each 1-D pass, this wave appears to be two waves propagating in opposite directions. These two waves are shown in the space-time plots in the lower two panels at the right. The sum of these waves moving in opposite directions gives the x-velocity signals shown in the top panel. At the initial instant, the two waves are launched perfectly in phase, and they interfere in such a way that the x-velocity signal propagates at precisely the square root of 2 times the sound speed. However, as time passes, the x-velocity signal slows until it actually propagates to the right more slowly than the sound speed (see the portion of the space-time plot in the top panel at the right that is about a third of the way up from the bottom). The signal speed fluctuates about the speed of sound, moving first faster, then slower, then faster again.

The wave propagation problem shown at the right, using one of the PPM-based programs developed for instructional purposes, displays the long-time solution that is approximated in the x-pass of an operator split numerical algorithm. If this solution is utilized for only a small fraction of the period of the oscillation of the wave speed about the speed of sound, so that the signal propagates only a

tiny fraction of one wavelength, then its propagation speed, $\sqrt{2}$ times the sound speed, is just what we require. If instead it is used for an entire half wavelength of signal propagation, then its average propagation speed is the sound speed, too small for our purposes by a factor of $1/\sqrt{2}$.

This behavior comes purely from the directional operator splitting, and not from any other property of the numerical algorithm. It should therefore result in errors in the propagation speeds of oblique sound waves for both the PPM and PPB schemes. However, PPM strongly damps all waves of such high frequency that the Courant condition would allow them to move more than a trivial fraction of a half wavelength on a single 1-D pass. Therefore the error does not appear at a noticeable level in PPM simulations. PPB, on the other hand, is so accurate in its 1-D computation that it can maintain undamped a sound wave of only 3 or 4 cells per wavelength. At a large Courant number (one near unity), such a high frequency sound wave will move too slowly through the grid if it propagates in a diagonal direction. Judging by eye in the space-time plots given here, this effect would be noticeable in just a few time steps only for oblique sound waves of wavelengths near $2\Delta x$ that are computed with Courant numbers near $3/4$ or so. However, if we run for a large number of time steps, the error will show up for sound waves of larger wavelengths and for smaller Courant numbers.

After realizing this effect and understanding its inescapable nature, we decided that the best way to utilize the tremendously accurate PPB advection scheme within gas dynamical computations was to apply it to Riemann invariants, such as the entropy, that propagate at the fluid velocity. When applied to these Riemann invariants, the PPB scheme with directional splitting introduces no special phase error of the type discussed above for sound waves. Such an application of PPB advection within PPM gas dynamical computations is presently under development, and I will show some preliminary results, achieved after the end of this project but using the algorithmic technology developed under this project, in the next section.

Although I did abandon development of PPB advection of sound wave Riemann invariants in directionally split numerical schemes, an application of this technology has recently come to light through a new collaboration with the RAGE code team at Los Alamos. The cell-by-cell AMR technique implemented in the RAGE code makes it extremely difficult to introduce into that code numerical methods like PPM that use wide difference stencils. As one looks to grid cells further and further away from the cell of interest either to the left or to the right in a 1-D pass, the nonuniformity of the AMR grid becomes a strong impediment (this is of course not the case for block-based AMR). The RAGE team has therefore requested that the more accurate moment-conserving hydrodynamical schemes, using PPB advection of Riemann invariants, be tested in the RAGE code. These schemes have optimally narrow difference stencils, achieved at the cost of introducing additional moment data in each grid cell. Such experimentation is planned in the coming year. It is an unexpected outcome of the research supported by this project, but like the multifluid interface tracking discussed below, it has the potential to bring substantial benefits to hydrodynamical simulations at the DoE labs. For AMR codes like RAGE, it is important to note that the grid cells on all but the finest levels will be advanced with small Courant numbers, for which the phase errors from directional splitting will be very small even for very short wavelength sound waves.

Results for the PPB6 Advection Scheme Embedded in a PPM Gas Dynamics Code

The moment-conserving advection schemes discussed here can be most naturally applied to the advection of a distribution function of particles in a 4- or 6-D phase space. In such an application, the advection has very special properties. Along each grid strip in any of the 6 dimensions, all grid cells move at precisely the same velocity in precisely the same direction. This fact can be exploited to collapse some of the computation involved in the PPB schemes and thus to make them more cost effective. Also, for such applications a simple constraint that the particle distribution function be positive is sufficient, so that the PPB schemes can be made more cost effective still. Given the high dimensionality of these problems, these simplifying features are very welcome. Also, the high resolving power of the PPB

schemes is welcome, since without this property the number of grid points required to span the appropriate regions of a 6-dimensional phase space would be prohibitive, even on the very largest computers. Given the accuracy of the PPB schemes and the expected clumping of the particle distribution function in limited regions of the velocity dimensions for many physics problems of interest, it should be possible to obtain an adequate description with only 20 to 30 grid cells per velocity dimension. Since the PPB6 scheme with positivity constraints executes on a 2.5 GHz Intel Pentium-4 CPU at over 1.4 Gflop/s, it should be practical on modern supercomputers to use up to 10 billion grid cells. This would allow 20 cells in each of 3 velocity dimensions and 100 cells in each of 3 spatial dimensions, with 2 billion cells left over. The computations of the two-stream instability problem we have just seen on 128×128 grids give some idea of the resolution we would be able to achieve in such a calculation.

A less obvious but perhaps more useful application for the PPB schemes for DoE problems is advection of especially important variables within fluid dynamics simulations. Such a special variable could be the entropy, or, as the meteorologists call it, the potential temperature in a climate simulation. Also, wind borne pollutants or chemically reacting trace concentrations could be such special variables. A particularly interesting potential application is to use the PPBshear or PPB6 scheme to advect the fractional volumes of different gases in multifluid gas dynamics problems. These fractional volume variables play a special role deserving of highly accurate treatment because they tend to delineate multifluid interfaces where a variety of fluid instabilities develop that can profoundly affect the global development of a fluid flow. Numerical diffusion at these interfaces is especially to be avoided, since it is not only unphysical but it also can profoundly alter the simulated flow behavior. I am now working in collaboration with the RAGE code team at Los Alamos to explore PPB6 multifluid interface tracking in both the PPM and RAGE codes.

Like the Boltzmann (or Vlasov) equation problems of stellar dynamics or plasma physics discussed above, advection of fluid fractions in gas dynamics problems involves certain simplifications that tend to counter the potential cost of using the PPB schemes. In most such problems we know that in the vast bulk of the grid cells the fluid fraction variables will have either the value 0 or 1. In such regions there is really no work to do. All the computational labor in such advection problems can be concentrated on the multifluid interface and mixing regions, which are likely to occupy only a small fraction of the problem domain. This special feature of multifluid advection can reduce not only the computational requirements but also the memory requirements. In any cell where the fluid fraction is either 0 or 1 (the vast bulk of the cells), there is no point in storing any high-order moments of the distribution.

The key concern for using an advection scheme such as PPBshear or PPB6 for multifluid problems is numerical diffusion of the multifluid interfaces. Multifluid gas dynamics has for many decades been pursued either by use of Lagrangian coordinates, in one form or another, so that numerical diffusion of multifluid interfaces vanishes, or it has been pursued using interface tracking techniques of one type or another. We will not discuss Lagrangian techniques, since they make little sense in 3-D unless the flow is of such a character that it is basically 1-D in any event. Interface tracking techniques attempt to introduce subgrid-scale information about the configuration of the fluids in mixed cells. The demand that the reconstructed multifluid interface be infinitely thin in such approaches introduces the potential for a variety of numerically generated glitches, since at the scale of a grid cell the representation of the multifluid interface, or at least its numerical treatment, simply cannot be smooth.

We adopt here the philosophical position that a grid cell represents a sort of quantum limit for a numerical simulation, and it therefore does not make sense to construct elaborate and detailed representations of a grid cell's interior. One might object that by, potentially, prescribing up to 27 moments of the subgrid-scale structure we are violating our own philosophy. However, these 10 or 27 moments serve only to determine a smooth, internal structure, with no sudden internal jumps and, in the 10-moment case, with the character of a simple parabola along any particular line through the grid cell. Our large quantity of subgrid-scale information serves only to make the determination of our smooth internal representation of the function highly accurate. We do not use this information to insert into the cell interior sudden

I have decided to design such a test of PPB6 advection within the classic wind tunnel simulation problem that was originally introduced by Emery in 1967 and which I discussed at great length in my 1980 and 1984 articles with Phil Colella (see reference list). In this 2-D gas dynamics problem, air (or any other gamma-law gas) enters a 2-D duct, or wind tunnel, supersonically at the left, travels down the duct, striking and flowing around a forward facing step on its way, and flows out of the duct supersonically at the right. The supersonic flow-in boundary condition at the left is trivially implemented, the supersonic outflow at the right is also easily treated, and the walls of the duct are modeled as impenetrable, thermally insulating, perfectly reflecting surfaces. In the 1984 discussion with Phil Colella, we pointed out that the corner of the step in this wind tunnel is a singular point, where the gas is invited to turn a corner instantly – a feat that, although possible in theory, is impossible in practice (and, for that matter, also impossible in nature). In order to make 7 different numerical schemes all converge to a single result on grids that today seem ludicrously coarse, we decided to force the flow solution at the corner of the step to behave “properly.” In the test program implemented here, I do not do anything special at the corner of the step, leaving the numerical scheme to generate entropy there through the action of numerical viscosity. This creates a boundary layer along the top of the step, and the presence of this boundary layer, of numerical origin, profoundly affects the macroscopic flow behavior. Nevertheless, the simulated flow converges as the grid is refined, and it is interesting to speculate whether or not this converged solution represents a limit of Navier-Stokes viscous flows as the viscosity tends toward zero. However, these details are irrelevant to our desire and ability to test the capability of the PPB6 advection scheme to keep multifluid interfaces sharp.

added 8 user configurable streams of smoke which enter the duct through the left-hand boundary. These smoke streams are passively advected with the flow, and they are advected by the PPB6 scheme.

I have chosen a wind tunnel problem that exercises the PPB6 advection scheme especially well, including testing its behavior tracking interfaces caught up in developing Kelvin-Helmholtz instabilities. The smoke streams near the outset of this problem are shown along with the user interface controls in the figure at the top of this page. Air enters the wind tunnel at the left at Mach 4. There are 8 streams of smoke, inside each of which the fractional volume variable that we advect with PPB6 has the value 1. We generate a smoke visualization by rendering an image of the product of this smoke fractional volume and the density of the air. Therefore, for example, the developing bow shock in front of the step is clearly seen by the sudden brightening of the smoke streams that pass through this front. To add visual interest, the initial density inside the wind tunnel was given a linear variation, assuming its smallest value, 1, at the lower left corner, 3 at the upper left and 5, at the upper right corner. The air entering the wind tunnel, however, has a uniform density of 1 in the bottom $3/8$ of the duct width and a uniform density of 1.25 in the upper $5/8$. The Mach number of this entering air is everywhere 4, so that the denser, upper air is traveling less rapidly down the duct.

The initial pressure and density mismatches along the wind tunnel entrance cause, along its entire length, shocks to run into the entering air and rarefactions to run into the air originally in the duct, leaving contact discontinuities at the original location of the entrance in the inflowing air. These three wave fronts can be distinguished from the shade variations in the smoke streams in the first figure of this sequence, which shows the flow at time 0.12. Because of the slower flow down the duct in its upper $5/8$, all three of these wave fronts jog to the left $3/8$

of the way up from the bottom toward the top of the duct. The development of this wind tunnel flow, as visualized by the 8 smoke streams advected by the PPB6 method, is shown in the image sequence on this and the next few pages. These images, which, except for the first one at time 0.12, show the flow at time intervals of 0.3 (beginning at time 0.3), come from a simulation on a grid of 1024×512 cells, for which each of the entering smoke streams is 32 cells wide.

The images from the 1024×512 simulation give a sense of the "correct" flow behavior; they are not intended to show off the accuracy of the PPB6 advection. Nevertheless, one can marvel at the entrainment of these smoke streams in the many vortices, both small and large, that develop within this flow. Also, the definition of the very thin streams of smoke that result from the strong shear associated with the double Mach reflection of the bow shock at the upper wall of the duct is exceptional. The tracing of the strands of smoke entrained in the vortex at the tip of the jet that squirts forward along the upper wall also poses a strong challenge to any advection scheme. What is absolutely absent from these images is any noticeable diffusion of the edges of the smoke streams. To see that such diffusion is truly absent, we must examine runs of this same complex flow problem on much coarser grids, where any errors will appear much more glaringly to the eye.

The images in the sequence just discussed are generated by representing each grid cell with 4 subcell values. These values are interpolated for the smoke fractional volume using the moment data, carefully constrained in 2-D to lie in the range from 0 to 1. The density values in the 4 subcells come from PPM cell average data that has been interpolated with the PPM algorithm in both the x- and y-dimensions. The behavior of the subcell density distribution is not constrained along the cell diagonals, so it is possible to obtain small negative density values from this interpolation process. These

were not removed, since they appear at roughly the 1% level, and they do not mar the visualizations. However, if we are to assess the resolving power of the PPB6 advection in detail, it is best to change the wind tunnel simulation program so that only cell averaged values are displayed. This will also help us to see the individual grid cells in the resulting displays, and thus to assess more precisely the behavior of the advection scheme.

To see clearly what the PPB6 advection scheme is doing at the scale of individual grid cells, we need to reduce the resolution of the wind tunnel simulation. On this page are a series of smoke visualizations in which only grid cell averages of the product of the density and the fractional volume of smoke are shown. The grid used in this simulation is 256×128 cells, and it is possible by careful inspection of these images to pick out the individual cells. At this grid resolution, each smoke stream enters the duct at the left precisely 8 cells wide. Although a simulation on a 128×64 grid would make the individual cell values clearer, its smoke streams, at only 4 cells wide, would pose an unreasonable challenge to the advection scheme (see final image sequence).

The third image in the sequence at the top of this page clearly shows that the smoke stream nearest to the upper wall of the duct is sheared and compressed into a stream less than 2 cells wide which is transported in a swirling flow without loss of definition. This behavior of the PPB6 advection scheme is remarkable. One might argue that the path traversed by this smoke stream is rather short, so that diffusion does not have so long to act in spreading the stream out, but the various smoke streams do not appear to become much less distinct as they travel down the length of the wind tunnel, they only become dimmer, since the gas density decreases in this region. In the fifth image of this sequence, the smoke streams are sheared into thin strips that are only one cell wide and that are separated by only one cell. It is truly

remarkable that these thin strips remain distinct until they are crowded together even more closely than this.

In this coarser flow simulation a minor flaw, caused by PPM and not by PPB6, is very noticeable in the second and third images of the sequence. This is a brief stretch of several of the smoke streams that develops a zig-zag shape after passing through the bow shock. An examination at this point in the problem of the distribution of the vertical component of velocity reveals that this has developed a brief region of oscillatory behavior just behind the bow shock. This is a numerical instability first identified in my article with Colella in 1984. It is caused by a slowly moving strong shock front that is nearly aligned with the mesh. In 1984 we called this instability a Cray instability, because at the time one needed to own a Cray supercomputer in order to perform simulations on sufficiently fine meshes to see it develop. Here, in 2002, it has developed on my laptop machine.

In 1984 we identified the cause and a solution for this instability. In the present version of PPM, a still more effective form of this solution approach has been implemented, which explains why this oscillatory behavior of the post-shock velocity field develops only briefly at one time and location in this problem. Setting the coefficient of the smart diffusion in PPM a bit larger, from its default value of 0.3 to, say, 0.5, completely eliminates this behavior. However, it does so at the cost of spreading out the bow shock so that the definition of the resulting solution is degraded. The choice of the value 0.3 is a compromise, admitting slight known inaccuracies, which in this case are amplified by the highly accurate PPB6 smoke advection. The vertical velocity oscillation that causes the zig-zagging of the smoke streams is an oscillation of such high frequency that it is immediately damped by PPM and is invisible in the density and pressure plots at the times of these smoke visualizations, but its effect on the smoke streams is accurately preserved by the PPB6 advection.

Building PPB6 advection of multifluid fractional volumes into a true multifluid PPM code:

The above preliminary results illustrate PPB6 advection embedded in a full hydrodynamics simulation. However, to use PPB6 to track multifluid interfaces we need to incorporate the method in a true multifluid version of PPM, in which the two fluids can have very different equations of state. Then the advection is not passive, as it is in the examples for smoke advection just discussed. Instead the PPB6 advection will react back on the flow dynamics. Such development is now underway as a part of my new collaboration with the RAGE code group at Los Alamos.

The development work with PPB advection schemes and their incorporation into gas dynamics simulation codes that was sponsored by this project especially during the final grant period has led to a vigorous program of research that is fueling an active new collaboration with researchers at Los Alamos. New schemes will first be tested within the relatively simple context of PPM gas dynamics codes and in the simplified PPM cell-by-cell AMR code. However, it is the very much more complicated and difficult tests that can be carried out within the RAGE code that will make the case for significant use of these new

algorithms in major DoE programs. The results of those tests are of course not yet in, but in my opinion the development to date has enjoyed considerable success.

Large-Scale Simulations of Turbulent Flows in 3-D

This project has supported application of the numerical algorithms developed under the grant to large-scale fluid flow simulations of fundamental interest. In the early years of the project, these simulations involved 2-D shear instabilities in the supersonic regime, with active collaborations with the groups of Andy Majda, then at Princeton, and Karl-Heinz Winkler at Los Alamos (see reference list). More recently the focus of this work shifted to 3-D simulations of turbulent flows, both homogeneous, compressible turbulence and such turbulence driven by flow instabilities such as convection and the Rayleigh-Taylor and Richtmyer-Meshkov instabilities of multifluid interfaces. This work involved active collaborations with Annick Pouquet, then in France and now at NCAR, Karl-Heinz Winkler at Los Alamos, and with the ASCI turbulence group at Livermore, then led by Bill Dannevik. A new collaboration is now developing in this area with Daryl Holm, Len Margolin, and Bob Weaver at Los Alamos. Over the life of this project, this work on large-scale simulations has driven a very active research program in flow visualization as well.

Two contrasting visualizations of homogeneous, compressible turbulence as computed on a 1024^3 grid with the PPM scheme in 1993. In the upper panel, vorticity structures near the dissipation range of length scales are shown in a small region of the fully developed turbulent flow. In the lower panel, the data has been filtered before the vorticity image was rendered. These vorticity structures are in the Kolmogorov inertial range of length scales. Each region shown is the same width relative to that of the vortex tubes it contains. The two images here were rendered for direct and unbiased comparison. In each sub-volume shown, the same volume fraction is occupied by each opacity/color level. The very different appearance of these images therefore reflects real differences in their dynamics. The relatively straight vortex tubes near the dissipation range do not readily kink to form still smaller structures.

In the beginning of this project, it was evident that the numerical dissipation of the PPM scheme was ideal for simulating turbulent flows. This dissipation targets only the highest wavenumbers using a combination of standard numerical dissipation, monotonicity switches, and variable diffusion coefficients based upon the character of the local flow. This dissipation acts locally, depositing the dissipated kinetic energy in place in the form of heat. At the same time the high-order features of the PPM scheme result in very little numerical dissipation of low frequency disturbances. We exploited these features of PPM in our extensive studies of turbulent compressible convection (under NSF and NASA support) and of turbulent shear flows over the years. In the late 80s we undertook, through a long and productive collaboration with Annick Pouquet, to apply PPM technology to the study of homogeneous, compressible turbulence. The aim of this work was to determine a subgrid-scale model of turbulence that could be added to our PPM codes. The focus was therefore on obtaining high quality data from numerical experiments. The PPM Euler technique, with its very special numerical dissipation, allowed us to limit

The PPM Euler technique, with its very special numerical dissipation, allowed us to limit

the range of scales in which viscous effects are important to a very narrow range near the highest resolvable wavenumbers. This made possible numerical experiments that could accurately capture the statistical behavior of the flow in the range of scales in the Kolmogorov inertial range, where viscous effects are negligible.

Our first major discovery in this work was a flattening of the velocity power spectrum just above the dissipation range, with a characteristic k^{-1} velocity power spectrum (see the upper sequence of spectra at the right, for the solenoidal component of the velocity field, taken from Sytine et al. 1998). We reported this discovery in 1991 and 1992 (see reference list) and found from private discussions with other researchers that this behavior had been observed before, but apparently the calculations were not of sufficient grid resolution to give the investigators involved the confidence in the results to report them publicly to the community. Our initial discovery was made with a PPM simulation of homogeneous, compressible turbulence on a grid of 256^3 cells, which

was regarded as an enormous grid at the time. Using the University of Minnesota's new CM-5 Connection Machine a year or two later, we were able to increase this grid resolution to 512^3 , a record at the time. With this resolution in a simulation of decaying turbulence, we saw a clear section of the Kolmogorov inertial range, of about a factor of 3 in wavenumber, followed by the flattened region of the spectrum and then by the dissipation range. We interpreted the flattening of the velocity power spectrum as a bottleneck in the forward transfer of energy through the turbulent cascade. We presumed that it was caused by the absence of active modes at higher wavenumbers with which the modes in the flattened region of the spectrum would need to interact in order to transfer their energy. We also visualized the flow in the flattened part of the spectrum and found that it consisted of a tremendous tangle of long, thin vortex tubes (see the figure on the previous page). This made sense, because the Fourier transform of a single, long, straight vortex tube has a k^{-1} behavior. We also filtered out these high wavenumber motions from our data and visualized the flow structures on larger scales within the Kolmogorov inertial range. We found that these vorticity structures were not nearly so long and thin, resembling elbows of pasta, macaroni, in contrast to the "spaghetti" of the near dissipation range (see figure). It is clear in the figure on the previous page that the structures in the Kolmogorov inertial range are more chaotic, involving less coherent structures than we see in the near dissipation range. We therefore concluded that in order to obtain data from numerical experiments that could tell us the dynamics and statistics of the Kolmogorov inertial range, we needed to go to truly enormous grids. In 1993, in collaboration with Silicon Graphics, we therefore performed the first 1024^3 turbulence simulation.

The development of the velocity power spectrum with increasing grid resolution can be seen in the figure on the previous page, taken from Sytine et al. 1998. As the grid is refined, the flattened region of the spectrum moves progressively to higher wavenumbers, revealing an increasingly long Kolmogorov inertial range. With PPM, the inertial range appears in these decaying turbulence simulations first at 512^3 grid resolution. We also reported in Sytine et al. (1998) results of Navier-Stokes simulations with our PPM code with added dissipation terms. We also made the unusual demand that these smooth Navier-Stokes flows have converged velocity power spectra, in the sense that if the computational grid were refined by a factor of 2 in each dimension and time but the coefficients of viscosity and heat conduction were left unchanged, the velocity power spectrum would also remain unchanged. We found that for these resolved Navier-Stokes flows there was no Kolmogorov inertial range at all on a 512^3 grid, the highest grid resolution that we used. Instead, the Navier-Stokes flows showed the same flattening of the power spectrum before the dissipation range was reached that we first saw in our PPM Euler simulations. The flattening of the spectrum was a bit broader than we observed in PPM simulations, but appeared to be clearly the same physical phenomenon. Many in the turbulence community had resisted the claim that this flattening of the spectrum was “real” rather than just an effect of our particular code’s dissipation. Finding it clearly present in Navier-Stokes simulations thus settled this issue, we thought, and made our initial discovery of the phenomenon more significant. At the Supercomputing 2002 conference, Japanese investigators presented velocity power spectra from Navier-Stokes simulations of turbulence carried out on grids as fine as 4096^3 . These results, shown on this page, confirm our discovery beyond doubt. Note that the spectra in the figure turn upwards at the highest wavenumbers, so that they are clearly not resolved Navier-Stokes simulations in the sense defined earlier.

Compensated velocity power spectra for Navier-Stokes simulations of homogeneous incompressible turbulence, taken from Yokokawa et al. 2002. The Kolmogorov trend is horizontal for these compensated spectra. Note the flattened portion of the spectra, here visible as a bump, in the region just before the dissipation sets in.

Building a subgrid-scale model of turbulence for use in the PPM code:

The above research, supported by this project over several years, established the need for PPM simulations of turbulence on grids of 1024^3 or, preferably, 2048^3 in order to generate reliable data for use in designing and validating subgrid-scale turbulence models. We have built an entire infrastructure to handle the many terabytes of simulation data and the many statistical functions we wish to perform on the data. This infrastructure allowed us, we believe for the first time, to make visualizations of such abstruse but important quantities as the rate of forward energy transfer from large scales to small in a turbulent flow. This capability has led to physical insights that would otherwise not have been possible.

We have derived what we believe to be a new insight from our study, with the Livermore team, of the Richtmyer-Meshkov instability of a shock-accelerated multifluid interface (cf. Woodward et al. 2001, Cohen et al. 2002, Woodward et al. 2002). Perhaps largely as a result of the symmetry of the Richtmyer-Meshkov flow we studied, which is shown in the figure on the next page, we perceived a visual correlation between the local topology of the flow and the transfer of energy down the spectrum from larger-scale motions to smaller-scale ones (this is generally referred to as the “forward” transfer of energy, since the turbulent cascade proceeds in this direction). This correlation checks out quantitatively (see below) for both the Richtmyer-Meshkov data and data that we generated in a PPM simulation of homogeneous, Mach 1 turbulence on a grid of 1024^3 cells. Briefly, we have found that the rate of forward transfer of energy tends to be positive (increasing turbulence) when the local flow is expanding in two

dimensions and compressing in the third, while it tends to be negative (decreasing turbulence) when the local flow is expanding in only one dimension and compressing in the other two.

How could we have missed this insight in the past? In order to see it, we needed to visualize the spatial distribution of the rate of forward energy transfer. Although we have been studying turbulence for many years, it was not until we obtained our data from the Richtmyer-Meshkov simulation on a grid of 8 billion cells, that we dared to generate an image of such a quantity. To make such an image, one needs to achieve a significant separation of scales within the flow. One introduces a Gaussian filter, which in the case of our Richtmyer-Meshkov data we chose with a full width at half maximum of 68 grid cells. (This filter damps a sine wave of wavelength 128 grid cells by a factor of $1/e$ and one with a 64-cell wavelength by a factor of $1/e^4$.) By filtering the data one averages over all the small-scale motions to reveal the flow structures on the larger scales. The formula for the rate of forward energy transfer involves a small-scale stress tensor, τ_{ij} , defined by: $\tau_{ij} = \overline{\rho u_i u_j} - \bar{\rho} \tilde{u}_i \tilde{u}_j$. In this formula, the overbars denote filtered quantities, while the tildes denote mass-weighted filtering: $\tilde{u}_i = \overline{\rho u_i} / \bar{\rho}$. The rate, F_{SGS} , of forward kinetic energy transfer from scales larger than the filter width to those smaller than this width is given by (cf. Woodward et al. 2001 or Cohen et al. 2002): $F_{SGS} = -(\partial_j \tilde{u}_i) \tau_{ij}$. To evaluate this quantity with any reasonable level of confidence from numerical simulation data, we must resolve the small-scale motions below the scale of the filter on our computational grid. The 8 billion cells of our Richtmyer-Meshkov simulation allowed us to use a filter of width 68 cells without smearing the simulation so much that little structure would survive the filtering process. At the same time this filter is wide enough so that our computation with a simplified version, sPPM, of our PPM code was able to do a good job of resolving the flow behavior on scales smaller than the filter width.

The Richtmyer-Meshkov Numerical experiment: Our simulation of the Richtmyer-Meshkov instability was intended to follow the experiments of Vetter and Sturtevant (1995), so that comparisons with those experiments might serve to validate the results. Such comparisons are given in Cohen et al. (2002), but of course the calculation yielded a wealth of detailed data on the flow that the experiment could not produce (and thus, of course, cannot validate). The calculation itself was a major effort, performed on the IBM system at Livermore, and was awarded the 1999 Gordon Bell Prize in the performance category. Vetter and Sturtevant sent a Mach 1.5 shock through air and into sulfur hexafluoride gas, 4.88 times denser than the air. A membrane supported by a wire mesh initially held these two gases apart. As the shock struck this membrane, it forced it through the wire mesh, rupturing it and allowing the two gases to mix. The flexing of the membrane within the square spaces in the wire mesh and the flexing of the entire wire mesh itself naturally created relatively large perturbations of the multifluid interface position on two distinct length scales. These two distinct scales are evident in the visualizations of our simulation of the experiment shown here.

The 2 images at the left in the figure on the next page show the entropy and the vorticity in a thin slice of the flow (128 cells thick) near the end of our simulation. The entropy, a near-constant of the fluid motion once the primary shock has passed, is an excellent indicator of the degree of fluid mixing near the interface. That this mixing is associated with turbulence is made clear by the visualization of the square of the vorticity (the enstrophy) in the same thin slice, which is shown at the lower left in the figure. An analysis of the velocity power spectrum within the mixing layer also reveals the classic $k^{-5/3}$ Kolmogorov trend. In the entropy image, regions of pure denser fluid appear white, while regions of pure air are transparent (hence black in this volume rendering). The many intermediate colors display the range of mixtures of the two fluids in between.

A model equation for turbulent transfer of energy to small scales: The images at the right in the figure have been obtained by applying the Gaussian filter with full width at half maximum of 68 cells to the simulation data. The filter removes the small-scale turbulence and leaves the energy containing scales of motion that come directly from the perturbation introduced by the wire mesh. The spatial variation of

F_{SGS} , defined above, is shown in the lower-right image in the figure. Note the small blue regions in this image that mark the locations of inverse energy transfer (from small to large scales). It is natural to correlate this forward energy transfer rate with the magnitude of the deviatoric symmetric strain tensor,

$$\Pi^2 = \sum_{ij} \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \nabla \cdot \tilde{\mathbf{u}} \right)^2.$$

This quantity is visualized in the middle-right image of the figure. It is large where shear is large and there is strong forward energy transfer, as we would expect, however it is also substantial where we have inverse energy transfer. To see this, look carefully at the central plume of denser gas, or at the large plume seen in cross section in the foreground of the image. The key to a correlation of F_{SGS} with Π that gets the signs right is the image at the top right in the figure, where we see the enstrophy (vorticity squared).

We see that F_{SGS} is large and positive at the top of the large dense plume, where the fluid is expanding in two dimensions and compressing in the third, while it is negative at the bottom of the principal ring vortex that forms this plume's "mushroom cap," where the flow is compressing in two dimensions and expanding in the third. Since compression and expansion correspond to oppositely signed eigenvalues of Π , and since the determinant, $\det(\Pi)$, is the product of the 3 eigenvalues, it is clear that this determinant changes sign in what appear to be the proper places. Indeed, we

found that we obtained a very strong positive correlation from the simulation data between F_{SGS} and the right-hand side of the following formula:

$$F_{SGS} \propto \mathcal{K}_f^2 \bar{\rho} (|\partial_i \tilde{u}_j|^2 \nabla \cdot \tilde{\mathbf{u}} + \frac{1}{2} \det \Pi_f)$$

where Π_f represents Π for the filtered velocity field, and λ_f is a filter wavelength, equal to 128 cells for our Gaussian filter with full width at half maximum of 68 cells. The density and velocities in this formula are those obtained by applying the Gaussian filter (the filter for the velocities is mass-weighted). The first term on the right in this expression accounts for the relatively obvious dependence of the energy transfer rate on the net fluid compression. Where the fluid is compressing as a whole, turbulent motions that are located just above the filter scale are compressed, so that they can no longer survive the application of the filter. Where the fluid is generally expanding, turbulent motions on scales just below the filter scale expand so that they now survive the application of the filter.

We believe that the insight that F_{SGS} is best correlated with $\det(\Pi)$, which has come from a detailed analysis of 3 TB of simulation data, could be the key to an effective subgrid-scale turbulence model that we could implement in our PPM gas dynamics codes. Our work to date on homogeneous turbulence and on turbulent convection indicates that such a subgrid-scale model could enable us to obtain equally accurate simulations of turbulent flow regions with grids that are 4 times coarser in each dimension. We expect that such a model could be used quite generally, although such a conjecture would need to be rigorously tested. To develop and test such a model, we intend to generate an extremely accurate data set for Mach 1 decaying homogeneous, compressible turbulence which is well resolved (at least 2048^3) in both space and time. As we have already done on a 1024^3 grid, we will stir a cube of air at constant density and pressure at time zero by introducing a random superposition of smooth sinusoidal velocity disturbances clustering in wavelength around half the width of the cube and producing a Mach 1 rms velocity distribution. Then we will compute the subsequent evolution, with periodic boundary conditions, on a very fine grid with our PPM code, saving hundreds of snap shots of the flow. We should then be able to apply the same sort of Gaussian filtering we used with the Richtmyer-Meshkov flow data to obtain from the filtered data an accurate representation of the time development of a turbulent flow that we might hope to compute on a much coarser grid using PPM with a subgrid-scale model of the small-scale turbulent motions. In addition, the full simulation data will give us the results of a small, direct simulation of the small-scale turbulence for each “cell” of the presumed coarser grid. This data can then be used to test full subgrid-scale turbulence models or parameterizations of individual quantities such as F_{SGS} . To date, we have only been able to carry out this procedure using a grid of 1024^3 cells, but in the near future, we plan to do this using a 2048^3 grid.

Checking the model for F_{SGS} with homogeneous turbulence simulation data: The data from our one-billion-cell simulation of homogeneous, compressible turbulence with our PPM code (which has greater resolving power than the sPPM code used in the Richtmyer-Meshkov simulation) allows us to generate spatial distributions of F_{SGS} using filter widths of 34 grid cells. When our run on the 2048^3 grid is completed, we will be able to increase this filter width to 68 cells, so that the turbulence on scales smaller than the filter can be well resolved. Using the coarser 1024^3 data, we have obtained the images shown on the following pages, which represent the states of this developing turbulent flow over the time period from 0.7 to 1.3 sound crossing times of the energy containing scales, when the principal transition to small-scale turbulence takes place. On the next page, we see the flow at time 1.0, in the midst of this transition, while on the following page, it is seen first earlier ($t = 0.7$, top) and then later ($t = 1.3$, bottom). It is evident from the volume rendered visualizations of the magnitude of the vorticity at the upper left in each set of images that during this time range the flow undergoes a transition from vortex sheet structures (shear layers) to vortex tube structures on the finest scales. At time 0.7 the vortex sheets are just beginning to go unstable, while at time 1.3 only masses of intertwined vortex tubes are visible. At the top center in each group of images are volume renderings of the vorticity magnitude which have been constructed using the filtered velocities. In these images the same transition is evident. We note that on these large scales our earlier convergence studies of PPM simulations of turbulence (Sytnie et al. 2000) clearly show that no direct or indirect influence of the viscosity of the PPM Euler method is felt. This is reinforced by the evidence of the velocity power spectra from this run. At the top right in each group of

images is a volume rendering of the turbulent kinetic energy, or, more properly, the kinetic energy of the flow evaluated on the fine scale grid and then averaged using the Gaussian filter.

On the bottom row of images in each group we show volume renderings of the rate of forward energy transfer, first evaluated (at the left) from its definition, then (in the center) from our model, and finally (at the right) we show the correlation between these two. These correlations are among the best we have seen in years of manipulating such data, and they are reinforced by the visual correlations evident in the volume renderings of the actual function and its model. This quantity, F_{SGS} , is a fundamental ingredient to any subgrid-scale model of turbulence. Our model allows us to predict with astonishing accuracy, using only the physical state variables on a coarse mesh, the rate at which energy is extracted from this larger-scale flow and deposited in unresolved, subgrid-scale turbulent motions. From the volume renderings it is clear that we can also predict where energy from this turbulence is transferred instead to the resolved scales of motion. Although one must always be cautious until the most careful tests can be applied using the best possible data, we believe that this discovery could possibly be a breakthrough, allowing us for the first time to predict with some degree of confidence when and where kinetic energy transfer goes from large scales to small or in the reverse direction. Our correlation of this sign of the direction of energy transfer with the flow geometry is consistent with our earlier experience simulating convection flows in horizontal, stratified layers. In those simulations, carried out at the Pittsburgh Supercomputer Center as part of our NSF Grand Challenge team project, we found that small-scale turbulence was diminished in the centers of convective upflows, where stretched vortex tubes tended to be aligned, since they were all being stretched in roughly the same direction. These vortex tubes tended to combine, as do eddies in 2-D flows. This combining of aligned and continually stretched vortex tubes produces the relative stability of bathtub vortices and tornado funnels. At the top of the upflow regions of our convection simulations, however, where the gas expanded in all directions along the upper surface of the confined layer, vortices were being stretched along multiple directions, which allowed them to interact disruptively, producing strong small-scale turbulence. This contrasting behavior can be clearly seen in

the movie animations of these earlier simulations that can be downloaded from our Web site at

www.lcse.umn.edu/MOVIES and that were published in the *Astrophysical Journal*.

The velocity power spectra shown at the top of this page indicate that by the end of our simulation we have about one whole decade of scales that are within the Kolmogorov inertial range. With a simulation on a 2048^3 grid, we will have a factor of 20 in scale from the beginning of the Kolmogorov inertial range to the near dissipation range, where the spectrum flattens slightly. This will allow us to test our model very well, since we will be able to set our filter width at multiple, well separated levels within this inertial range, each time having well resolved computed flows on scales beneath the filter width. If all is as it should be (namely, self-similar), the same formula for F_{SGS} should apply regardless of the filter width. In addition, because our data from the simulation will be well resolved in time, we will be able to observe the transport of kinetic energy across the scale of the filter without having our results contaminated by strong dissipation of the flow just below this scale. By placing our filter near the beginning of the inertial range, we will have a very accurate and detailed numerical simulation of the “subgrid-scale” motions, including their cascade in a finite time period down to the much smaller scales where they are dissipated into heat. We will be able to observe this time delay between introduction of turbulent energy on the largest “subgrid” scales and its ultimate dissipation and we should be able to extract fairly reasonable measurements of this time delay as well. In any model of subgrid-scale turbulence that involves inverse energy transfer from subgrid to resolved scales, one must build this time delay into the model, since one must be very careful not to extract energy from subgrid turbulence that has in fact already been dissipated into heat.

Thus the long series of simulations and data analysis that has been supported under this project has led us to the threshold of a subgrid-scale model for turbulence based upon our insight of the correlation between the tendency for turbulent motions to be enhanced or diminished and the local geometry of the flow. The key to this development has been the power of our PPM code, running on very large parallel systems on the finest possible grids, to perform numerical experiments that produce high quality data in such quantity and detail that we are able to test alternative conjectures both quantitatively and visually. Without the ability to handle this voluminous data, however, we could not have arrived at the insights reported above.

Analysis and Visualization of Multi-Terabyte Data Sets

It was this project that in 1986-7 provided the critical support, along with the Minnesota Supercomputer Institute, that enabled our team to build and operate our first interactive scientific visualization system. This was a duplicate of the system developed at Los Alamos by Karl-Heinz Winkler and his team. That original system allowed us to stream data from the Minnesota Supercomputer Institute’s Cray-2 supercomputer to our graphics system, where we could render the simulation data into images and animate them as movies under interactive user control. This capability was at that time unique in the

academic world. We combined it with the University of Minnesota's very large grants of Cray-2 time to embark upon the path that has led so many years later to the kind of numerical experiments and associated data analysis presented in the previous section. Although our work in this area began with a rather specialized system incorporating the flight simulator technology of the day, we very quickly executed a transition to more easily acquired, maintained, and operated workstation equipment. We accomplished this transition through an active collaboration with Silicon Graphics, Seagate, and Ciprico – companies which later became the founding industrial partners of our Laboratory for Computational Science & Engineering (LCSE). We ended up developing an entire software environment for very large scale data analysis and visualization, since no appropriately capable software was then available. We managed this effort by keeping it very narrowly focused on our own needs. Thus we developed only pixel painting techniques in the early years, when we were working exclusively in 2-D, and it was therefore natural, once we moved into 3-D, to concentrate exclusively on perspective volume rendering. With support from the University of Minnesota's Army High Performance Computing Research Center (AHPCRC), we took our visualization software and packaged a subset of it for general distribution as the volume rendering tool BoB, which was widely used for many years. With assistance from industry collaborations, we developed the PowerWall visualization system, which we demonstrated in the Silicon Graphics exhibit booth at Supercomputing 94 in Washington, D. C. PowerWalls are now in use all over the country, in one form or another, and they have been very popular in the DoE's ASCI program. More recently, with support from the NSF PACI program as well as the DoE ASCI program, we have built our scientific visualization tools into a more generally useful form and have distributed them on our lab's Web site at www.lcse.umn.edu/HVR. These latest tools are the modern descendants of those initially developed under direct support from this project. They are in active use at NCSA and Los Alamos. Our PowerWall display, a third generation system, is now housed in the University of Minnesota's new Digital Technology Center.

Summary

This project has provided the core funding to my research in scientific computation for a great many years. The products of that research are improved and enhanced numerical algorithms, improved methods for implementing those algorithms in parallel for large SMP clusters, a body of very large scale fluid dynamical simulations performed with those algorithms, and an entire software and systems environment for dealing with and visualizing the enormous amounts of data those simulations produce. Many of these developments have attracted the interest of other agencies and/or programs, so that the efforts begun with support from this project grew beyond the initial level of effort. Some of these developments, like the PowerWall visualization system, became adopted by others, both in academia, government, and industry, so that their influence was greatly expanded. PPM algorithms have on the one hand been incorporated in a number of community-oriented production codes, most notably FLASH, supported at the University of Chicago by the DoE's ASCI program, while on the other hand the techniques they embody have been adopted by others, as has occurred in the meteorological community through our team's collaborations on an NSF Grand Challenge team centered at Boulder and on an NCSA team working with the mesoscale meteorology code COMMAS. Although this is a final report, the work stimulated through this project continues as will, hopefully, our team's support from the MICS Program.

References

<http://www.lcse.umn.edu> and links from this main page.

www.lcse.umn.edu/ppmlib (library of PPM gas dynamics code modules and documentation)

www.lcse.umn.edu/hvr (LCSE Hierarchical Volume Rendering software and documentation)

1. Anderson, S. E., P. R. Woodward, and D. H. Porter 1995, "A Simplified 3-D PPM Implementation for the Clustered SMP Architecture, a Parallel Programming Template," 3-D sPPM template code, and associated materials, available at www.lcse.umn.edu/research/sppm (the LCSE Web site).
2. Bassett, G. M., and P. R. Woodward, "Numerical Simulation of Nonlinear Kink Instabilities of Supersonic Shear Layers," *Journal of Fluid Mechanics* **284**, 323-340 (Feb. 10, 1995).
3. Bassett, G. M., and P. R. Woodward, "Simulation of the Instability of Mach 2 and Mach 4 Gaseous Jets in 2 and 3 Dimensions," *Astrophysical Journal* **441**, (March 10, 1995).
4. Boris, J. P., F. F. Grinstein, E. S. Oran, and R. J. Kolbe, "New insights into large eddy simulation," *Fluid Dyn. Res.* **19**, 19 (1992).
5. Chen, S., D. D. Holm, L. G. Margolin, and R. Zhang, "Direct numerical simulations of the Navier-Stokes alpha model," *Physica D* **133**, 66-83 (1999).
6. Cohen, R. H., W. P. Dannevik, A. M. Dimits, D. E. Eliason, A. A. Mirin, Y. Zhou, D. H. Porter, and P. R. Woodward, "Three-Dimensional Simulation of a Richtmyer-Meshkov Instability with a Two-Scale Initial Perturbation," *Physics of Fluids*, **14**, 3692-3709 (2002); also available as LLNL Preprint UCRL-JC-144836 (2000).
7. Cohen, R. H., W. P. Dannevik, A. M. Dimits, D. E. Eliason, A. A. Mirin, D. H. Porter, O. Schilling, and P. R. Woodward, "Three-dimensional high-resolution simulations of the Richtmyer-Meshkov mixing and shock-turbulence interaction," in Proc. Of the 6th International Workshop on the Physics of Compressible Turbulent Mixing, Marseille, France, 18-21 June, 1997, pp. 128-133. LLNL Report UCRL-JC-125309.
8. Cohen, R. H., A. M. Dimits, A. A. Mirin, W. P. Dannevik, R. G. Eastman, D. E. Eliason, O. Schilling, D. H. Porter and P. R. Woodward, "Three-Dimensional PPM Simulations of Re-shocked Richtmyer-Meshkov Instability," VHS Video (1997), LLNL Report UCRL-MI-128783.
9. Colella, P., and P. R. Woodward, "The Piecewise-Parabolic Method (PPM) for Gas Dynamical Simulations," *J. Comput. Phys.* **54**, 174-201 (1984).
10. Dai, Wenlong, and P. R. Woodward, "A High-Order Iterative Implicit-Explicit Hybrid Scheme for 2-D Hydrodynamics," in *Numerical Methods for Fluid Dynamics V*, pp. 377-383, eds. K. W. Morton and M. J. Baines, Oxford Science Publications (1995).
11. Dai, Wenlong, and P. R. Woodward, "A High-Order Godunov Scheme for Ideal Magnetohydrodynamics Equations Based on a Nonlinear Riemann Solver," in *Proceedings of the Fifth International Conf. On Hyperbolic Problems: Theory, Numerics, Applications*, pp. 293-299, eds. J. Glimm, M. J. Graham, J. W. Grove, and B. J. Plohr, World Scientific (1996).
12. Dai, Wenlong, and P. R. Woodward, "Iterative Implementation of an Implicit-Explicit Hybrid Scheme for Hydrodynamics," *Journal of Computational Physics* **124**, 217-229 (1996).
13. Dai, Wenlong, and P. R. Woodward, "A Second-Order Iterative Implicit-Explicit Hybrid Scheme for Hyperbolic Systems of Conservation Laws," *Journal of Computational Physics* **128**, 181-196 (1996).
14. Dai, Wenlong, and P. R. Woodward, "On the Divergence-Free Condition and Conservation Laws in Numerical Simulations for Supersonic Magnetohydrodynamical Flows," *Astrophysical J.*, **493** (1998).
15. Dai, Wenlong, and P. R. Woodward, "Numerical Simulations for Nonlinear Heat Transfer in a System of Multimaterials," *Journal of Computational Physics* **139**, 58-78 (1998).
16. Dai, Wenlong, and P. R. Woodward, "Numerical Simulations for Radiation Hydrodynamics. I. Diffusion Limit," *Journal of Computational Physics* **141**, 182-207 (1998).

17. Dai, Wenlong, and P. R. Woodward, "A Simple Finite Difference Scheme for Multidimensional Magnetohydrodynamical Equations," *Journal of Computational Physics* **142**, 331-369 (1998).
18. Dai, Wenlong, and P. R. Woodward, "Numerical Simulations for Radiation Hydrodynamics II. Transport Limit," *Journal of Computational Physics* **157**, 199-233 (2000).
19. Dai, Wenlong, and P. R. Woodward, "Implicit-Explicit Hybrid Schemes for Radiation Hydrodynamics Suitable for Distributed Computer Systems," in *Parallel Computational Fluid Dynamics*, Elsevier, Eds. Keyes, Ecer, Satofuka, Fox, and Periaux, p. 179-187 (2000).
20. Dannevik, W. P., A. M. Dimits, R. H. Cohen, D. Eliason, O. Schilling, D. H. Porter and P. R. Woodward, "Three-Dimensional Compressible Rayleigh-Taylor Simulation on the ASCI Blue-Pacific ID System," VHS Video (1996), LLNL Report UCRL-MI-125449.
21. Dinge, D., and P. R. Woodward, "A Parallel Cell-by-Cell AMR Method for the PPM Hydrodynamics Code," submitted to *J. Computational Phys.* (see also www.lcse.umn.edu/research/micsstuff).
22. Edgar, B. K., Woodward, P. R., and Anderson, S. E., PPM code library, with documentation and examples, available at www.lcse.umn.edu/PPMlib.
23. Fryxell, B., P. R. Woodward, P. Colella, and K.-H. Winkler, "An Implicit-Explicit Extension of the PPM Scheme for Lagrangian Hydrodynamics in One Dimension," *J. Comput. Phys.* **63**, 283 (1986).
24. Habermann, J., and P. R. Woodward 1995, "A Simplified 2-D PPM Implementation for the Clustered SMP Architecture, a Parallel Programming Template," 2-D sPPM template code, and associated materials, available at www.lcse.umn.edu/2Dsppm (the LCSE Web site).
25. Hayes, P. R., M. T. O'Keefe, P. R. Woodward, and A. Gopinath, "High-order-compact time-domain numerical simulation of optical waveguides," *Optical and Quantum Electronics* **31**: 813-826 (1999).
26. Karamanos, G.-S., and Karniadakis, G. E., "A Spectral Vanishing Viscosity Method for Large-Eddy Simulations," *Journal of Computational Physics* **163**, 22-50 (2000).
27. Mirin, A. A., R. H. Cohen, W. P. Dannevik, A. M. Dimits, D. E. Eliason, D. H. Porter, O. Schilling and P. R. Woodward, "Three-Dimensional Simulations of Compressible Turbulence on High-Performance Computing Systems," Eighth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis (1997), LLNL Report UCRL-JC-125949.
28. Mirin, A. A., R. H. Cohen, B. C. Curtis, W. P. Dannevik, A. M. Dimits, M. A. Duchaineau, D. E. Eliason, D. R. Schikore, S. E. Anderson, D. H. Porter, P. R. Woodward, L. J. Shieh, and S. W. White, "Very High Resolution Simulation of Compressible Turbulence on the IBM-SP System," the IEEE's 1999 Gordon Bell Award was conferred in the performance category for this work; also available as Lawrence Livermore National Laboratory Report UCRL-MI-134237, July, 1999.
29. Ofelt, D., D. Porter, T. Varghese, R. Schmid, T. Ruwart, and P. R. Woodward, "PPM Graphics Tools," distributed as a Minnesota Supercomputer Institute Report, 1989.
30. Pedelty, J. A., and P. R. Woodward, "Numerical Simulations of the Nonlinear Kink Modes in Linearly Stable Supersonic Slip Surfaces," *J. Fluid Mech.*, **225**, 101-120 (1991).
31. Porter, D. H. 2001. Early release of LCSE's HVR hierarchical volume rendering software is available at www.lcse.umn.edu/hvr.
32. Porter, D. H., A. Pouquet, I. Sytine, and P. R. Woodward, "Turbulence in Compressible Flows," *Physica A* **263**, 263-270 (1999).

33. Porter, D. H., A. Pouquet, and P. R. Woodward, "Supersonic Homogeneous Turbulence," *Lecture Notes in Physics*, **392**, 105-125 (1991).
34. Porter, D. H., A. Pouquet, and P. R. Woodward, "A Numerical Study of Supersonic Turbulence," *Theoretical and Computational Fluid Dynamics*, **4**, 13-49 (1992).
35. Porter, D. H., A. Pouquet, and P. R. Woodward, "Three-Dimensional Supersonic Homogeneous Turbulence: A Numerical Study," *Phys. Rev. Lett.*, **68**, 3156-3159 (1992).
36. Porter, D. H., A. Pouquet, and P. R. Woodward, "Kolmogorov-Like Spectra in Decaying Three-Dimensional Supersonic Flows," *Phys. Fluids A*, **6**, 2133-2142 (1994).
37. Porter, D. H., A. Pouquet, and P. R. Woodward, "Compressible Flows and Vortex Stretching," in *Small-Scale Structures in Three-Dimensional Hydrodynamic and MHD Turbulence*, ed. A. Pouquet, M. Meniguzi, & P-L. Sulem, Springer Verlag, p. 51-58 (1995).
38. Porter, D. H., A. Pouquet, and P. R. Woodward, "Intermittency in Compressible Flows," Proc. Seventh European Turbulence Conference, St. Jean C. F., France, July, 1998, ed. Uriel Frisch.
39. Porter, D. H., Pouquet, A., and Woodward, P. R., "Measures of Intermittency in Driven Supersonic Flows," *Physical Review E* **66**, 026301 (2002).
40. Porter, D. H., and P. R., Woodward, "3-D Simulations of Turbulent Compressible Convection," *Astrophysical Journal Suppl.* **127**, 159-187 (2000); available at www.lcse.umn.edu/conv3d.
41. Porter, D. H., P. R. Woodward, and M. L. Jacobs, "Convection in Slab and Spheroidal Geometries," Proc. 14th International Florida Workshop in Nonlinear Astronomy and Physics: Astrophysical Turbulence and Convection, Univ. of Florida, Feb., 1999; in *Annals of the New York Academy of Sciences* **898**, 1-20 (2000); available at www.lcse.umn.edu/convsph.
42. Porter, D. H., P. R. Woodward, and A. Pouquet, "Inertial Range Structures in Decaying Turbulent Flows," *Physics of Fluids* **10**, 237-245 (1998).
43. Sawdey, A., D. Lee, T. Ruwart, P. R. Woodward, M. O'Keefe, and R. Bleck, "Interactive Smooth-Motion Animation of High Resolution Ocean Circulation Calculations," Proceedings of the ICASE Workshop on Scientific Visualization, Hampton, VA, Sept., 1995.
44. Schilling, O., R. H. Cohen, W. P. Dannevik, A. M. Dimits, D. E. Eliason, A. A. Mirin, D. H. Porter and P. R. Woodward, "Three-Dimensional High-Resolution Simulations of Compressible Rayleigh-Taylor Instability and Turbulent Mixing," Sixth Int'l. Workshop on the Physics of Compressible Turbulent Mixing, Marseille, France (1997), LLNL Report UCRL-JC-125308.
45. Silicon Graphics and the University of Minnesota, "Grand Challenge Computing, A No-Frills Technical Documentary," a video documentary produced by Silicon Graphics in October, 1993, describing my group's collaboration with them in Sept., 1993.
46. Stevens, R., Woodward, P. R., DeFanti, T., and Catlett, C., "From the I-Way to the National Technology Grid," *Communications of the ACM* **40**, 51-60 (Nov., 1997).
47. Sytine, I. V., D. H. Porter, P. R. Woodward, S. W. Hodson, and K.-H. Winkler 1998, "Convergence Tests for Piecewise Parabolic Method and Navier-Stokes Solutions for Homogeneous Compressible Turbulence," submitted to *J. Comput. Phys.*, also UMSI 98/169 Oct 1998.
48. Vetter, M., and Sturtevant, B., 1995. "Experiments on the Richtmyer-Meshkov Instability of an Air/SF6 Interface," *Shock Waves*, **4**, 247-252.
49. Winkler, K.-H., J. W. Chalmers, S. W. Hodson, P. R. Woodward, and N. J. Zabusky, "A Numerical Laboratory," *Physics Today*, Oct., 1987.

50. Woodward, P. R., "Numerical Methods for Astrophysicists," in *Astrophysical Radiation Hydrodynamics*, eds. K.-H. Winkler and M. L. Norman, Reidel, 1986, pp. 245-326.
51. Woodward, P. R., "Interactive Scientific Visualization of Fluid Flow," *IEEE Computer*, Vol. **26**, No. 10, pp. 13-25 (October, 1993).
52. Woodward, P. R., "Superfine Grids for Turbulent Flows," *IEEE Computational Science & Engineering*, Vol. **1**, No. 4, pp. 4-5+cover illustration, (December, 1994).
53. Woodward, P. R., "Perspectives on Supercomputing: Three Decades of Change," *IEEE Computer*, **29**, 99-111 (October, 1996); available at www.lcse.umn.edu/computer .
54. Woodward, P. R., and S. E. Anderson, "Portable Petaflop/s Programming: Applying Distributed Computing Methodology to the Grid Within a Single Machine Room," Proc. of the 8th IEEE International Conference on High Performance Distributed Computing, Redondo Beach, Calif., Aug., 1999; available at www.lcse.umn.edu/HPDC8.
55. Woodward, P. R., and P. Colella, "The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks," *J. Comput. Phys.* **54**, 115-173 (1984).
56. Woodward, P. R., D. H. Porter, B. K. Edgar, S. E. Anderson, and G. Bassett, "Parallel Computation of Turbulent Fluid Flow," Proc. Parallel CFD '94 Conference, Kyoto, Japan, May, 1994; published in *Computing Applications Mathematics*, Vol. **14**, no. 1, pp 97-105 (1995).
57. Woodward, P. R., D. H. Porter, B. K. Edgar, W. Dai, G. Bassett, S. E. Anderson, "Application of Monotonicity-Preserving Godunov Schemes to Compressible Flow Simulation," Proceedings of the Godunov Conference, Univ. of Michigan, Ann Arbor, May, 1997.
58. Woodward, P. R., Porter, D. H., and Jacobs, M., "3-D Simulations of Turbulent, Compressible Stellar Convection," Proc. 3-D Stellar Evolution Workshop, Univ. of Calif. Davis I.G.P.P., July, 2002; also available at www.lcse.umn.edu/3Dstars.
59. Woodward, P. R., D. H. Porter, I. Sytine, S. E. Anderson, A. A. Mirin, B. C. Curtis, R. H. Cohen, W. P. Dannevik, A. M. Dimits, D. E. Eliason, K.-H. Winkler, and S. W. Hodson, "Very High Resolution Simulations of Compressible, Turbulent Flows," in *Computational Fluid Dynamics*, Proc. of the 4th UNAM Supercomputing Conference, Mexico City, June, 2000, edited by E. Ramos, G. Cisneros, R. Fernández-Flores, A. Santillan-González, World Scientific (2001); available at www.lcse.umn.edu/mexico.
60. Yokokawa, M., Itakura, K., Uno, A., Ishihara, T., and Kaneda, Y., "16.4 Tflops Direct Numerical Simulation of Turbulence by a Fourier Spectral Method on the Earth Simulator," Proc. Supercomputing 2002, Baltimore, Nov., 2002.
61. LCSE movies from a variety of turbulent fluid flow simulations can be downloaded and/or viewed at www.lcse.umn.edu/MOVIES.