INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2002; **40**:209–216 (DOI: 10.1002/fld.310)

A parallel adaptive projection method for low Mach number flows

J. B. Bell*, M. S. Day, A. S. Almgren, M. J. Lijewski and C. A. Rendleman

MS 50A-1148, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, U.S.A.

SUMMARY

We describe an adaptive projection method for numerically simulating low Mach number flows. The projection method formulation enforces the velocity divergence constraint resulting from the low Mach number approximation. It is implemented on an adaptive hierarchy of logically rectangular grids, where each finer level is refined in space and in time. The adaptive algorithm has been shown in previous papers to be robust and second-order accurate, and to satisfy the principles of conservation and free-stream preservation as applicable. Here, the parallelization is described in some detail, and the methodology is demonstrated on two examples from premixed, low Mach number combustion. Published in 2002 by John Wiley & Sons, Ltd.

1. INTRODUCTION

In this paper, we describe an adaptive projection method for numerically simulating low Mach number flows. Low Mach number approximations for different flow phenomena can be derived from asymptotic expansions of the compressible flow equations in Mach number (see, e.g. Reference [1]). The incompressible Navier–Stokes equations represent the simplest low Mach number approximation. By incorporating additional physics one can derive low Mach number models for multiphase flow, combustion and atmospheric flows. In the latter two cases, the low Mach number approximation includes bulk compressibility effects but analytically filters out acoustic waves. Typical equation sets governing low Mach number flows consist of conservation equations for mass and momentum, evolution equations for auxiliary quantities, and a constraint on the velocity field. Thus, we consider systems of the form

$$\frac{\partial(\rho U)}{\partial t} + \nabla \cdot (\rho U U) = -\nabla\pi + \nabla \cdot \tau + F \tag{1}$$

*Correspondence to: J. B. Bell, MS 50A-1148, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, U.S.A.

Contract/grant sponsor: Applied Mathematical Sciences Program of the DOE Office of Mathematics, Information, and Computational Sciences; contract/grant number: DE-AC03-76SF00098

Received May 2001
Revised September 2001

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U) = 0 \tag{2}$$

$$\frac{\partial(\rho \mathbf{\Psi}_i)}{\partial t} + \nabla \cdot (U \rho \mathbf{\Psi}_i) = S_{\mathbf{\Psi}_i} \tag{3}$$

$$\nabla \cdot (\alpha U) = S_U \tag{4}$$

Here ρ is the density, U is the velocity, π is the perturbational variation from ambient pressure, τ is the stress tensor, F represents external forces and $\mathbf{\Psi}_i$ represents auxiliary variables such as species mass fractions, moisture content or energy.

The right-hand side of (3), $S_{\mathbf{\Psi}_i}$, includes effects such as diffusion, chemical reactions, and other sinks or sources. The coefficient, α, in (4) may be identically one in the case of incompressible flow or low Mach number combustion, or it may represent ambient density stratification in the case of the atmosphere. S_U may be identically zero, as for incompressible flow, or contain terms representing bulk compressibility effects such as expansion from heat release during combustion.

In the next two sections we briefly describe an adaptive numerical algorithm for solving systems of this type using a projection formulation. Details of this approach for incompressible flow are discussed in Almgren *et al.* [2]. Sussman *et al.* [3] extend the method to multiphase flow; Day and Bell [4] describe the methodology for low Mach number combustion with complex chemistry; and Almgren *et al.* [5] discuss extension to anelastic atmospheric modelling. In Section 4 we discuss the implementation of this methodology on distributed memory parallel architectures. Section 5 contains results from two calculations, which illustrate the methodology applied to low Mach number combustion models with comprehensive kinetics.

2. PROJECTION DISCRETIZATION

Our overall computational approach uses a staggered grid spatial discretization with cell-centred values for U, ρ and $\mathbf{\Psi}_i$ and node-centred values for π. The values of π are also staggered in time. The temporal discretization is a fractional step scheme based on a projection formulation that accommodates large density contrasts. Below we sketch the basic discretization scheme for a single grid. In the next section we discuss incorporating this algorithm into an adaptive mesh framework.

The evolution equations, (1)–(3), are discretized using a time-explicit second-order Godunov scheme for the advective terms with appropriate second-order treatments of other physics (e.g. a Crank–Nicolson discretization of the diffusive terms). The Godunov algorithm for computing advective derivatives requires a time-centred, edge-based advection velocity, U^{ADV}, that satisfies constraint (4). To obtain U^{ADV} we first construct a provisional time-centred approximation to the normal velocity at cell edges, $U^{\text{ADV},*}$, using the cell-centred data at t^n and the lagged pressure gradient, $\pi^{n-1/2}$. In general $U^{\text{ADV},*}$ fails to satisfy the divergence constraint at $t^{n+1/2}$. We apply a discrete projection by solving the elliptic equation

$$D^{\mathrm{MAC}}\left(\frac{\alpha}{\rho^n}G^{\mathrm{MAC}}\phi^{\mathrm{MAC}}\right)=D^{\mathrm{MAC}}(\alpha U^{\mathrm{ADV},*})-\left(S_U^n+\frac{\Delta t^n}{2}\frac{S_U^n-S_U^{n-1}}{\Delta t^{n-1}}\right) \tag{5}$$

for ϕ^{MAC}, where D^{MAC} represents a centred approximation to a cell-based divergence from edge-based velocities, and G^{MAC} represents a centred approximation to edge-based gradients from cell-centred data. The second term on the right-hand side of (5) represents a time-extrapolated estimate of $S_U^{n+1/2}$. The solution, ϕ^{MAC}, is used to define

$$U^{\mathrm{ADV}}=U^{\mathrm{ADV},*}-\frac{1}{\rho^n}G^{\mathrm{MAC}}\phi^{\mathrm{MAC}}$$

We use U^{ADV} to compute the advective derivatives in (1)–(3) using an explicit second-order Godunov discretization. At this point the density can be updated by

$$\rho^{n+1}=\rho^n+\Delta t[\nabla\cdot(\rho U^{\mathrm{ADV}})]^{n+1/2}$$

Updating the $\mathbf{\Psi}_i$ requires additionally a suitably accurate evaluation of $S_{\mathbf{\Psi}_i}$. This step may be trivial if $S_{\mathbf{\Psi}_i}=0$, it may simply involve an implicit treatment of diffusion in a Crank–Nicolson step, or it may be quite complex. For low Mach number combustion this step involves a specialized operator-split treatment of complex chemistry using a stiff ODE integration package and non-linear diffusion terms. In an analogous manner we next compute an intermediate velocity field, $U^{n+1,*}$, using the lagged pressure gradient, by solving

$$\begin{aligned}\rho^{n+1/2}\left(\frac{U^{n+1,*}-U^n}{\Delta t}+[(U^{\mathrm{ADV}}\cdot\nabla)U]^{n+1/2}\right)\\ =\frac{1}{2}(\nabla\cdot\tau^n+\nabla\cdot\tau^{n+1,*})-\nabla\pi^{n-1/2}+\frac{1}{2}(F^n+F^{n+1})\end{aligned}$$

where $\tau^{n+1,*}=\mu^{n+1}((\nabla+\nabla^{\mathrm{T}})U^{n+1,*}-2/3\delta_{ij}S_U^{n+1}), S_U^{n+1}$ is evaluated with the newly computed ρ^{n+1} and $\mathbf{\Psi}^{n+1}$, and $\rho^{n+1/2}=\frac{1}{2}(\rho^n+\rho^{n+1})$.

The intermediate velocity field, $U^{n+1,*}$, does not, in general, satisfy the constraint at t^{n+1}. We apply an approximate projection to update the perturbational pressure and to project $U^{n+1,*}$ onto the constraint surface. In particular, we solve

$$\begin{aligned}L^\rho\pi^{n+1/2}&=D\left(\alpha\left(U^{n+1,*}+\frac{\Delta t}{\rho^{n+1/2}}G\pi^{n-1/2}\right)\right)-S_U^{n+1}\\ U^{n+1}&=U^{n+1,*}-\frac{1}{\rho^{n+1/2}}G(\pi^{n+1/2}-\pi^{n-1/2})\end{aligned} \tag{6}$$

for nodal values of $\pi^{n+1/2}$, where L^ρ is a standard bilinear finite element approximation to $\nabla\cdot(\alpha/\rho)\nabla$ with ρ evaluated at $t^{n+1/2}$. In this step, D is a discrete second-order operator that approximates the divergence at nodes from cell-centred data, and $G=-D^{\mathrm{T}}$ approximates a cell-centred gradient from nodal data.

The explicit procedure for the treatment of advection terms necessitates a CFL-type time-step restriction based on the advection velocity; however, this restriction is much larger than the time scale associated with the acoustic waves.

3. ADAPTIVE MESH REFINEMENT (AMR)

AMR is based on a sequence of nested logically rectangular grids with successively finer spacing in both time and space. In our approach, fine grids are formed by dividing coarse cells by a refinement ratio, r, in each direction. Typically in our applications, r is 2 or 4. Increasingly finer grids are recursively embedded in coarse grids until the solution is adequately resolved with each level contained in the next coarser level. An error estimation procedure based on user-specified criteria evaluates where additional refinement is needed and grid generation procedures dynamically create or remove rectangular fine grid patches as resolution requirements change.

The adaptive time-stepping algorithm advances the data at different levels using time steps appropriate to that level based on CFL considerations. The time-step procedure can most easily be thought of as a recursive algorithm, in which we first advance level l $(0 \leqslant l \leqslant l_{\max})$ as if it were the only level, supplying boundary conditions from level $l-1$ (if level $l > 0$), and from the physical domain boundaries. If $l < l_{\max}$, we then advance level $(l+1)$ r times with time step $\Delta t^{l+1} = (1/r)\Delta t^{l}$, using boundary conditions supplied from level l and from the physical domain boundaries. Finally, we synchronize the data between levels l and $l+1$, and interpolate corrections to higher levels if $l+1 < l_{\max}$.

The 'advance' part of the adaptive algorithm, as outlined above, solves the evolution and constraint equations on the union of grids at each level independently of all coarser or finer levels, with the exception of Dirichlet boundary data supplied from the next coarser level. While this separation of levels is necessary for subcycling in time, it creates a discrepancy between the data on fine grids and the data on the coarse grids underlying them; it also creates flux mismatches at coarse/fine boundaries that result in errors in the overall solution. The first discrepancy is easily solved with an averaging procedure, in which conserved quantities on the fine grids are conservatively averaged onto the coarser level. For the flux mismatches, the nature of each equation solved determines not only the type of flux mismatch, but also the nature of the correction equation that must be solved to synchronize the data between levels. For the explicit advective terms, for example, the edge-based fluxes as calculated on the coarse/fine boundary differ between levels; the synchronization step is composed entirely of an explicit flux correction at the boundary. For the parts of the algorithm that require the solution of parabolic and elliptic equations, however, the synchronization also requires solution of parabolic and elliptic equations, respectively. The flux mismatch in each case is again localized on the coarse/fine boundary, but the correction itself must be distributed throughout the domain. In addition, corrections applied in earlier steps of the synchronization procedure must be correctly accounted for in the source terms for later steps of the synchronization.

Of the synchronization steps that require the solution of a parabolic or elliptic equation, all but one are computed on the coarser of the two levels containing the mismatch. For reasons of accuracy, the nodal 'synchronization projection', which enforces the divergence constraint on the new-time multilevel velocity field, is a two-level solution procedure. The difficulties this causes in the parallelization of the overall algorithm are discussed in the next section.

4. PARALLEL IMPLEMENTATION

The adaptive methodology is embodied in a hybrid C++/FORTRAN software system. In this framework, memory management and control flow are expressed in the C++ portions of the program and the numerically intensive portions of the computation are handled in FORTRAN. The software is written using a layered approach, with a foundation library, BoxLib, that is responsible for the basic algorithm domain abstractions at the bottom, and a framework library, AMRLib, that marshalls the components of the AMR algorithm, at the top. Support libraries built on BoxLib are used as necessary to implement application components such as interpolation of data between levels, the coarse/fine interface synchronization routines, and linear solvers used in the projections and diffusion solvers.

The fundamental parallel abstraction is the MultiFab, which encapsulates the FORTRAN-compatible data defined on unions of Boxs; a MultiFab can be used as if it were an array of FORTRAN-compatible grids. The grids that make up the MultiFab are distributed among the processors, with the implementation assigning grids to processors using the distribution given by the load balance scheme described in Crutchfield [6] and in Rendleman *et al.* [7]. This load balance scheme is based on a dynamic programming approach for solving the knapsack problem: the computational work in the irregularly sized grids of the AMR data structures is equalized among the available processors. (For non-reacting flows, the number of cells per grid is often a good work estimate; for flows involving additional physics, such as chemical kinetics, the amount of work per cell is often highly variable, and more complicated work estimates are needed for good parallel performance.) Non-MultiFab operations and data structures are replicated across all of the processors. This non-parallel work is usually measured to be small. Because each processor possesses the global data layout, the processor can post data send and receive requests without a prior query for data size and location.

MultiFab operations are performed in one of the three ways, depending on the implicit communications pattern. In the simplest case, such as evaluation of thermodynamic properties, there is no interprocessor communication; the calculation is parallelized trivially with an *owner computes* rule with each processor operating independently on its local data. Different parallel constructs are necessary when data communication involves more than one MultiFab, an example of which is the *fill patch* operation, which interpolates coarse cell data onto overlying fine grid patches. Such constructs cannot be implemented by simply nesting loops because outer loop bodies for sub-grids that are off processor will not be executed. They must be implemented by our second method using two stages: data is exchanged between processors and then the local targets are updated.

The third, more complicated case, arises in parallelizing loops in the multilevel ‘synchronization projection’. In three dimensions, coarse/fine interfaces may be faces, edges or corners; as a result the construction of discretization stencils is done in a generalized assembly procedure, in which loops must be applied in a specific order with possible coupling from loop body to loop body. In addition, the boundary patch-filling operation copies in stages, with data from initial stages contributing to data at later stages. These order dependencies in operator evaluation give rise to what may be called *weakly sequential loops*.

Weakly sequential loops can be characterized using the language of graph theory. Loop bodies correspond to nodes in the graph and dependencies in the iterations of the loop body correspond to directed edges in the graph. Naive implementation of the construction of the dependency graph results in an operation count of $O(N^2)$, where N is the number of loop

Table I. Performance data for vortex flame interaction.

CPUs	Base grid	Cells advanced ($\times 10^6$)	Time (s)	Time per step (s)
4	48×160	5.7	1 06 676	711
16	96×320	32.7	2 04 418	681
64	192×640	216.3	5 04 021	840

elements, usually proportional to the number of grids at a level. This could be significant when there are thousands of elements in the loop, a situation that is not uncommon. However, careful implementation reduces computation cost to $O(N/P)^2$, which exhibits lower growth because the number of processors, P, used in a calculation is an increasing function of the number of grids. The careful implementation removes tasks from the task loop as they are added if the task does not use data local to that processor, or if it uses data only local to that processor and the task does not depend on prior tasks in the task list.

The principle disadvantage of the task list approach is that it encourages an unnatural coding style: a helper class must be implemented for each loop in the program. For the projection, which consists of approximately 13 000 lines of C++ (and 16 000 lines of `FORTRAN`), fewer than a dozen helper classes are needed.

5. RESULTS AND CONCLUSIONS

In this section we demonstrate the parallel adaptive methodology described above applied to premixed, low Mach number combustion. The first example illustrates the interaction of a vortex with a lean, premixed methane flame. The methane chemistry in this example uses GRI-Mech 3.0 [8] with nitrogen chemistry which includes 65 species and 447 reactions and a mixture model for preferential diffusion. For this example, we integrate a $1.2 \times 4.0\ \mathrm{cm}^2$ domain to a fixed time using three, successively finer base grids, each with two levels of factor-of-two refinement. In Plate 1 we present a snapshot of the solution for the finest base grid along with a blowup showing the location of grids around the flame. To assess the parallel performance, we have scaled the number of processors for each case by the number of cells in the base grid. The computations were performed on an IBM SP/Power 3. Statistics summarizing the runs are given in Table I.

For these runs a work estimate for integrating the chemical kinetics is used to balance the work load for the chemistry. A reasonable measure of parallel performance is given by the time per complete coarse time step which includes all subcycled steps on refined patches as well as necessary synchronization. Since the number of cells in the base grid per processor remains constant, ideal parallel scaling would produce a constant value per time step. The reduction in the time per step at the medium resolution reflects efficiency gains resulting from adaptive refinement. The increase in the time per step for the finest run reflects, in part, not having a sufficient number of grids to distribute to the processors to achieve an efficient load balance. A uniform grid computation on a 192×640 mesh that yields the same resolution as the coarsest adaptive case advances 73.7 million cells and required 528467 s on four processors. Thus, for this case the use of adaptive refinement reduced the computational cost by a factor of five. We note that the computational time per cell is larger in the adaptive

Table II. Wall-clock execution times for the (a) methane diffusion flame, and (b) premixed hydrogen flame examples. Here N is the number of processors used.

Three-dimensional premixed H-flame	
CPUs	CPU time (s)
8	4310
12	3205
16	2634
24	2093

computations than in the uniform computation. This does not reflect poor performance of the adaptive code. This increase arises because the adaptive code focuses computational work in regions of the flame where the chemistry integration is substantially more time consuming.

In the second example, we compute the time-dependent response of a premixed hydrogen flame to decaying turbulence in the fuel stream. This investigation, which follows research by Zhang and Rutland [9], includes detailed transport and hydrogen reaction chemistry. The objective of these computations is to predict turbulent flame speed and to study the modulation of flame chemistry in a turbulent flame interaction.

The computational domain is a doubly periodic three-dimensional box, $5 \times 5 \times 10\ \text{mm}^3$. Turbulent flow enters the bottom of the box, passes through the flame, and exits the top. Plate 2 shows the wrinkled flame surface late in the computation, and features the vorticity magnitude in a slice plane through the data. Two levels of factor-of-two refinement were placed over the base $32 \times 32 \times 64$ grid, for an effective resolution of $\Delta x = 39\ \mu\text{m}$ near the flame. The first level was localized near regions of high fluid vorticity, and the second level additionally refined regions of high HO_2 concentration, which exists only at the flame surface.

The computational expense of this computation precludes a detailed study of the type performed for the two-dimensional example above. To generate timings presented in Table II, the calculation was restarted at the time depicted in the figure and run for a single coarse-grid time step (corresponding to two time steps at the intermediate level and four time steps at the finest level, as well as the necessary synchronizations). The problem required too much memory to be run on four processors, and for more than 24 processors there were too few grids for the knapsack algorithm to effectively distribute a balanced workload. The data for 8, 12, 16 and 24 processors of an IBM SP/Power 3 shows a parallel efficiency of approximately 70–80% using four CPUs per node.

In summary, we have presented an adaptive projection method for numerically simulating low Mach number flows. The adaptive algorithm has been shown in previous papers to be robust and second-order accurate, and to satisfy the principles of conservation and free-stream preservation as applicable. Here the parallelization is described in some detail, and two computational examples of low Mach number combustion are given, demonstrating the capabilities of the methodology. Future work includes improvements to the parallel performance, exploration of higher order discretizations, and investigation of error estimation procedures to most effectively utilize the adaptive capability to improve the overall solution accuracy. In the combustion area, the objective is to apply the methodology to model combustion efficiency and pollutant formation in turbulent flames.

REFERENCES

1. Majda A, Sethian JA. Derivation and numerical solution of the equations of low Mach number combustion. *Combinatorial Science and Technology* 1985; **42**:185–205.
2. Almgren AS, Bell JB, Colella P, Howell LH, Welcome M. A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. *Journal of Computational Physics* 1998; **142**:1–46.
3. Sussman M, Almgren AS, Bell JB, Howell LH, Welcome ML. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics* 1999; **148**(1):81–124.
4. Day MS, Bell JB. Numerical simulation of laminar reacting flows with complex chemistry. *Combustion Theory Modelling* 2000; **4**:535–556.
5. Almgren AS, Bell JB, Colella P, Howell LH, Welcome M. A high-resolution adaptive projection method for regional atmospheric modeling. In *Proceedings of the U.S. EPA NGEMCOM Conference*, August 1995.
6. Crutchfield WY. Load balancing irregular algorithms. *Technical Report UCRL-JC-107679*, Lawrence Livermore National Laboratory, July 1991.
7. Rendleman CA, Beckner VE, Lijewski ML, Crutchfield WY, Bell JB. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science* 2000; **3**(3):147–157.
8. Smith GP, Golden DM, Frenklach M, Moriarty NW, Eiteneer B, Goldenberg M, Bowman CT, Hanson RK, Song S, Gardiner, Jr WC, Lissianski VV, Qin Z. http://www.me.berkeley.edu/gri_mech.
9. Zhang S, Rutland CJ. Premixed flame effects on turbulence and pressure-related terms. *Combustion and Flame* 1995; **102**:447–461.
10. Sullivan N, Jensen A, Glarborg P, Day MS, Bell JB, Grcar JB, Pope C, Kee RJ. Ammonia conversion in laminar coflowing nonpremixed methane–air flames. *Poster presented at the 2001 Joint Meeting of the US Section of the Combustion Institute*, Oakland, CA, March 25–28, 2001.

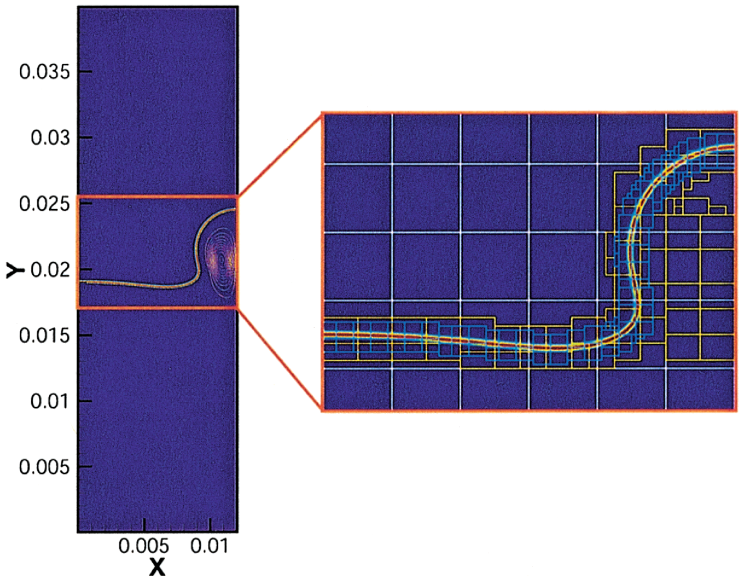

Plate 1. Vortex flame interaction for a lean premixed methane flame. The left frame shows mole fraction of CH with superimposed vorticity contours. The right frame is a blowup of the region around the flame showing the location of grid patches.

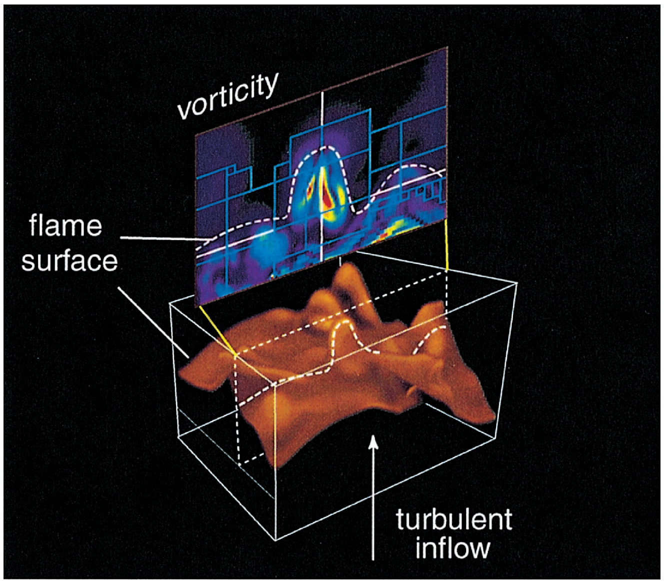

Plate 2. Premixed hydrogen flame sheet perturbed by a turbulent reactant stream. The location of the flame is indicated by a volume rendering of the HO_2 field. The removed two-dimensional slice depicts vorticity magnitude and the refined grids near the flame surface.