Terascale Spectral Element Algorithms and Implementations

H. M. Tufo[1] and P. F. Fischer[2]

August 6, 1999

Abstract

We describe the development and implementation of an efficient spectral element code for multimillion gridpoint simulations of incompressible flows in general two- and three-dimensional domains. Key to this effort has been the development of scalable solvers for elliptic problems and a stabilization scheme that admits full use of the method's high-order accuracy. We review these and other recently developed algorithmic underpinnings that have resulted in good parallel and vector performance on a broad range of architectures and that, with sustained performance of 319 GFLOPS on 2048 nodes of the Intel ASCI-Red machine at Sandia, readies us for the multithousand node terascale computing systems now coming on line at the DOE labs.

1 Introduction

One of the primary motivations driving high-performance computing is to augment scientific experiments as a means of investigation. To this end, we are working with several collaborators on the development and use of a spectral element code for comparative numerical and experimental studies on challenging problems in fluid mechanics and heat transfer. As illustrated in Fig. 1, these problems include the generation of hairpin vortices resulting from the interaction of a flat-plate boundary layer with a hemispherical roughness element [25, 26]; flow in a carotid artery; Rayleigh-Taylor instabilities [28]; forced convective heat transfer in grooved and grooved-flat channels [12]; and modeling the geophysical fluid flow cell (GFFC) space laboratory experiment of buoyant convection in a rotating hemispherical shell [14]. This paper presents a brief overview of the critical algorithmic and implementation features of our numerical approach that have led to efficient simulation of these problems on modern parallel architectures.

Our simulations are based on numerical integration of the unsteady incompressible Navier-Stokes equations,

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} \\ -\nabla\cdot\mathbf{u} &= 0, \end{aligned}$$

coupled with appropriate boundary conditions on the velocity, $\mathbf{u}$. Temporal discretization is based on stable, high-order, operator-splitting formulations that permit large time steps (typ. a convective CFL of 1–5). Spatial discretization is based on spectrally convergent, high-order, weighted residual techniques employing tensor-product polynomial bases on deformed quadrilateral or hexahedral elements. The resultant symmetric systems are solved using conjugate gradient iteration with scalable Jacobi and additive Schwarz preconditioners. For the latter, we have developed a fast parallel coarse-grid solver that readily scales to thousands of processors. The tensor-product bases lead to matrix-free operator evaluations having favorable storage and work estimates of $O(KN^d)$ and $O(KN^{d+1})$, respectively, for discretizations in $\mathbb{R}^d$ involving K elements of order N. Moreover, the tensor-product-based operator evaluation can be cast as matrix-matrix products, implying that the work complexity estimate has an extremely low constant on modern cache-based architectures. Because the weighted residual formulation requires only C^0 continuity, the use of boundary-minimal

[1]Center on Astrophysical Thermonuclear Flashes, University of Chicago, Chicago, IL 60637.
[2]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439.

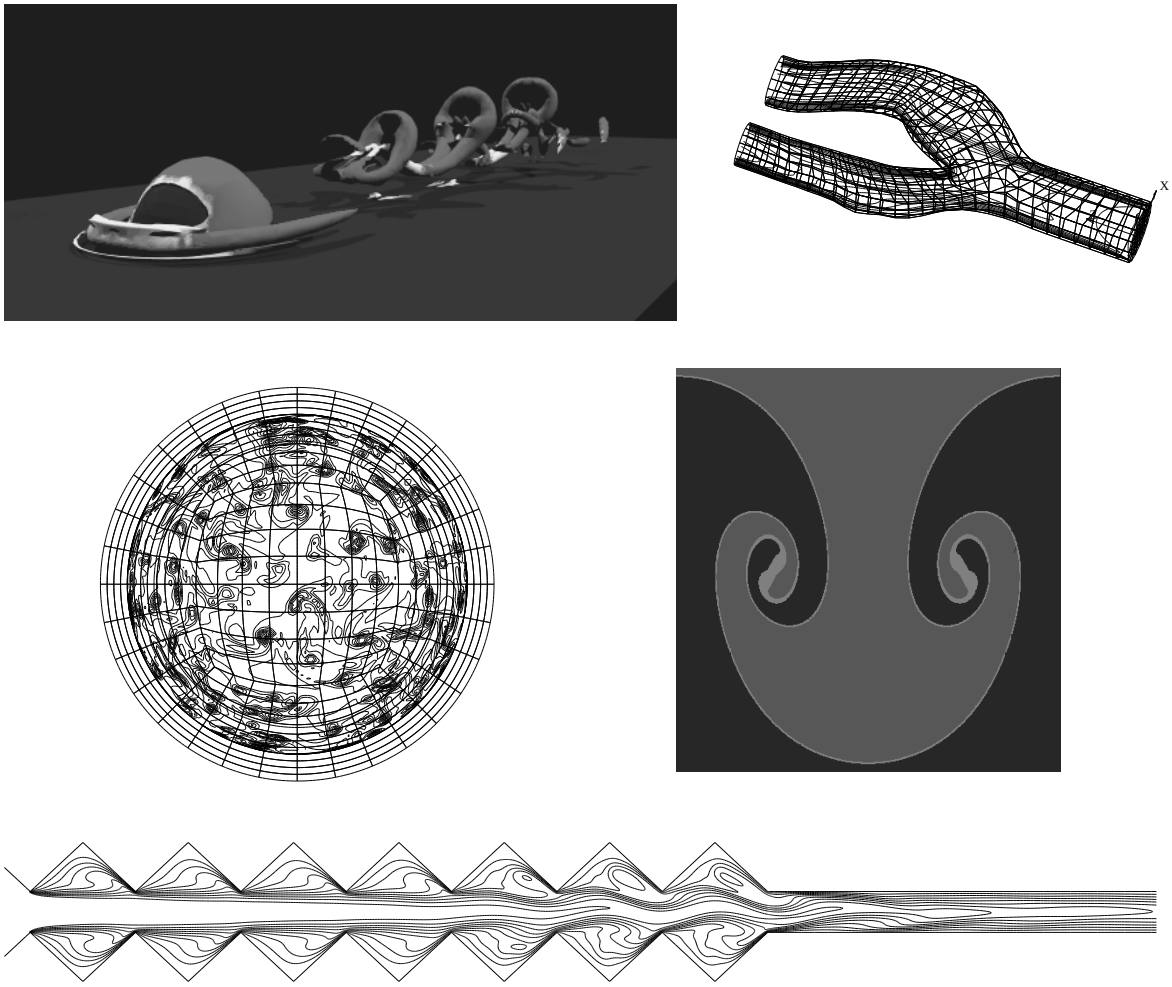

Figure 1: Recent spectral element simulations. Clockwise from top left: hairpin vortex generation in wake of hemispherical roughness element ($Re_\delta = 700$); flow in a carotid artery; two-dimensional Rayleigh-Taylor instability; temporal-spatial evolution of convective instability in heat-transfer augmentation simulations; spherical convection simulation of the geophysical fluid flow cell (GFFC) at $Ra = 1.1 \times 10^5$, $Ta = 1.4 \times 10^6$.

bases implies that the stencil depth does not increase with order, and interprocessor communication costs are consequently equivalent to standard low-order formulations.

Our production code runs on a number of different platforms, including the Cray T3E, the SGI Origin2000, the IBM SP, networks of workstations, and the ASCI-Red machine at Sandia. The code handles general axisymmetric, two-dimensional, and three-dimensional flow configurations; supports a broad range of boundary conditions for hydrodynamics and multiple-species transport; and is currently being used for a variety of applications, as illustrated in Fig. 1. For performance benchmarking, we consider the first of these, the interaction of a flat-plate boundary layer with an isolated hemispherical roughness element, at Reynolds number $Re_\delta = 1600$. Simulations of the hairpin vortex problem have been run on 2048 333 MHz nodes of ASCI-Red in both in single- and dual-processor mode, with sustained performance of 319 GF being achieved for the latter.

The paper is organized as follows. Section 2 provides an overview of the spectral element method. Section 3 discusses matrix-free operator evaluation. Section 4 gives a brief outline of the time advancement scheme. Section 5 describes the principal components of the linear solvers. Section 6 discusses implementation and tuning issues. Performance results for simulations on ASCI-Red are presented in Section 7. A brief conclusion is given in Section 8.

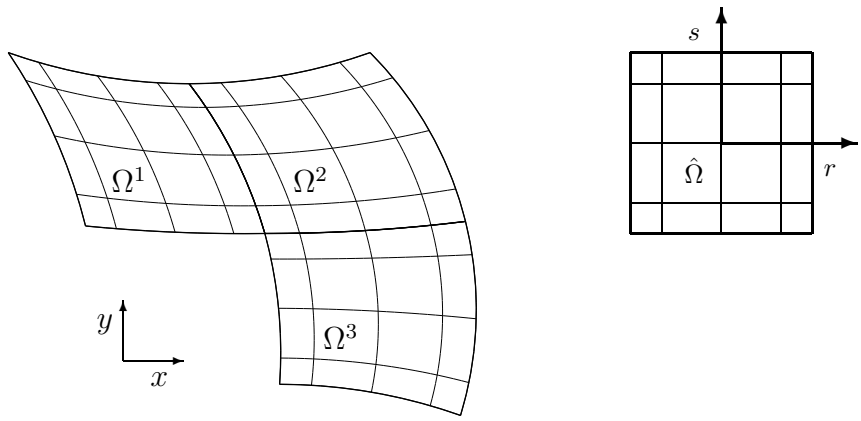

Figure 2: Spectral element discretization in $\mathbb{R}^2$ showing GL nodal lines for $(K, N) = (3, 4)$.

2 Spectral Element Discretization

The spectral element method is a high-order weighted residual technique developed by Patera and coworkers in the '80s that couples the tensor product efficiency of global spectral methods with the geometric flexibility of finite elements [18, 21]. Locally, the mesh is structured, with the solution, data, and geometry expressed as sums of Nth-order tensor product Lagrange polynomials, based on the Gauss or Gauss-Lobatto (GL) quadrature points. Globally, the mesh is an unstructured array of K deformed hexahedral elements and can include geometrically nonconforming elements. The discretization is illustrated in Fig. 2, which shows a three-element mesh in $\mathbb{R}^2$ with the GL grid for the case $N = 4$. Also shown is the reference (r, s) coordinate system used for all function evaluations. Functions in the mapped coordinates are of the form

$$u(\mathbf{x}^k(r,s))\big|_{\Omega^k} \;=\; \sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij}^k h_i^N(r) h_j^N(s)\,, \tag{1}$$

where u_{ij}^k is the nodal basis coefficient; h_i^N is the Lagrange polynomial of degree N based on the GL quadrature points, $\{\xi_j^N\}_{j=0}^N$; and $\mathbf{x}^k(r,s)$ is the coordinate mapping from the reference domain, $\hat{\Omega} := [-1,1]^d$, to Ω^k. The use of the GL basis for the interpolants leads to efficient quadrature for the weighted residual schemes and greatly simplifies operator evaluation in the case of deformed elements.

For problems having smooth solutions, such as the incompressible Navier-Stokes equations, exponential convergence is obtained with increasing N, despite the fact that only C^0 continuity is enforced across element interfaces. This is demonstrated in Table 1, which shows the error in computed growth rates when a small-amplitude Tollmien-Schlichting wave is superimposed on plane Poiseuille channel flow at $Re = 7500$, following [9, 20]. The amplitude of the perturbation is 10^{-5}, implying that the nonlinear Navier-Stokes results can be compared with linear theory to about five significant digits. From Table 1, it is clear that doubling the number of points in each spatial direction yields several orders of magnitude reduction in error, implying that just a small increase in resolution is required for very good accuracy. The significance of this is underscored by the fact

Table 1: Spatial and temporal convergence, Orr-Sommerfeld problem, $K = 15$

	$\Delta t = 0.003125$		$N = 17$	2nd Order		3rd Order	
N	$\alpha = 0.0$	$\alpha = 0.2$	Δt	$\alpha = 0.0$	$\alpha = 0.2$	$\alpha = 0.0$	$\alpha = 0.2$
7	0.23641	0.27450	0.20000	0.12621	0.12621	171.370	0.02066
9	0.00173	0.11929	0.10000	0.03465	0.03465	0.00267	0.00268
11	0.00455	0.01114	0.05000	0.00910	0.00911	161.134	0.00040
13	0.00004	0.00074	0.02500	0.00238	0.00238	1.04463	0.00012
15	0.00010	0.00017	0.01250	0.00065	0.00066	0.00008	0.00008

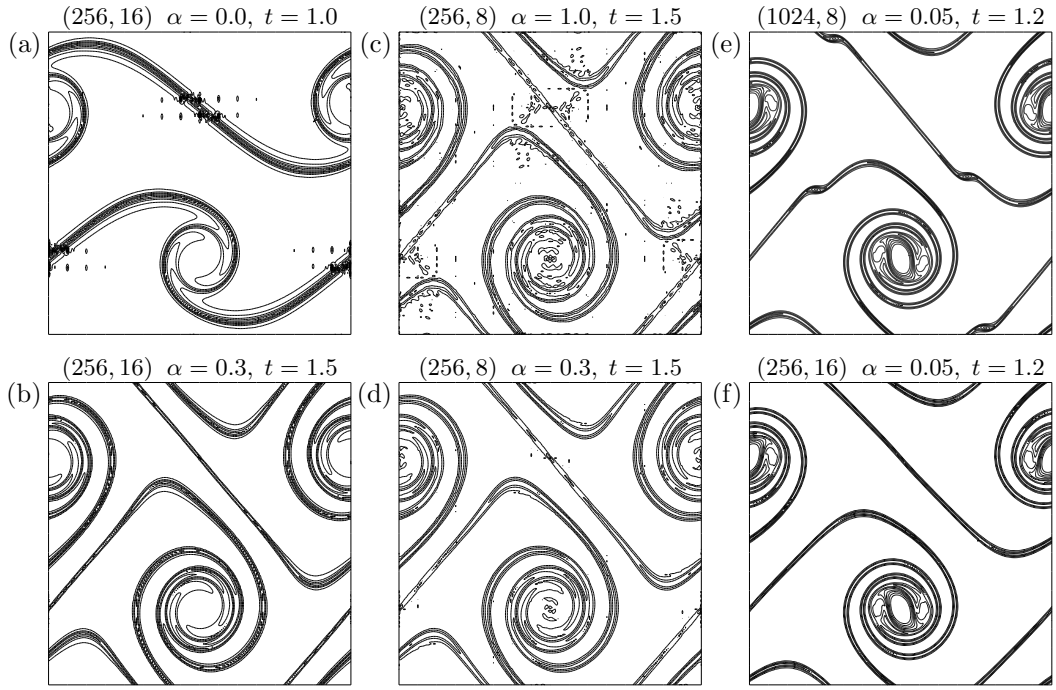

Figure 3: Vorticity contours for high Reynolds number simulations of shear layer roll-up at different (K, N) pairings: (a–d) "thick" shear layer, $\rho = 30$, $Re = 10^5$, contours from -70 to 70 by 140/15; (e–f) "thin" shear layer, $\rho = 100$, $Re = 40,000$, contours from -36 to 36 by 72/13 (cf. Fig. 3c in [4]).

that, in three dimensions, the effect on the number of gridpoints scales as the cube of the relative savings in resolution.

While accuracy is important for simulation it is equally necessary to have stability. We have recently developed a filter-based stabilization procedure for spectral element methods that greatly improves their performance in high-Reynolds number applications [11]. The filter is applied once per timestep and only requires (inexpensive) local interpolation to suppress the Nth mode in each element. The parameter α in Table 1 reflects the strength of the filter, with $\alpha = 0$ implying no filtering and $\alpha = 1$ implying complete suppression of the Nth mode. The results in Table 1 show that the filter slightly degrades spatial accuracy, but that exponential convergence is nonetheless attained. More remarkable are the temporal convergence results, which show that $O(\Delta t^2)$ and $O(\Delta t^3)$ convergence is attained for the filtered case, despite the fact that the third-order scheme on its own ($\alpha = 0$) is unstable. In this case, the stability provided by the filter permits the use of higher-order temporal schemes, thereby allowing a larger timestep for a given accuracy.

The benefits of the stable high-order schemes are perhaps best demonstrated by results in Fig. 3 for the high Reynolds number shear layer roll-up problems studied in [3, 4]. Doubly periodic boundary conditions are applied on $\Omega := [0, 1]^2$, with initial conditions

$$u = \begin{cases} \tanh(\rho(y - 0.25)) & \text{for } y \leq 0.5 \\ \tanh(\rho(0.75 - y)) & \text{for } y > 0.5 \end{cases}, \quad v = 0.05 \sin(2\pi x).$$

Each case consists of a 16×16 array of elements, save for (e), which is 32×32. The timestep size is $\Delta t = .002$ in all cases, corresponding to CFL numbers in the range of 1 to 5. Without filtering, we are unable to simulate this problem at any reasonable resolution. In (a), we see the results just prior to blowup for the unfiltered case with $N = 16$, corresponding to an $n \times n$ grid with $n = 256$. Unfiltered results for $N = 8$ ($n = 128$) and $N = 32$ ($n = 512$) are similar. Filtering with $\alpha = 0.3$ yields dramatic improvement for $n = 256$ (b) and $n = 128$ (d). Although full projection ($\alpha = 1$) is also stable, it is clear by comparing (c) with (d) that partial filtering ($\alpha < 1$) is preferable. Finally, (e) and (f) correspond to the difficult "thin" shear layer case [4]. The spurious vortices in (e) are eliminated in (f) by increasing the order to $N = 16$ at fixed resolution ($n = 256$). We note that the converged result in (f) was unattainable by the second- or fourth-order schemes considered in [4] at resolutions of $n = 512$ or $n = 256$, respectively.

3 Operator Evaluation

The computational efficiency of spectral element methods derives from their tensor product bases (1). To illustrate, we express the stiffness matrix for an undeformed element k in $\mathbb{R}^2$ as a tensor product sum of one-dimensional operators,

$$A^k \;=\; \widehat{B}_y \otimes \widehat{A}_x \;+\; \widehat{A}_y \otimes \widehat{B}_x \,, \tag{2}$$

where $\widehat{A}_*$ and $\widehat{B}_*$ are the one-dimensional stiffness and mass matrices associated with the respective spatial dimensions. If $\underline{u}^k = u^k_{ij}$ is the matrix of nodal values on element k, then a typical matrix-vector product required of an iterative solver takes the form

$$\begin{aligned} (A^k \underline{u}^k)_{lm} &= \sum_{i=0}^{N}\sum_{j=0}^{N} (\widehat{B}_{y,mj}\widehat{A}_{x,li}\underline{u}^k_{ij} \;+\; \widehat{A}_{y,mj}\widehat{B}_{x,li}\underline{u}^k_{ij}) \\ &= \widehat{A}_x \underline{u}^k \widehat{B}_y^T \;+\; \widehat{B}_x \underline{u}^k \widehat{A}_y^T \,. \end{aligned} \tag{3}$$

The latter form illustrates how the tensor product basis leads to matrix-vector products ($A\underline{u}$) being recast as *matrix-matrix* products, a feature central to the efficiency of spectral element methods.

Similar forms result for other operators and for complex geometries. For example, evaluation of the discrete Laplacian for a deformed hexahedral element in $\mathbb{R}^3$ takes the form

$$A^k \underline{u}^k \;=\; \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix}^T \begin{pmatrix} G_{rr} & G_{rs} & G_{rt} \\ G_{rs} & G_{ss} & G_{st} \\ G_{rt} & G_{st} & G_{tt} \end{pmatrix} \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix} \underline{u}^k \,, \tag{4}$$

where $D_r \;=\; (I \otimes I \otimes \hat{D})$, and so forth, and the G_{ij}'s are diagonal matrices of order $(N+1)^3$ that combine the quadrature weights with the Jacobian and metrics associated with the transformation from the physical to computational domain. The total work per element for the evaluation of (4) is $12N^4 + 15N^3$. As the main memory to cpu bandwidth on modern cache-based architectures lags processor performance, an important criterion in algorithm selection is the number of required memory references. The total number of such references is $7N^3$ per element, which is on par with standard low-order schemes. Note that A^k in (4) is full, implying that the work and storage would be $O(N^6)$ if it was explicitly computed and stored. Similar storage and memory access requirements hold for the other operator evaluations.

4 Time Advancement

The Navier-Stokes time advancement is based on the second-order operator-splitting methods developed in [2, 19]. The convective term is expressed as a material derivative, and the resultant form is discretized via a stable second-order backward-difference formula:

$$\frac{\underline{\tilde{\mathbf{u}}}^{n-2} - 4\underline{\tilde{\mathbf{u}}}^{n-1} + 3\underline{\mathbf{u}}^n}{2\Delta t} = S(\underline{\mathbf{u}}^n) \,,$$

where $S(\underline{\mathbf{u}}^n)$ is the linear symmetric Stokes problem to be solved implicitly, and $\underline{\tilde{\mathbf{u}}}^{n-q}$ is a velocity field at time step $n-q$ that is computed as the explicit solution to a pure convection problem. The subintegration of the convection term permits values of Δt corresponding to convective CFL numbers of 1–5, thus significantly reducing the number of (expensive) Stokes solves.

The Stokes problem is of the form

$$\begin{bmatrix} \mathbf{H} & -\mathbf{D}^T \\ -\mathbf{D} & 0 \end{bmatrix} \begin{pmatrix} \underline{\mathbf{u}}^n \\ \underline{p}^n \end{pmatrix} = \begin{pmatrix} \mathbf{B}\underline{\mathbf{f}} \\ \underline{0} \end{pmatrix}$$

and is also treated by second-order splitting, resulting in subproblems of the form

$$H\underline{u}^n_i \;=\; \underline{f}_i \,,$$

for the each velocity component ($i = 1, \ldots, 3$), and

$$E\underline{p}^n \quad = \quad \underline{g}^n \, .$$

Here, H is a diagonally dominant Helmholtz operator representing the parabolic component of the momentum equations and is readily treated via Jacobi-preconditioned conjugate gradients; $E := \mathbf{D}\mathbf{B}^{-1}\mathbf{D}^T$ is the Stokes Schur complement governing the pressure; and $\mathbf{B}$ is the (diagonal) mass matrix in the velocity space. E is a consistent Poisson operator and is effectively preconditioned by using an overlapping additive Schwarz procedure based on low-order Laplacians [9, 10].

5 Solvers

Efficient solution of the Navier-Stokes equations in complex domains depends on the availability of fast solvers for sparse linear systems. For unsteady incompressible flows, the pressure operator is the leading contributor to stiffness, as the characteristic propagation speed is infinite. Our pressure solution procedure involves two stages. First, we exploit the fact that we are solving similar problems from one step to the next, by projecting the current solution onto a subspace of previous solutions. The remaining component is then computed using a scalable domain-decomposition-based iterative solver.

When solving unsteady problems with iterative methods, significant computational savings can be realized by first projecting the solution at time level n onto the space of l previous solutions to generate a high-quality initial guess and then solving only for the perturbation. For the pressure problem, this amounts to computing

$$E\Delta\underline{p}^n = \underline{g}^n - E\bar{\underline{p}}^n, \quad \bar{\underline{p}}^n := \arg\min_{\underline{q}\in V_l} \left\|\underline{p} - \underline{q}\right\|_E, \quad V_l := \{\underline{p}^{n-1}, \ldots, \underline{p}^{n-l}\}\,,$$

(typ., $1 \le l \le L \sim 25$). The projection is implemented via the following Gram-Schmidt process

$$\begin{aligned}
&(i) \qquad \bar{\underline{p}} = \sum_{i=1}^{l} \alpha_i \tilde{\underline{p}}_i, \;\; \alpha_i := \tilde{\underline{p}}_i^T \underline{g}^n. \\
&(ii) \qquad \textit{Solve}: \;\; E\Delta\underline{p} = \underline{g}^n - E\bar{\underline{p}} \;\; \textit{to tolerance } \epsilon. \\
&(iii) \qquad \tilde{\underline{p}}_{l+1} = (\Delta\underline{p} - \sum_{i=1}^{l} \beta_i \tilde{\underline{p}}_i)/||\Delta\underline{p} - \sum_{i=1}^{l} \beta_i \tilde{\underline{p}}_i||_E, \;\; \beta_i := \tilde{\underline{p}}_i^T E\Delta\underline{p}.
\end{aligned} \tag{5}$$

The first step computes an initial guess, $\bar{\underline{p}}$, as a projection in the E-norm ($||\underline{p}||_E := (\underline{p}^T E\underline{p})^{\frac{1}{2}}$) of $\underline{p}^n$ onto an existing basis, $\{\tilde{\underline{p}}_1, \ldots, \tilde{\underline{p}}_l\}$. The second computes the remaining (orthogonal) perturbation

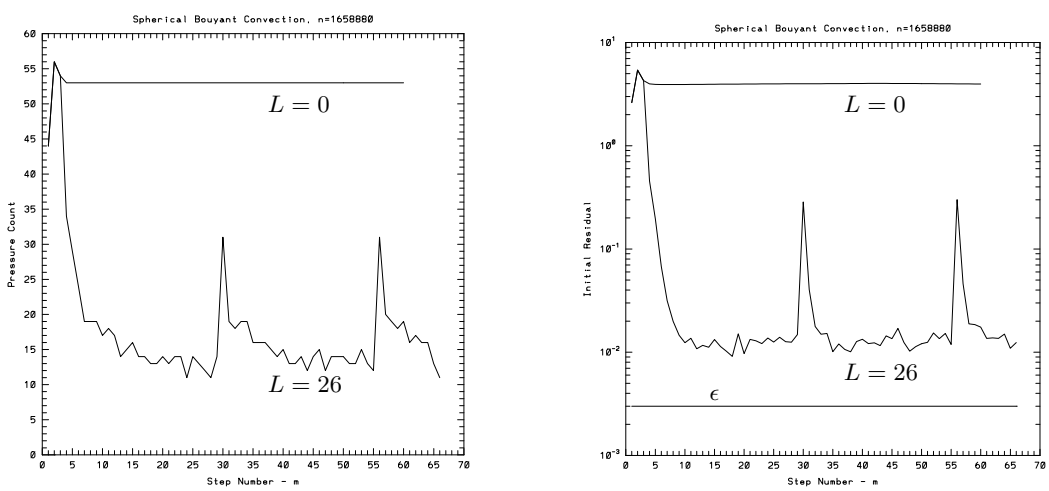

Figure 4: Iteration count (left) and residual history (right) with and without projection for a 1,658,880 degree-of-freedom pressure system associated with spherical convection problem of Fig. 1.

to a specified absolute tolerance, ϵ. The third constructs an orthonormal basis for V_l by augmenting the approximation space with the current (orthonormalized) solution. The approximation space is restarted once ($l > L$) by setting $\tilde{\underline{p}}_1 := \underline{p}^n / ||\underline{p}^n||_E$. The projection scheme requires two matrix-vector products per timestep, one in step *(ii)* and one in step *(iii)*. (Note that it is not possible to use $\underline{g}^n - E\bar{\underline{p}}$ in place of $E\Delta\underline{p}$ in *(iii)* because *(ii)* is satisfied only to within ϵ.)

As shown in [7], the projection procedure can be extended to any parameter-dependent problem and has many desirable properties. It can be coupled with any iterative solver, which is treated as a black box (5*ii*). It gives the *best fit* in the space of prior solutions and is therefore superior to extrapolation. It converges rapidly, with the magnitude of the perturbation scaling as $O(\Delta t^l)+O(\epsilon)$. The classical Gram-Schmidt procedure is observed to be stable ($\Delta\underline{p}$ is nearly orthogonal to V_l) and has low communication requirements because the inner products for the basis coefficients can be computed in concert. Under normal production tolerances, the projection technique yields a two- to fourfold reduction in work. This is illustrated in Fig. 4, which shows the reduction in residual and iteration count for the buoyancy-driven spherical convection problem of Fig. 1, computed with $K = 7680$ elements of order $N = 7$ (1,658,880 pressure degrees of freedom). The iteration count is reduced by a factor of 2.5 to 5 over the unprojected ($L = 0$) case, and the initial residual is reduced by two and one-half orders of magnitude.

The perturbed problem (5*ii*) is solved using conjugate gradients, preconditioned by the additive overlapping Schwarz method introduced by Dryja and Widlund [5]. The spectral element implementation summarized here was developed in [9, 10]. The preconditioner is expressed as

$$M_o^{-1} := R_0^T A_0^{-1} R_0 \; + \; \sum_{k=1}^{K} R_k^T \tilde{A}_k^{-1} R_k \, .$$

It requires a local solve for each (overlapping) subdomain ($\tilde{A}_k^{-1}$), plus a coarse-grid solve (A_0^{-1}) based on the spectral element vertex mesh. The operators R_k and R_k^T are simply Boolean restriction and prolongation matrices that map data between the global and local representations, while R_0 and R_0^T map between the fine and coarse grids. The method has a natural parallel aspect in that the subdomain problems can be solved independently. Parallelization of the coarse-grid component is less trivial and is discussed below. The local subdomain solves exploit the tensor product basis of the spectral element method. Elements are extended by a single gridpoint in each of the directions normal to their boundaries, and a low-order finite element Laplacian, $\tilde{A}_k$, is constructed on the extended domain, $\tilde{\Omega}^k$, using a form identical to (2). Fig. 5 contrasts a two-dimensional example of the domain extension with an earlier unstructured approach based on the finite element method (FEM) [9]. With the tensor product-based construction, it is possible to exploit the fast diagonalization method (FDM) [17], in which the inverse of $\tilde{A}_k^{-1}$ (2) is expressed as

$$\tilde{A}_k^{-1} \; = \; (S_y \otimes S_x)[I \otimes \Lambda_x \; + \; \Lambda_y \otimes I]^{-1} (S_y^T \otimes S_x^T) \, ,$$

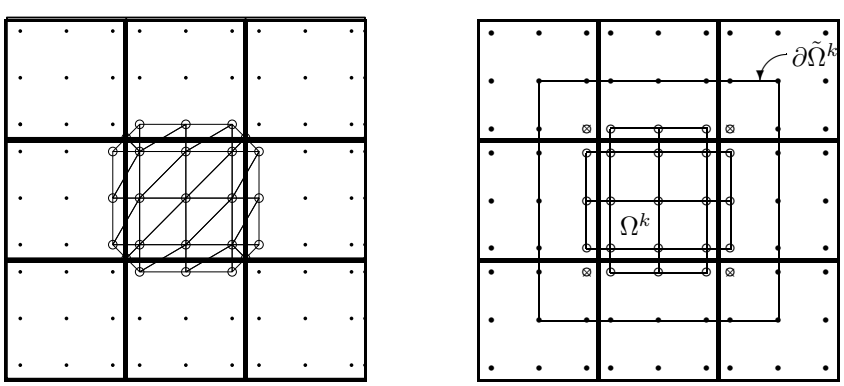

Figure 5: Degrees of freedom (open circles) for FEM-based (left) and tensor-product-based (right) discretizations of local problems. Values at nodes marked "$\otimes$" are set to zero by R_k. Homogeneous Dirichlet boundary conditions are applied on $\partial\tilde{\Omega}^k$.

Table 2: Additive Schwarz for cylinder problem, $N = 7$, $\epsilon = 10^{-5}$

	FDM		$N_o = 0$		$N_o = 1$		$N_o = 3$		$A_0 = 0$	
K	iter	cpu	iter	cpu	iter	cpu	iter	cpu	iter	cpu
93	67	4.4	121	10	64	5.9	49	5.6	169	19
372	114	37	203	74	106	43	73	39	364	193
1488	166	225	303	470	158	274	107	242	802	1798

where S_* is the matrix of eigenvectors and Λ_* the diagonal matrix of eigenvalues solving the generalized eigenvalue problem $\tilde{A}_*\underline{z} = \lambda\tilde{B}_*\underline{z}$. The tensor product forms involving S_* can be applied via fast matrix-matrix products as in (3). A significant advantage of the tensor product basis is the complexity for the local solves, which is of the same order as the matrix-vector product evaluation, $O(KN^3)$ storage and $O(KN^4)$ work in $\mathbb{R}^3$, with significantly smaller constants because the Laplacian is simpler than the consistent Poisson operator, E. While the tensor product form (2) is not strictly applicable to deformed elements, it suffices for preconditioning purposes to build $\tilde{A}_k$ on a rectilinear domain of roughly the same dimensions as Ω^k [10].

Table 2 shows the performance of the FDM-based overlapping Schwarz procedure for the two-dimensional model problem of start-up flow past a cylinder at $Re_D = 5000$ considered in [9]. The polynomial degree is $N = 7$, and the meshes are obtained through two rounds of quad-refinement from an initial mesh having $K = 93$ elements. The results are contrasted with the FEM-based local solves of varying overlap, with $N_o = 0$ corresponding to block-Jacobi preconditioning (no overlap) and $N_o = 1$ corresponding to the standard minimal overlap case (one-point extension). The importance of the coarse-grid component is illustrated by the $A_0 = 0$ case, which shows an eightfold increase in iteration count over the corresponding case of $N_o = 3$ *with* a coarse-grid solve. As noted in [9], the increase in iteration count with K is due to the presence of high aspect ratio elements in this model problem. In practice, this effect is mitigated by the fact that low wave number modes that degrade the performance are removed by the projection procedure described above. Table 2 shows that the FDM is competitive with the FEM in terms of iteration count, but is faster overall because of the speed of local solves. In three dimensions, the unstructured FEM approach is not competitive with the tensor product approach due its higher computational complexity.

As the example of Table 2 illustrates, a coarse-grid component is central to the efficiency of preconditioners for problems having slowly decaying Green's functions. As is well-known, the solution of the coarse-grid problem, $\underline{x}_0 = A_0^{-1}\underline{b}_0$, is frequently a source of inefficiency on large distributed-memory architectures [6, 13]. The difficulty arises because the solution and data are distributed vectors, and A_0^{-1} is completely full, implying the need for an all-to-all communication. Moreover, because there is very little work on the coarse grid (typ. O(1) d.o.f. per processor), the problem is communication intensive. We have recently developed a fast coarse-grid solution algorithm based on projection that readily extends to thousands of processors [8, 24]. For $A_0 \in \mathbb{R}^{n\times n}$ symmetric positive definite, and $X := (\tilde{\underline{x}}_1, \ldots, \tilde{\underline{x}}_n)$ a matrix of A_0-orthonormal vectors satisfying $\tilde{\underline{x}}_i^T A_0 \tilde{\underline{x}}_j = \delta_{ij}$, the projection of $\underline{x}$ onto the range of X ($\mathcal{R}(X)$) is computed as

$$\bar{\underline{x}} := \sum_{i=1}^{n} \alpha_i \tilde{\underline{x}}_i = XX^T\underline{b}, \qquad \alpha_i := \tilde{\underline{x}}_i^T\underline{b}. \tag{6}$$

Since $\bar{\underline{x}}$ is the best fit in $\mathcal{R}(X) \equiv \mathbb{R}^n$, we have $\bar{\underline{x}} = \underline{x}$ and $XX^T = A_0^{-1}$.

The projection procedure (6) is similar to (5i), save that the basis vectors $\{\tilde{\underline{x}}_i\}$ are chosen to be *sparse*. Such sparse sets can be readily found by recognizing that, for any gridpoint i exterior to the stencil of j, there exists a pair of A_0-conjugate unit vectors, $\hat{\underline{e}}_i$ and $\hat{\underline{e}}_j$. For example, for a regular n-point mesh in $\mathbb{R}^2$ discretized with a standard five-point stencil, one can immediately identify half of the unit vectors (associated, e.g., with the "red" squares) in $\mathbb{R}^n$ as unnormalized elements of X. The remainder of X can be created by applying Gram-Schmidt orthogonalization to the remainder of $\mathbb{R}^n$. In [8, 24], it is shown that nested dissection provides a systematic approach to identifying a sparse basis and yields a factorization of A_0^{-1} with $O(n^{\frac{2d-1}{d}})$ nonzeros for n-point grid problems in $\mathbb{R}^d$, $d \geq 2$. Moreover, the required communication volume on a P-processor machine is bounded by

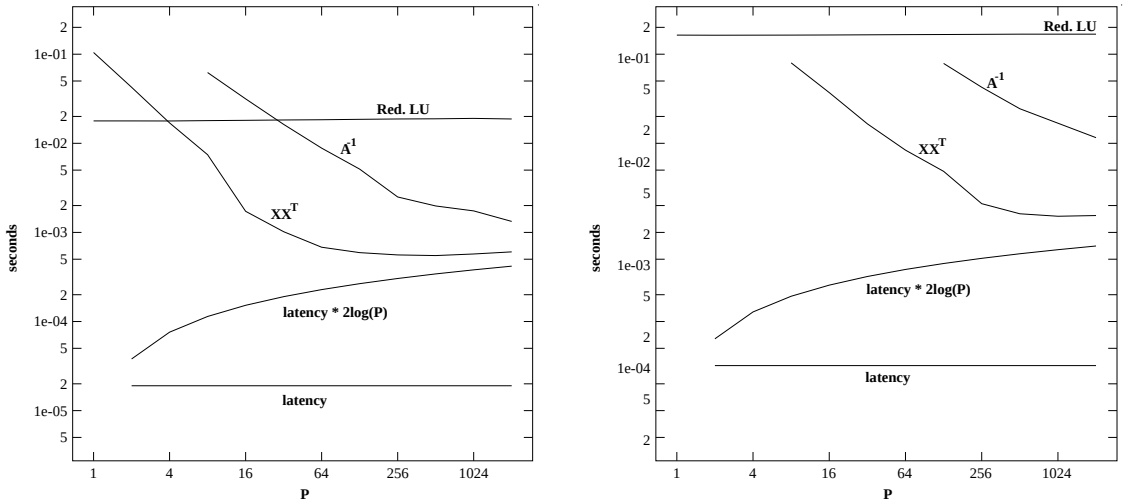

Figure 6: ASCI-Red solve times for a 3969 (left) and 16129 (right) d.o.f. coarse grid problem.

$3n^{\frac{d-1}{d}} \log_2 P$, a clear gain over the $O(n)$ or $O(n \log_2 P)$ costs incurred by other commonly employed solution methods.

The performance of the XX^T scheme on ASCI-Red is illustrated in Fig. 6 for a (63×63) and (127×127) point Poisson problem ($n = 3069$ and $n = 16129$, respectively) discretized by a standard five-point stencil. Also shown are the times for the commonly used approaches of redundant banded-LU solves and row-distributed A_0^{-1}. The *latency*$*2 \log P$ curve represents a lower-bound on solution time, assuming that the required all-to-all communication uses a contention-free fan-in/fan-out binary tree routing. We see that the XX^T solution time decreases until the number of processors is roughly 16 for the $n = 3969$ case, and 256 for the $n = 16129$ case. Above this, it starts to track the latency curve, offset by a finite amount corresponding to the bandwidth cost, which is bounded by $3n^{\frac{1}{2}} \log P$ in the two-dimensional case. We note that XX^T approach is superior to the distributed A^{-1} approach from a work *and* communication standpoint, as witnessed by the substantially lower solution times in each of the work- and communication-dominated regimes. Further performance results and analysis of the XX^T algorithm are presented in [24].

6 Implementation and Tuning

Our parallel implementation follows the standard message-passing-based SPMD model [15] in which contiguous groups of elements are distributed to processors and computation proceeds in a loosely synchronous manner. The code is constructed from flexible and efficient C and Fortran modules. The computational kernel is built from optimized basic linear algebra subroutines (BLAS) (primarily level 3). Our communication routines are built on top of the NX or MPI message-passing libraries.

For a given polynomial degree N define $N_1 := N+1$ and $N_2 := N-1$. Then the local subdomain solves, $(\tilde{A}_k^{-1})$, and application of the derivative, D_*, Helmholtz, H, and pressure, E, operators require matrix-matrix products of form $(n_1 \times n_2) \times (n_2 \times n_3)$, where $n_1 = N_1, N_1^2, N_2$, or N_2^2, $n_2 = N_1$ or N_2, and $n_3 = N_1, N_1^2, N_2$, or N_2^2. In addition, mapping from (to) the pressure mesh to (from) the coarse grid requires matrix-matrix products of form $(2 \times N_2) \times (N_2 \times 2)$ and $(N_2 \times 2) \times (2 \times N_2)$, respectively.

As matrix-matrix products account for over 90% of the flops in a simulation, maximizing DGEMM performance is paramount. Table 3 shows performance figures, obtained on one 333 MHz node of ASCI-Red in single-processor mode, for the matrix-matrix product calling configurations (n_1, n_2, n_3) encountered in an order $N = 15$ simulation. On ASCI-Red we have several versions of DGEMM to choose from: the standard version obtained with the `-lkmath` link option (lkm); the version jointly developed by Sandia and Intel obtained with the `-lcsmath` link option (csm); and a version being developed for matrices with $n_2 \leq 20$ by Greg Henry at Intel (ghm). In addition to the library

Table 3: MFLOPS for $(n_1 \times n_2) \times (n_2 \times n_3)$ matrix-matrix product kernel on ASCI-Red

n_1	n_2	n_3	lkm	ghm	csm	f3	f2
14	2	14	23	20	25	41	43
2	14	2	29	23	21	77	68
16	14	16	100	113	114	130	119
16	14	196	107	95	95	89	110
256	14	16	85	100	100	105	67
14	16	14	78	74	74	106	99
16	16	16	82	138	83	105	102
16	16	256	113	147	93	97	118
196	16	14	91	192	147	122	105
256	16	16	90	105	148	111	69

routines, we tested two hand-unrolled versions, both of which unroll the n_2 loop completely. The first, f2, has n_3 control the outer loop while the second, f3, has n_1 control the outer loop. We note that all data in the matrix-matrix product timings is noncached. Unfortunately, no single method was superior across all cases. For the performance study in Section 7, we selected one set from the best of the table (which we label as perf. in the results section) and one set without the new library (which we label as std.).

Since iterative solvers are used, the principal communication kernel is the gather-scatter operation required for the residual vector assembly procedure. Because data is always stored on an element-by-element basis, the gather-scatter procedure required for residual evaluation is combined into a single communication phase wherein nodal values which are shared by adjacent elements are exchanged and summed. This is a single *local-to-local* transformation, rather than separate gather and scatter phases common to many finite element implementations. Communication overhead is further reduced through use of a recursive spectral bisection based element partitioning scheme to minimize the number of vertices shared amongst processors [22].

The gather-scatter operation is implemented by using a stand-alone MPI/NX-based message-passing utility that supports a vector mode for problems having multiple degrees-of-freedom per vertex as well as a general set of commutative/associative operations [27]. The easy-to-use interface requires only two calls:

handle=gs-init(*global-node-numbers,n***)** and **ierr=gs-op(***u,op,handle***)**,

where *global-node-numbers()* associates the n local values contained in the vector *u()* with their global counterparts, and *op* denotes the reduction operation performed on shared elements of $u()$.

Each node on ASCI-Red consists of two Zeon 333 MHz Pentium II processors with 128 megabytes of shared memory that can be run in a message-passing (internode)/SMP (intranode) mode. The fact that we employ nonoverlapping storage for elements and have loops over blocks that exhibit little or no data dependence implies that we can easily gain additional, intranode, parallelism by splitting the loops and spawning additional threads. To achieve this, we use the `-Mconcur` compiler option and directives. Currently, we have cast the three most time-consuming routines into dual-processor mode: the matrix-vector product routines for the pressure operator, E; the Helmholtz operator, H; and the local subdomain solves, $\tilde{A}_k^{-1}$. This approach was sufficient to attain 82% dual-processor efficiency.

7 Performance Results

We have run our spectral element code on a number of distributed-memory platforms, including the Intel Paragon at Caltech, the Cray T3E-600 at NASA Goddard, the SGI Origin 2000 and IBM SP at Argonne, the SGI ASCI-Blue machine at Los Alamos, and the Intel ASCI-Red machine at Sandia. For this paper we concentrate on recent timing results for the hairpin vortex problem of Figs. 1 and 7. This problem is of interest in its own right because it provides a vehicle to study organized transition to turbulence [1, 16, 23]. For benchmarking considerations, we consider impulsively started flow at $Re = 1600$, with an initial condition consisting of a Blasius profile with

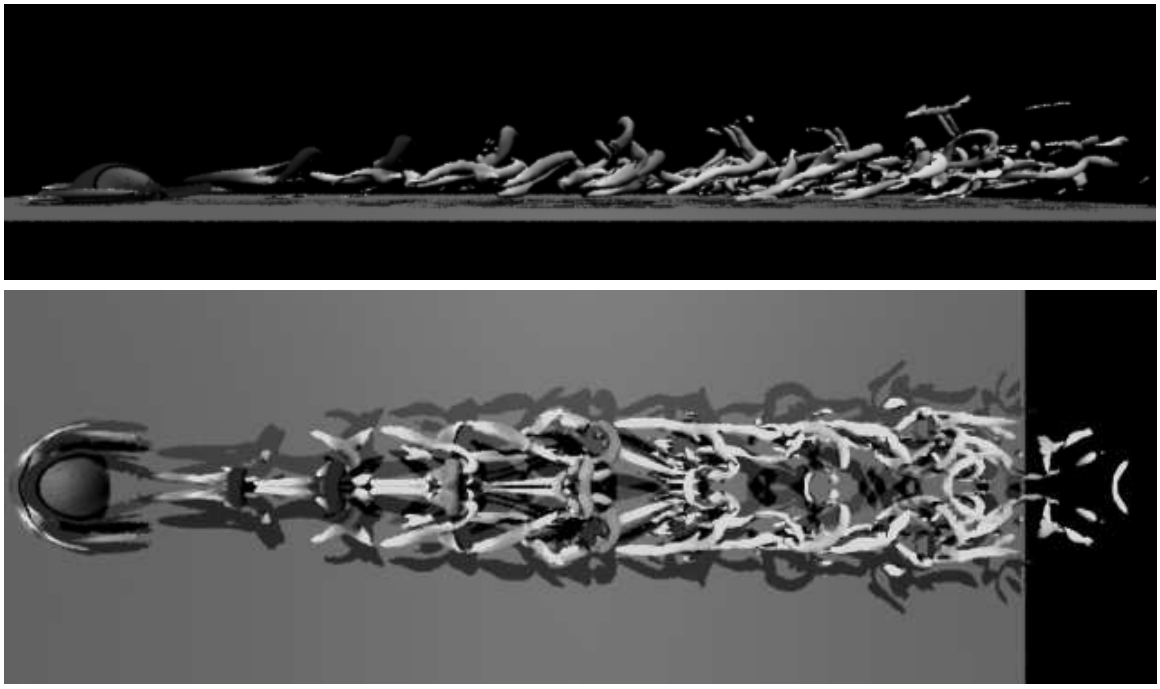

Figure 7: Profile (top) and planform (bottom) views of hairpin and secondary vortices generated in a boundary layer by a hemispherical roughness element for $Re_R = 850$. The spectral element parameters are $(K, N) = (1021, 11)$.

boundary layer thickness $\delta = 1.2R$. The mesh used for these simulations was obtained via an oct-refinement of the production mesh used for the transitional boundary layer/hemisphere calculation of Fig. 7 and contains $K = 8168$ elements of order $N = 15$ (27,799,110 gridpoints for velocity, 22,412,992 for pressure).

Performance results on ASCI-Red are presented for up to 2048 333 MHz nodes in single- and dual-processor mode. Total times are for the time-stepping portion of the runs only. During production runs, usually 14 to 24 hours in length, our setup and I/O costs are typically in the range of 2–5%. To determine floating-point operation count, we access the hardware operation counters via calls to the `perfmon` library. In addition, we have instrumented the code to provide various performance metrics, including a per processor flop count. The two methods yield results within 2% of each other. Finally, all floating-point calculations were done in 64-bit precision.

Figure 8 shows time per step for the first 26 timesteps (left) and the pressure and (x-component) Helmholtz iteration counts (right). The significant reduction in pressure iteration count is due to the difficulty of computing the initial transients and clearly shows the benefits gained from the pressure projection procedure. In typical production runs the number of pressure iterations per timestep settles in at between 30 and 50, which is consistent with the behavior exhibited in the iteration plot. We note that the average time per step for the last five steps of the 319 GF run is 17.5 seconds. Tables 4 and 5 present total time for the 26 timesteps and sustained performance. We note that additional routines were dual-processor enabled subsequent to the creation of Table 4 which resulted in a 10% increase in efficiency. Finally, the coarse grid for this problem has 10,142 distributed degrees of freedom and accounts for 4.0% of the total solution time in the worst-case scenario of 2048 nodes in dual-processor mode. If the A^{-1} approach were use instead this would have increased to 15%.

8 Conclusion

We have developed a highly accurate, stabilized spectral element code based on scalable solver technology that exhibits excellent parallel efficiency and sustains high MFLOPS. It attains exponential

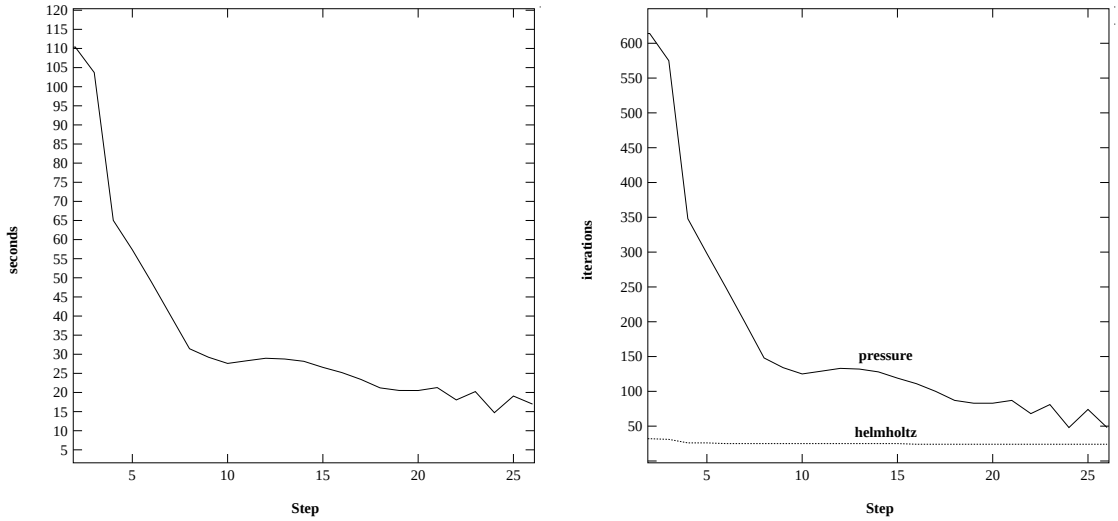

Figure 8: $P = 2048$ ASCI-Red-333 dual-processor mode results for the first 26 timesteps for $(K, N) = (8168, 15)$: solution time per step (left) and number of pressure and (x-component) Helmholtz iterations per step (right).

Table 4: ASCI-Red-333: total time and GFLOPS, std. DGEMM

	$K = 8168$, $N = 15$			
	single		dual	
P	time (s)	GFLOPS	time (s)	GFLOPS
512	6361	47	4410	67
1024	3163	93	2183	135
2048	1617	183	1106	267

convergence, allows a convective CFL of 1–5, and has efficient multilevel elliptic solvers, including a coarse-grid solver that requires minimal communication. The code currently runs on thousands of processors and is clearly ready to run on machines with tens of thousands of processors.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38; by the Department of Energy under Grant No. B341495 to the Center on Astrophysical Thermonuclear Flashes at University of Chicago, and by the University of Chicago. Computational time on the T3E-600 was provided with support from NASA microgravity grant NAG 32139 to the University of Maryland. We give special thanks to Greg Henry at Intel.

References

[1] M. S. Acalar and C. R. Smith, "A study of hairpin vortices in a laminar boundary layer. Part 1. Hairpin vortices generated by a hemisphere protuberance", *J. Fluid Mech.*, **175**, pp. 1–41 (1987).

[2] J. Blair Perot, "An analysis of the fractional step method", *J. Comput. Phys.*, **108**, pp. 51–58 (1993).

[3] J. B. Bell, P. Collela, and H. M. Glaz, "A second-order projection method for the incompressible Navier-Stokes equations," *J. Comp. Phys.*, **85**, pp. 257–283 (1989).

Table 5: ASCI-Red-333: total time and GFLOPS, perf. DGEMM

	$K = 8168$, $N = 15$			
	single		dual	
P	time (s)	GFLOPS	time (s)	GFLOPS
512	5969	50	3646	81
1024	2945	100	1816	163
2048	1521	194	927	319

[4] D. L. Brown and M. L. Minion, "Performance of under-resolved two-dimensional incompressible flow simulations," *J. Comp. Phys.*, **122**, pp. 165–183 (1995).

[5] M. Dryja and O. B. Widlund, "An additive variant of the Schwarz alternating method for the case of many subregions", *Tech. Rep. 339*, Dept. Comp. Sci., Courant Inst., NYU (1987).

[6] C. Farhat and P. S. Chen, "Tailoring domain decomposition methods for efficient parallel coarse grid solution and for systems with many right hand sides", *Contemporary Math.*, **180**, pp. 401–406 (1994).

[7] P. F. Fischer, "Projection techniques for iterative solution of $A\underline{x} = \underline{b}$ with successive right-hand sides", *Comp. Meth. in Appl. Mech.*, **163** pp. 193–204 (1998).

[8] P. F. Fischer, "Parallel multi-level solvers for spectral element methods", in *Proc. Intl. Conf. on Spectral and High-Order Methods '95, Houston, TX*, A. V. Ilin and L. R. Scott, eds., Houston J. Math., pp. 595–604 (1996).

[9] P. F. Fischer, "An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations", *J. of Comp. Phys.*, **133**, pp. 84–101 (1997).

[10] P. F. Fischer, N. I. Miller, and H. M. Tufo, "An overlapping Schwarz method for spectral element simulation of three-dimensional incompressible flows," *ANL/MCS Preprint* P730–1098 (1998).

[11] P. F. Fischer and J. S. Mullen, "Filter-based stabilization of spectral element methods" *ANL/MCS Preprint* 0899, (1999).

[12] R. Faulkner, M. Greiner, P. Fischer, and R. Wirtz, "Direct numerical simulation of three-dimensional flow and heat transfer in a symmetrically grooved channel with constant temperature boundary conditions", presented at *1997 ASME Int. Mech. Eng. Congress and Exp.*, Dallas, TX.

[13] W. D. Gropp, "Parallel computing and domain decomposition", in *Fifth Conf. on Domain Decomposition Methods for Partial Differential Equations*, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., SIAM, Philadelphia, pp. 349–361 (1992).

[14] J. E. Hart, G. A. Glatzmaier, and J. Toomre, "Space-laboratory and numerical simulations of thermal convection in a rotating hemispherical shell with radial gravity", *J. Fluid Mech.* **173**, pp. 519–544 (1986).

[15] M. T. Heath, "The Hypercube: A Tutorial Overview", in *Hypercube Multiprocessors 1986*, M. T. Heath, ed., SIAM, Philadelphia, pp. 7–10 (1986).

[16] P. S. Klebanoff, W. G. Cleveland, and K. D. Tidstrom, "On the evolution of a turbulent boundary layer induced by a three-dimensional roughness element", *J. Fluid Mech.*, **92**, pp. 101–187 (1992).

[17] R. E. Lynch, J. R. Rice, and D. H. Thomas, "Direct solution of partial difference equations by tensor product methods", *Numerische Mathematik*, **6**, pp. 185–199 (1964).

[18] Y. Maday and A. T. Patera, "Spectral element methods for the Navier-Stokes equations", in *State of the Art Surveys in Computational Mechanics*, A. K. Noor, ed., ASME, New York, pp. 71–143 (1989).

[19] Y. Maday, A. T. Patera, and E. M. Rønquist, "An operator-integration-factor splitting method for time-dependent problems: application to incompressible fluid flow", *J. Sci. Comput.*, **5**(4), pp. 263–292 (1990).

[20] M. R. Malik, T. A. Zang, and M. Y. Hussaini, "A spectral collocation method for the Navier-Stokes equations", *J. Comput. Phys.*, **61** pp. 64–88 (1985).

[21] A. T. Patera, "A spectral element method for fluid dynamics: Laminar flow in a channel expansion", *J. Comput. Phys.*, **54**, pp. 468–488 (1984).

[22] A. Pothen, H. D. Simon, and K. P. Liou, "Partitioning sparse matrices with eigenvectors of graphs", *SIAM J. Matrix Anal. Appl.*, **11** 3 pp. 430–452 (1990).

[23] B. A. Singer, "The formation and growth of a hairpin vortex", in *Instability, Transition, and Turbulence*, M. Y. Hussaini, A. Kumar, and C. L. Street, eds., Springer-Verlag, New York, pp.367 (1992).

[24] H. M. Tufo and P. F. Fischer, "Fast parallel direct solvers for coarse-grid problems", *J. Dist. Par. Comp.* (accepted).

[25] H. M. Tufo, P. F. Fischer, M. E. Papka, and M. Szymanski, "Hairpin vortex formation, a case study for unsteady visualization", 41st CUG Conference, 1999.

[26] H. M. Tufo, P. F. Fischer, M. E. Papka, and K. Blom, "Numerical simulation and immersive visualization of hairpin vortices", SC'99, 1999.

[27] H. M. Tufo, "Algorithms for large-scale parallel simulation of unsteady incompressible flows in three-dimensional complex geometries", Ph.D. thesis, Brown University (1998).

[28] Y.-N. Young, H.M. Tufo, and R. Rosner, "On the Miscible Rayleigh-Taylor Instability: Mixing Layer Width Scaling in 2 and 3-D", submitted to *J. Fluid Mech.*