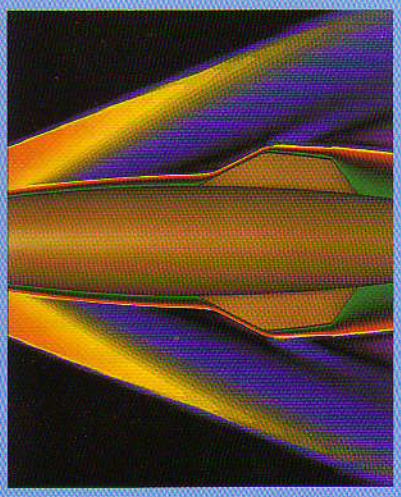


Parallel Finite-Element Computation of 3D Flows

Tayfun Tezduyar, Shahrouz Aliabadi, Marek Behr, Andrew Johnson, and Sanjay Mittal, University of Minnesota

Computations based on implicit methods and implemented on massively parallel computers provide a new capability for solving a large class of practical flow problems.

The capability to visualize flow around an aerospace or surface vehicle enables engineers to evaluate designs with a computer while reducing the need for building physical models and conducting costly wind-tunnel experiments. Designers of automobiles and watercraft, for example, can use these techniques to test new design concepts. Software is used to lay a mesh or grid over the geometry of the test design; intensive flow computation then determines flow variables such as density, pressure, and velocity at thousands of discrete points. The images obtained — sometimes enough to create an animation — enable engineers to visualize the flow of air or water under specified operating conditions.

Carrying out such computations requires solving a set of complex fluid-dynamics equations involving billions of operations; thus, the use of supercomputers has become indispensable. Our work includes the computation of moving boundaries and interfaces, and mesh update strategies. One of the challenges in these computations is parallelizing the software so that many operations can be performed simultaneously on multiprocessor machines, thereby reducing the time needed to obtain results. The computations we describe were carried out on Connection Machines, either a 1,024-node CM-200 or a 512-node CM-5 equipped with vector execution units; they were based on implicit methods, that is, the simultaneous solution of coupled equations.

0018-9162/93/1000-0027$03.00 ©1993 IEEE

Numerical stabilization

In a standard Galerkin finite-element formulation, the test and solution function spaces are the same, and the formulation is derived by starting with global integrations of the products of the governing equations and the test functions. However, finite-element computations based on the standard Galerkin formulation of a flow problem can involve numerical instabilities. We consider both compressible and incompressible flows. In compressible flows, the density of a fluid particle changes as it travels in the flow field. This happens when the flow speeds, measured in terms of Mach number, are sufficiently high to cause fluid bulks to compress or expand. Most flows encountered in aerospace engineering are compressible — for example, flows involving aircraft, missiles, and spacecraft. With incompressible flows, the density of a fluid particle is assumed to be constant as it travels in the flow field. This assumption is valid for low Mach numbers. Most flows involving water fall into this category.

In our parallel computations we always use some type of stabilized formulation.

The numerical instabilities related to the standard Galerkin formulation can result from the presence of a constraint, such as the incompressibility condition, and from the dominant advection terms in the governing equations. To stabilize the finite-element formulation of a given equation system involving advective terms and possibly an incompressibility constraint, we have been using stabilization techniques such as the streamline-upwind/Petrov-Galerkin (SUPG), Galerkin/least-squares (GLS), and pressure-stabilizing/Petrov-Galerkin (PSPG) formulations. With these formulations, potential numerical instabilities are prevented without introducing excessive numerical diffusion and therefore without compromising the accuracy of the solution. Stabilization methods of this type became quite well established after their earlier deployment for both incompressible[1] and compressible[2] flows. Some of these stabilization techniques are based on finite-element discretization in both space and time, and all of them are developed in the context of unstructured meshes. Being able to use unstructured meshes instead of just structured ones gives us the flexibility to lay a mesh over complex geometries encountered in practical problems.

In applications to compressible flows governed by the Euler and Navier-Stokes equations, the SUPG and GLS methods were also used in the context of the entropy variables formulation of the governing equations.[3] With a shock-capturing term added to the formulation, the solutions obtained with the entropy variables were highly accurate. Recently, LeBeau and Tezduyar[4] showed that the SUPG formulation with conservation variables, supplemented with a similar shock-capturing term, gives results that are nearly indistinguishable from those obtained with the entropy variables. In our parallel computations we always use some type of stabilized formulation, usually the SUPG and GLS methods. These stabilization techniques help significantly with the convergence of the iterative strategies used to solve the implicit equation systems arising from these formulations.

Computation of moving boundaries and interfaces

When flow problems involve moving boundaries and interfaces, for example, a liquid drop changing shape, computations are achieved by using the Deformable-Spatial-Domain/Stabilized-Space-Time (DSD/SST) method.[5] This is an accurate, general-purpose stabilized finite-element formulation for computing unsteady flow problems involving free surfaces, two-liquid interfaces, and fluid-structure and fluid-particle interactions.

In this method, the stabilized finite-element formulation of the governing equations is written over the space-time domain of the problem; therefore, the deformation of the spatial domain with respect to time is taken into account automatically. In the DSD/SST method, the mesh is updated in such a way that remeshing is performed only when necessary to prevent unacceptable degrees of mesh distortion. With this approach, less frequent remeshing reduces the projection errors involved in remeshing and also makes parallelizing the computations easier.

In several problems we considered, remeshing was eliminated by designing special meshes and mesh-moving schemes specific to a given problem.[6] In a more general setting, the motion of the mesh is governed by the modified equations of linear homogeneous elasticity,[7,8] thereby minimizing, and in some cases eliminating, the need for remeshing. This, of course, reduces cost and parallel setup overhead. The mesh update schemes we've developed can also use combinations of structured and unstructured meshes.

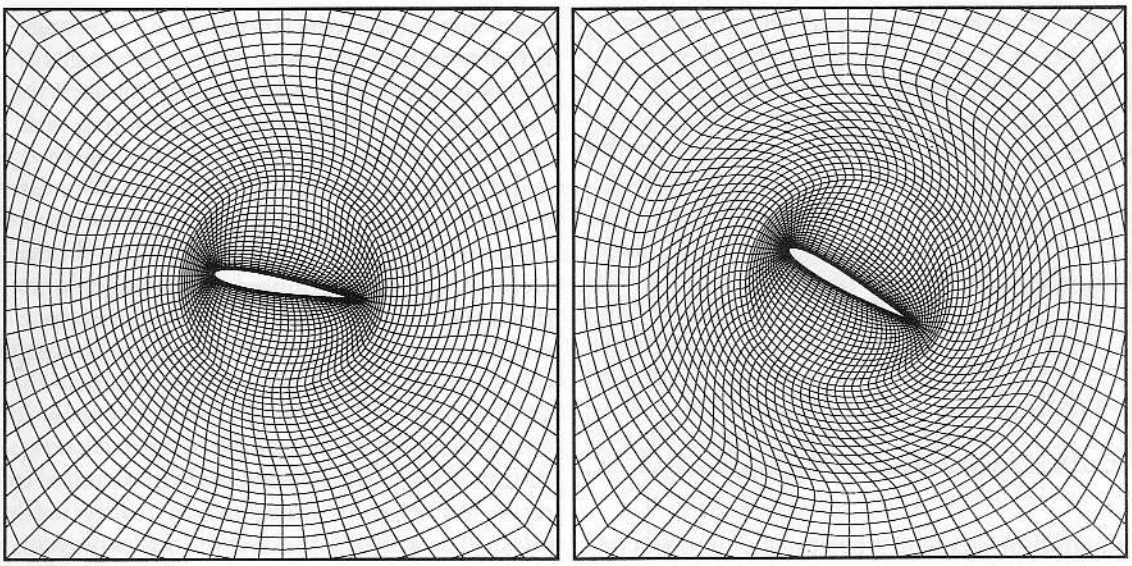


Figure 1. Mesh around a NACA0012 airfoil at two different angles of attack. The figure illustrates how the rotation of the airfoil is absorbed by the flexible circular region in the mesh while the mesh disk adjacent to the airfoil, as well as the outermost mesh region, remain rigid.

In recent years there has been significant progress in massively parallel finite-element computations based on implicit formulations and which assume unstructured grids.[7-10] The formulations for both incompressible and compressible flows have been implemented on massively parallel CM-200 and CM-5 supercomputers, making possible the three-dimensional simulations we describe later. We are now conducting essentially all of our computational fluid dynamics studies, including those in 3D and those involving moving boundaries and interfaces, on these massively parallel machines. The 3D problems successfully modeled include sloshing in a liquid-filled container subjected to vertical vibrations (52,000-plus equations), incompressible flow between two concentric cylinders (282,000-plus equations), supersonic flow past a delta wing (725,000-plus equations), and supersonic flow past a toy missile (1.1 million-plus equations).

Mesh update strategies

If the motion of an object in the domain has a predetermined order, a special problem-dependent algebraic mesh-update scheme can be used to facilitate this motion. Take, for example, a pitching airfoil.[6] We have a rigid airfoil that is allowed to rotate throughout a wide range of angles of attack. A mesh update scheme in which a mesh is moved to adapt to the desired angle of attack was designed specifically for this problem. Figure 1 shows meshes at different angles of attack; they contain a solid circular core that rotates rigidly with the airfoil. Outside this region is a deforming region between the solid rotation zone and the fixed portion of the mesh that lies outside the deforming zone.

If the motion of an object or an interface is more general, it is desirable to have a mesh update scheme that is independent of the type of mesh used and the type of motion. For these cases, we have implemented a new automatic mesh update scheme in which a linear elastostatics problem is solved each time the mesh is required to move. In this elastostatics problem, desired displacements (movements) at the boundaries and interfaces become boundary conditions. To better preserve the structure of the original mesh in the more refined areas, it is desirable to have greater stiffness where smaller elements are present. To achieve this, we drop from the formulation the Jacobian of the transformation from the element domain to the physical domain.

This mesh update strategy has been applied to the case of a viscous drop falling in a viscous fluid.[7] In this axisymmetric problem, gravity is applied to a combination of a heavier liquid embedded inside a lighter one. The heavier liquid (the drop) falls and deforms until a steady-state solution is reached. The whole computational domain is forced to translate with the center of gravity of the drop, and on top of this motion, our automatic mesh update scheme will deform the mesh so that it conforms to the desired shape of the drop. Figure 2 shows the mesh in its initial, undeformed shape and in its final shape at terminal velocity.

This mesh update method is applicable to unstructured meshes such as those with triangular elements generated with general-purpose mesh generators. Also, meshes involving a com-

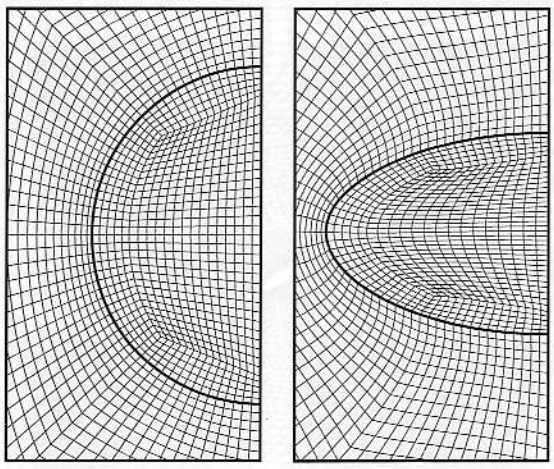


Figure 2. Initial and final meshes for a viscous drop falling in a viscous fluid. In this example the mesh in the entire domain deforms as a result of the deformation of the interface between the drop and the surrounding fluid (heavy line). The displacement of the nodes not belonging to that interface is determined by solving a linear elastostatics problem.

bination of element types can be used in conjunction with the automatic mesh update scheme. Figure 3 shows a mesh around a NACA0012 airfoil with bilinear-quadrilateral elements adjacent to the airfoil; triangular elements (generated with the Emc[2] mesh generator from INRIA, France) fill out the rest of the domain. In such a mesh, it is desirable to move the quadrilateral region as a solid body and let the triangular elements deform to accommodate the motion of the airfoil. Figure 3 shows this type of movement.

Massively parallel implementations

The bulk of the computations in an implicit finite-element program will typically occur in two stages. First is the formation of the element-level matrices and residual vectors. Second is the solution of a linear system of equations, whose components were constructed in the first stage. Efficient implementation of these two stages on a massively parallel architecture is at once the most important and the most challenging task.

The starting point in the design of a massively parallel implementation is deciding how to distribute the variables among the computer's processors. Since most of the massively parallel architectures are distributed-memory machines, the placement of the data is of paramount importance and can radically affect the performance of the implementation. We will assume that the architecture chosen has a developed concept of virtual processors, so that each of the data entries subject to a parallel operation can be assumed to reside on its own processor. In reality, each physical processor takes over the duties of several virtual processors.

We will use two primary data-storage modes, Elem and Eqn. The Elem mode is used for storing element-level data, with one element and its degrees of freedom associated with exactly one virtual processor. The Eqn mode will hold variables at the level of the global equation system, with the data corresponding to a single equation assigned to a single processor. The nodal data, coordinates, element-level properties, and element-level matrices are stored in the Elem mode. The global increment vector, global residual vector, and certain intermediate variables used in the solution stage are kept in the Eqn form.

The mapping between the two data sets is denoted LM (location matrix): Elem $\rightarrow$ Eqn. Communication operation Elem $\leftarrow$ Eqn is a gather operation, while movement of the data in

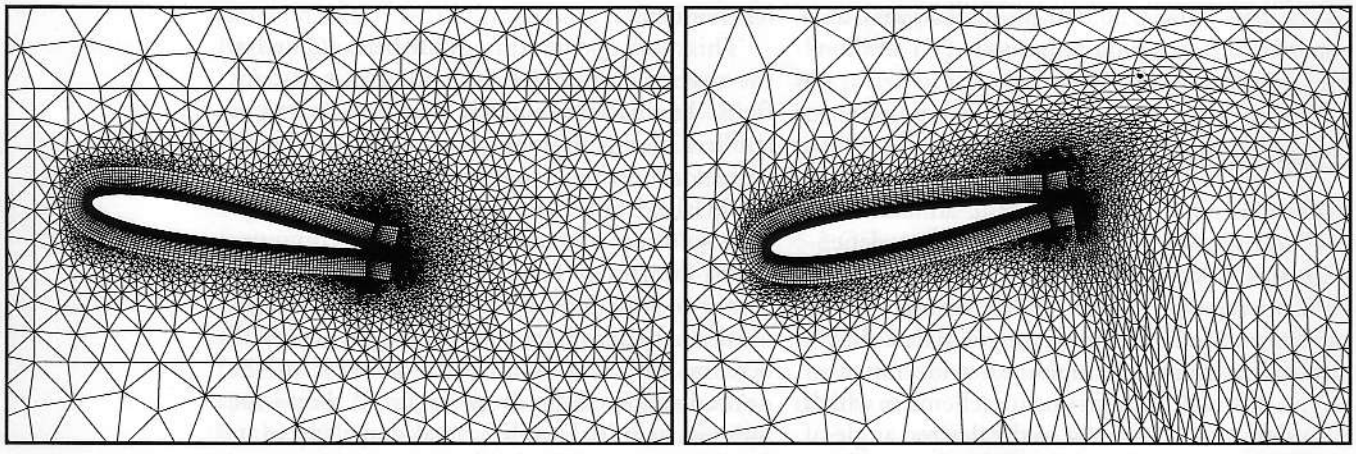


Figure 3. Mesh around a NACA0012 airfoil at two different angles of attack, with mixed structured-unstructured meshes. Here the mesh region adjacent to the airfoil is filled with a structured mesh of quadrilateral elements, ensuring that the solution in the boundary-layer region is of good quality. The outer region is filled with unstructured triangular elements, providing an interface between the obstacle and the outer boundary of the domain and other obstacles. Such an interface is hard to achieve with structured-mesh techniques.

the opposite direction, Elem → Eqn, is a scatter. The scatter is usually coupled with a combining operation — for example, addition or overwriting — at the destination. Both gather and scatter can be implemented efficiently on the Connection Machine computers, provided they are done repeatedly with a static communication pattern as defined by LM. When this is done, the communication trace can be saved the first time communication is performed, resulting in extremely fast subsequent gathers and scatters.

The process used here for solving the global linear system does not require assembly of the global matrix in any form. Instead, it operates on unassembled element-level matrices. Therefore, apart from a simple assembly, or scatter, of the global residual vector from element-level residuals, the task of forming the equation system takes place entirely at the element level. Consequently, in the matrix formation phase, no interprocessor communication is involved, and the parallelism of the operations can be fully exploited. Behr et al.[10] provide a more detailed description of the matrix formation phase, including a pseudocode fragment.

To solve the linear system, we've implemented[10] a data-parallel version of the GMRes technique. The Generalized Minimum Residual is an iterative method for solving large linear systems of equations with a nonsymmetric coefficient matrix. Belonging to the class of Krylov subspace methods, the GMRes algorithm projects the large linear system onto a much smaller subspace, where the approximate solution can be found by using standard solution techniques. It is normally used in conjunction with a preconditioner designed to improve the algebraic properties of the linear system. In solving our linear system, the bulk of the computations in the GMRes algorithm will involve Eqn structures only. These include the set of Krylov vectors (original and preconditioned) and the entries of the diagonal preconditioner. The computationally intensive task of Gramm-Schmidt orthogonalization of the Krylov vectors involves only on-processor operations and communication operations of the scan/reduce type. The only interplay between Elem and Eqn occurs when the matrix vector product is to be computed. Such a product involves a gather of Eqn-level values into the Elem data set, followed by a communication-free on-processor matrix-vector product, and finally a scatter back into the Eqn data set. The gather and scatter operations use the highly optimized Connection Machine Scientific Software Libraries available on the CM-200 and CM-5. In the current implementation, all variables related to the reduced system are stored on the scalar front-end processor, and the factorization and back substitution of that system are also performed in a scalar fashion.

Since they are memory intensive, the steady-state compressible-flow computations use a matrix-free version of the GMRes algorithm to eliminate even the need to store element-level matrices. Johan et al.[9] discuss the matrix-free GMRes implementation concepts.

The process for solving the global linear system operates on unassembled element-level matrices.

With one exception, so indicated, all performance measurements quoted in the next section are from either a 1,024-node CM-200 or a 512-node CM-5 equipped with vector execution units (VEUs). Note that "node" refers to either a Weitek floating-point unit on a CM-200 or to a four-VEU processing node on a CM-5. All computations are carried out in double (64-bit) precision, and all speeds reported are in gigaflops (billion floating-point operations per second). It is significant that while the communication libraries on the CM-200 were subjected to optimization efforts lasting several years, the development of their CM-5/VEU counterparts has just begun. Our results are based on a test version of the Thinking Machines Corp. software wherein the emphasis was on providing functionality and the tools necessary to begin testing the CM-5 vector units. This software release has not had the benefit of optimization or performance tuning; consequently, it does not necessarily represent the performance of the software's full version.

Numerical examples

Sloshing in a tank subjected to vertical vibrations. We performed a 3D study of liquid sloshing in a rigid container subjected to vertical vibrations. It is the continuation of a set of 2D computations inspired by a sloshing problem used to test the Arbitrary Lagrangian-Eulerian method.[11] In the 3D case, experimental and theoretical evidence[12] indicates the existence of multiple solution branches when the horizontal cross section of the tank is nearly square. Depending on the frequency of the vibrations, the competing wave modes interact, generating complex periodic — as well as chaotic — wave behavior. The case we consider here is based on the experiment performed by Feng and Sethna.[12]

The horizontal cross section of the tank is a

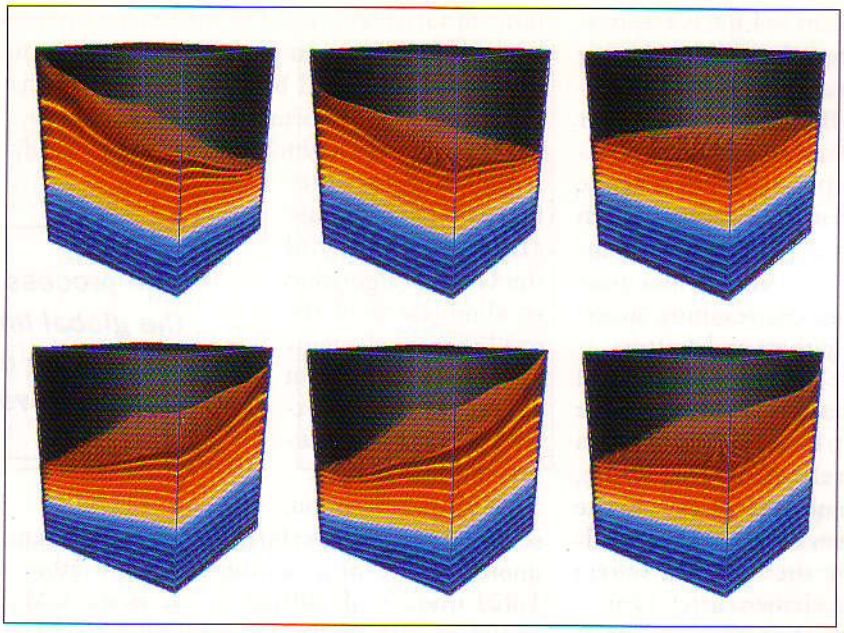


Figure 4. Three-dimensional sloshing in a tank. The tank is partially filled with water and subjected to vertical vibrations. The sequence of images (top left to right , then bottom left to right) shows the free surface and the pressure field. The number of nonlinear equations solved at each time step is 52,000 plus. The computation was carried out on the CM-200.

$W \times H$ rectangle, where $W = 0.178$ m (7.0 in.) and $H = 0.180$ m (7.1 in.). The water level, initially flat, is at $D = 0.127$ m (5.0 in.). Side and bottom surfaces of the tank allow slip in the direction tangent to the surface. The open surface of the water is assumed to be free from normal and shear stresses, and it moves with the normal component of the fluid velocity at the surface. The external forces acting on the fluid consist of a constant gravitational acceleration of magnitude $g = 9.81$ meters $\times$ (seconds)^2 and of a sinusoidal vertical excitation $Ag \sin \omega t$, with $\omega = 2\pi f$, $f = 4$ Hz, and A such that the amplitude of the oscillations remains at 1 millimeter. We select a time-step size to obtain 20 time steps per excitation period. The space-time mesh for each time slab consists of 14,112 nodes and 6,000 quadrilinear 4D elements.

In the GMRes solver, a Krylov space size of 40 was chosen, and the maximum number of outer GMRes iterations was five initially and 10 for larger fluid motions. At each time step, an average of three nonlinear iterations were performed.

On the CM-200 computer, formation of the element-level matrices and residuals took place at 0.80 gigaflops. We observed the speed of the GMRes solution phase as 0.87 Gflops. For a 3D space-time formulation, the bulk of time taken by the GMRes solution process is consumed by the matrix vector product. The entire code, including the parallel input/output operations and problem setup, performed at 0.79 Gflops. Figure 4 shows a sequence of images depicting the free surface and the pressure field.

Flow between two concentric cylinders (Taylor-Couette flow). The aim of this set of computations is to simulate the instabilities that develop between two concentric cylinders. The Reynolds number, a nondimensional ratio of the inertial effects to viscous effects, is based on the gap between the two cylinders and the speed of the inner cylinder; the outer cylinder is at rest. Beyond a certain critical Reynolds number, the regular Couette flow becomes unstable and we see the development of Taylor vortices. A further increase in the Reynolds number leads to an unsteady flow pattern — the wavy vortex flow.[13] The results we present here are for three different Reynolds numbers: 150, 250, and 1,498.

The finite-element mesh employed consists of 38,400 hexahedral elements and 45,024 nodes. The mesh contains six elements in the radial direction, 32 elements in the circumferential direction, and 200 elements in the axial direction. At each time step, a system of 282,366 nonlinear equations resulting from the finite-element discretization is solved iteratively using the GMRes technique with diagonal preconditioners. A Krylov space of dimension 30 is used. For this problem, the overall computation speed is 2.1 Gflops on the CM-200 and 3.6 Gflops on the CM-5. This timing excludes the input/output phase of the computations. The formation speed of the element-level matrices and the right-hand-side vector is 3.0 Gflops, while the speed for the GMRes part is 1.4 Gflops on the CM-200. The corresponding speeds on the CM-5 are 5.2 Gflops and 2.6 Gflops, respectively.

As boundary conditions, at the upper and lower boundaries, the axial component of the velocity and the x and y components of the stress vector are set to zero (the z axis lies in the axial direction).

Reynolds number = 150: Taylor vortex flow. This value of the Reynolds number is greater than the critical Reynolds number. Thus, for certain disturbances, one would expect the Couette flow to develop instabilities. We have an

interesting observation related to this flow. When the solution is computed with no external disturbances, a stable Couette flow is observed. However, if the solution is obtained with an initial condition that corresponds to an unsteady solution from a higher Reynolds number, a Taylor vortex flow is realized. Figure 5 (top) shows the axial velocity at the horizontal and vertical sections and the cylindrical section midway between the inner and outer cylinders. We observe that the solution at this Reynolds number is axisymmetric.

Reynolds number = 250: Wavy vortex flow. At Reynolds number 250, our computations reveal the presence of a wavy vortex flow; the Taylor vortex flow discussed above is itself unstable, and the solution is no longer axisymmetric. Figure 5 (center) shows the axial velocity at the horizontal and vertical sections and the cylindrical section midway between the inner and outer cylinders for a nearly temporally periodic solution. In addition to the cells in the axial direction, there are four waves traveling in the azimuthal direction.

Reynolds number = 1,498: Wavy vortex flow. At Reynolds number 1,498, we again observe the wavy vortex flow. Compared with the solution at Reynolds number 250, the vortices for this Reynolds number are much stronger. In this case, there are three waves traveling in the azimuthal direction. Figure 5 (bottom) shows the axial velocity at the horizontal and vertical sections and the cylindrical section midway between the inner and outer cylinders for a nearly temporally periodic solution.

Supersonic flow past a delta wing at Mach 3. In this air-flow problem, the angle of attack is 0 degrees and the Reynolds number, based on the free-stream values and the maximum chord length (along the plane of symmetry), is 1.1 million. The finite-element formulation used to solve this problem is in conservation variables.[4] Because of the assumed symmetry of the problem with respect to the $z = 0$ plane, only half of the domain is considered; but for better visualization, ghost nodes are created by a simple reflection and the results are presented over the entire domain. Chien Li from the NASA Johnson Space Center provided the geometry of the delta wing.

The delta wing has a wedge-type cross section. Its underbody merges smoothly with the flat surface on the top. The finite-element discretization is carried out using 143,920 trilinear hexahedral elements, with one integration point per element. To capture the details of the boundary layers, the first three layers of the elements are kept very close to the delta wing. At each time step, 725,688 nonlinear equations are solved simultaneously using a matrix-free GMRes search technique. This problem was solved on the CM-5 at a sustained speed of 10.4 Gflops. The front, side, and top views of the delta wing, and the Mach number distribution are shown in Figure 6.

Beyond a certain critical Reynolds number, we see the development of Taylor vortices.

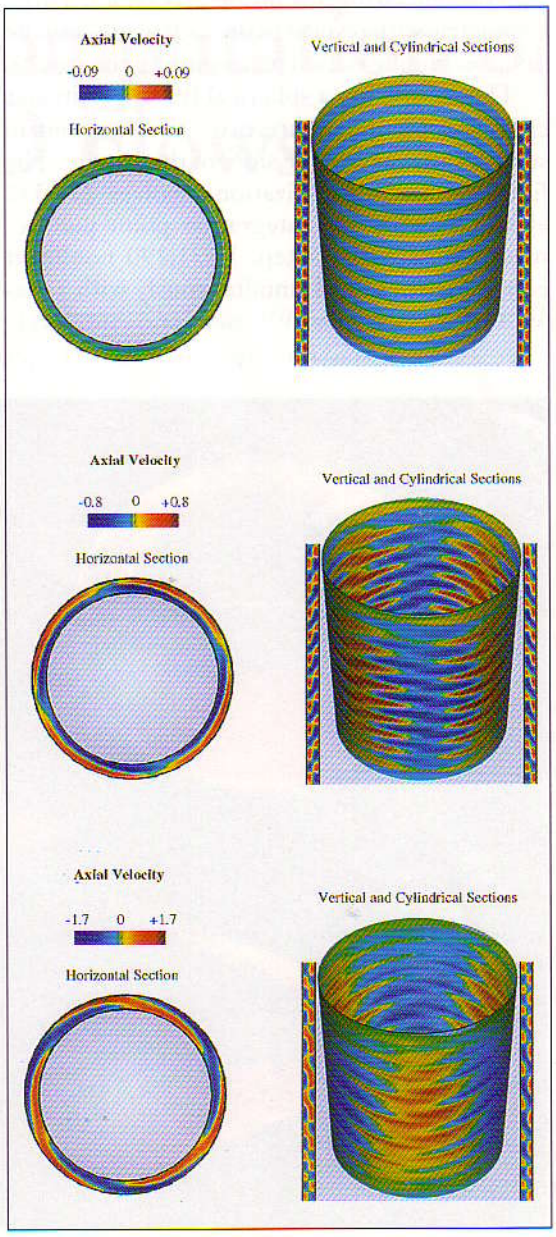


Figure 5. Three-dimensional incompressible flow between two concentric cylinders at Reynolds numbers 150 (top), 250 (center), and 1,498 (bottom). The images show the axial velocity at horizontal, vertical, and cylindrical sections. The number of nonlinear equations solved at each time step is 282,000 plus. The computations were carried out on the CM-5.

Table 1. Performance in gigaflops for implicit incompressible-flow implementations.

	Taylor-Couette Flow 282,366 Equations		Flow Past a Sphere 649,630 Equations		Flow Past a Sphere 1,052,216 Equations	
	CM-200	CM-5	CM-200	CM-5	CM-200	CM-5
Element computation	3.0	5.2	1.1	8.2	—	9.6
GMRes solution	1.4	2.6	1.3	3.1	—	2.9
Sustained	2.1	3.6	1.2	4.6	—	4.6

Supersonic flow past a toy missile at Mach 3.25. Our compressible finite-element formulation is also used to solve Mach 3.25 flow past a toy missile at a 0.5-degree angle of attack. The Reynolds number, based on the free-stream values and the length of the missile, is 110,000. As with the delta wing, the problem is assumed to be symmetric with respect to the $z = 0$ plane, and the same technique is used in presenting the results.

The missile has a spherical tip and a circular cross section. There are two "V"-type control surfaces attached to the sides of the missile. The finite-element discretization involves 224,332 elements, with one integration point per element. At each time step, 1,121,290 nonlinear equations are solved simultaneously with a matrix-free GMRes search technique. This problem was solved on the CM-5 at a sustained speed of 9.5 Gflops. The front, side, and top views of the missile, and the Mach number distribution are shown in Figure 7.

A benchmark computation: Flow past a sphere. The space-time formulation was also used to solve a benchmark problem involving incompressible flow past a sphere at Reynolds number 100. The problem was solved using a range of meshes that differ in refinement; the timings from the two largest meshes are reported here. For the mesh with 79,932 elements, 649,630 nonlinear equations were solved at each time step, while for the mesh with 129,736 ele-

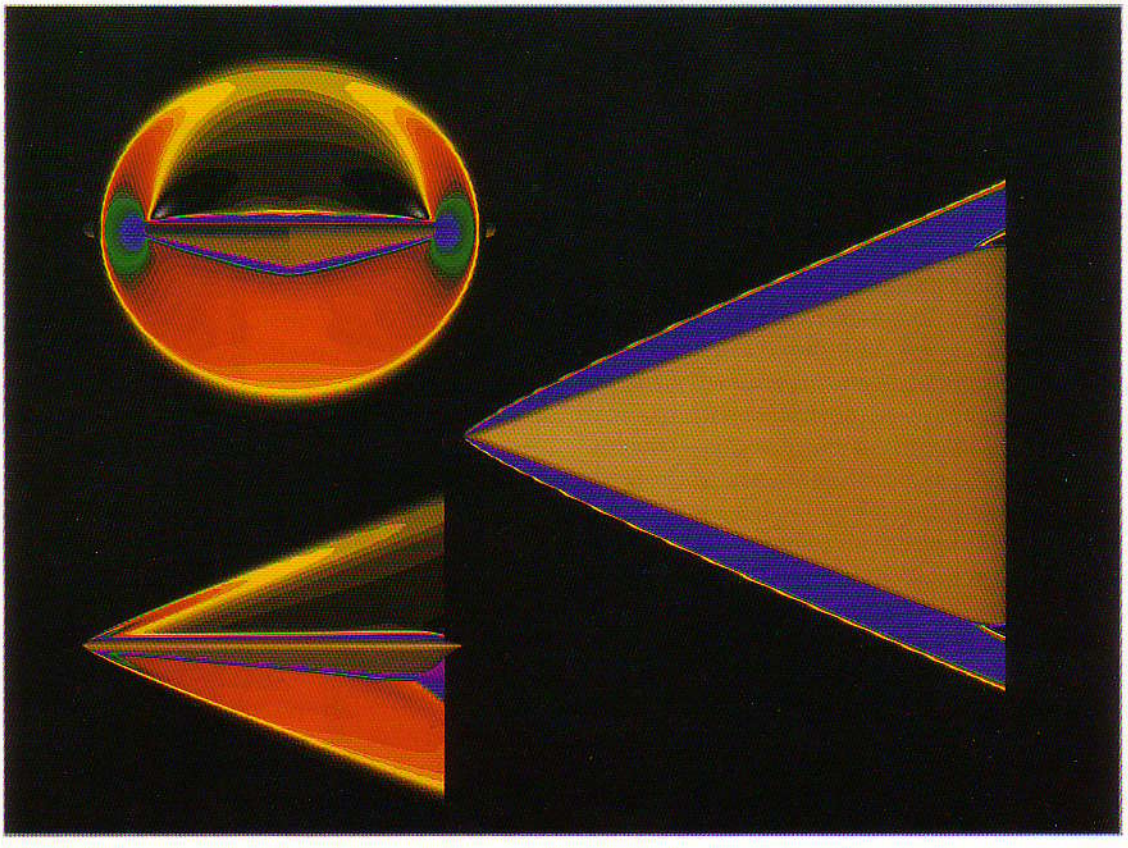


Figure 6. Three-dimensional supersonic flow past a delta wing at Mach 3. The Reynolds number is 1.1 million. The images show the front, side, and top views of the wing, and the Mach number distribution. The number of nonlinear equations solved at each time step is 725,000 plus. The computation was carried out on the CM-5 with a sustained speed of 10.4 gigaflops.

ments, the number of nonlinear equations solved was 1,052,216. In both cases the GMRes parameters included a Krylov space size of 20, and five outer iterations (restarts). For the smaller mesh the computation was possible on both the CM-200 and the CM-5. On the CM-200, the computational rates for the matrix formation and the GMRes solution stages were 1.1 and 1.3 Gflops, respectively, and the overall rate was 1.2 Gflops; for the CM-5 these numbers were 8.2, 3.1, and 4.6 Gflops. Computation with the larger mesh was not possible on the CM-200 with a nonmatrix-free implementation because of memory limitations. On the CM-5, the performance was measured at 9.6, 2.9, and 4.6 Gflops for the matrix formation, GMRes solution, and total speed, respectively.

It is apparent that the speed of the matrix formation stage is still benefiting from the increase in the subgrid length from around 40 elements per vector unit for the smaller mesh to more than 60 for the larger one. On the other hand, the speed of the GMRes solver is approximately the same for the two meshes.

Tables 1 and 2 summarize the parallel performance observed in computing the 3D problems described above.

In a recent computation of the delta wing problem, we used 1,002,684 elements, with eight integration points per element. The number of equations in this case was 5,001,031. The sustained performance in this computation was measured at 17.2 Gflops on the CM-5 with 512 processing nodes. On a CM-5 with 1,024 processing nodes, located at Los Alamos National Laboratory, sustained performance was measured at 37.5 Gflops.

Table 2. Performance in gigaflops for matrix-free implicit compressible-flow implementations.

	Flow Past a Delta Wing 725,688 Equations CM-5	Flow Past a Toy Missile 1,121,290 Equations CM-5
Element computation	15.4	17.1
Sustained	10.4	9.5

The type of equation systems considered here can occur in electrical, chemical, and civil engineering as well. Thus, the strategies we used may serve a broad range of applications, and comparable parallel performance can be expected.

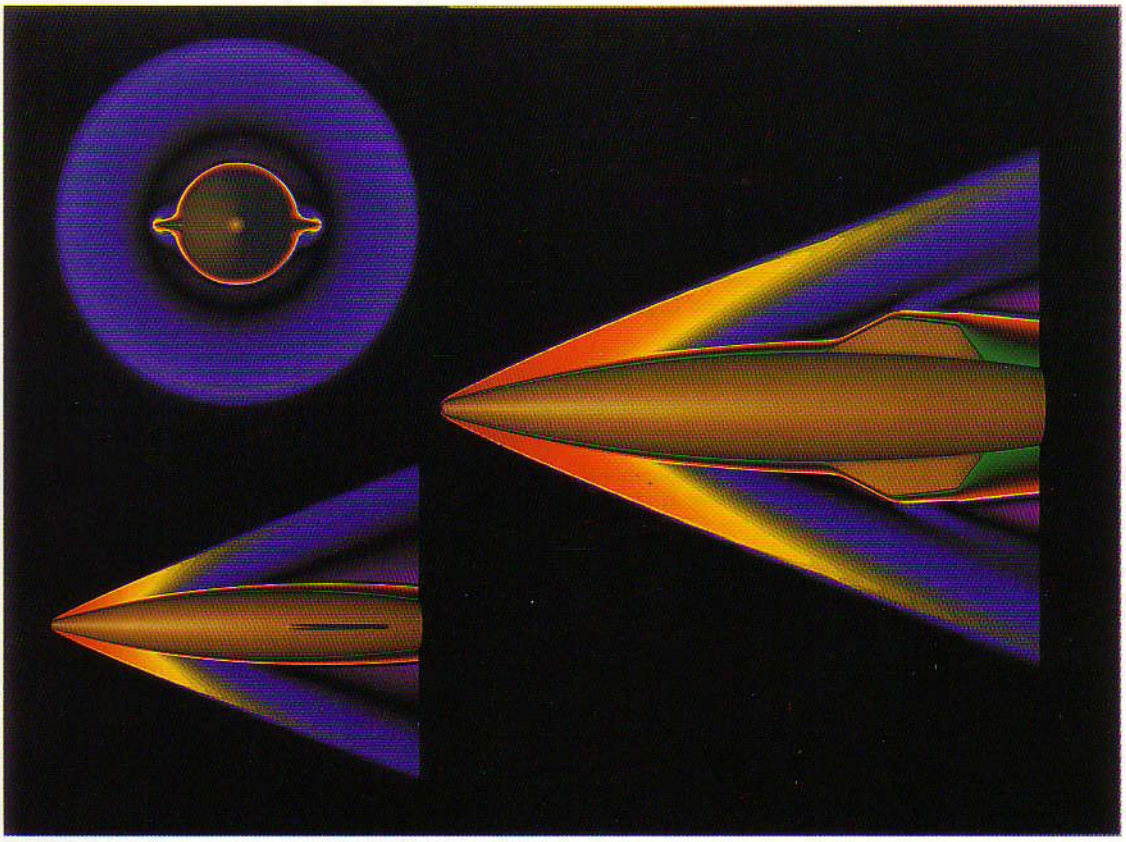


Figure 7. Three-dimensional supersonic flow past a toy missile at Mach 3.25. The Reynolds number is 110,000. The images show the front, side, and top views of the missile, and the Mach number distribution. The number of nonlinear equations solved at each time step is 1.1 million plus. The computation was carried out on the CM-5 with a sustained speed of 9.5 gigaflops.

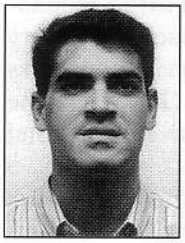
Acknowledgments

This research was sponsored by NASA-JSC under grant NAG 9-449, by the National Science Foundation under grants MSM-8796352 and ASC-9211083, and by ARPA under NIST Contract 60NANB2D1272. Partial support for this work also came from the Army Research Office Contract Number DAAL03-89-C-0038 with the Army High-Performance Computing Research Center at the University of Minnesota. We thank Thinking Machines Corp., in particular John Kennedy, for the technical support we received. We also thank the staff of the Department of Energy High-Performance Computing Research Center at Los Alamos National Laboratory.

References

1. T.J.R. Hughes and A.N. Brooks, "A Multidimensional Upwind Scheme with No Crosswind Diffusion," in *Finite-Element Methods for Convection-Dominated Flows*, T.J.R. Hughes, ed., AMD, Vol. 34, American Soc. Mechanical Engineers, New York, 1979, pp. 19-35.
2. T.E. Tezduyar and T.J.R. Hughes, "Finite-Element Formulations for Convection-Dominated Flows with Particular Emphasis on the Compressible Euler Equations," AIAA Paper 83-0125, *Proc. AIAA 21st Aerospace Sciences Meeting*, American Inst. of Aeronautics and Astronautics, New York, 1983.
3. T.J.R. Hughes, L.P. Franca, and M. Mallet, "A New Finite-Element Formulation for Computational Fluid Dynamics: VI. Convergence Analysis of the Generalized SUPG Formulation for Linear Time-Dependent Multidimensional Advective-Diffusive Systems," *Computer Methods in Applied Mechanics and Eng.*, Vol. 63, 1987, pp. 97-112.
4. G.J. LeBeau and T.E. Tezduyar, "Finite-Element Computation of Compressible Flows with the SUPG Formulation," in *Advances in Finite-Element Analysis in Fluid Dynamics*, FED-Vol. 123, M.N. Dhaubhadel, M.S. Engelman, and J.N. Reddy, eds., American Soc. Mechanical Engineers, New York, 1991, pp. 21-27.
5. T.E. Tezduyar, M. Behr, and J. Liou, "A New Strategy for Finite-Element Computations Involving Moving Boundaries and Interfaces — The Deforming-Spatial-Domain/Space-Time Procedure: I. The Concept and the Preliminary Tests," *Computer Methods in Applied Mechanics and Eng.*, Vol. 94, 1992, pp. 339-351.
6. S. Mittal and T.E. Tezduyar, "A Finite-Element Study of Incompressible Flows Past Oscillating Cylinders and Airfoils," *Int'l J. Numerical Methods in Fluids*, Vol. 15, 1992, pp. 1,073-1,118.
7. T.E. Tezduyar et al., "Computation of Unsteady Incompressible Flows with the Finite-Element Methods — Space-Time Formulations, Iterative Strategies, and Massively Parallel Implementations," in *New Methods in Transient Analysis*, P. Smolinski et al., eds., AMD, Vol. 143, American Soc. Mechanical Engineers, New York, 1992, pp. 7-24.
8. T.E. Tezduyar et al., "Massively Parallel Finite-Element Computation of Three-Dimensional Flow Problems," *Proc. Sixth Japan Numerical Fluid Dynamics Symp.*, Chuo University, Tokyo, Japan, 1992, pp. 15-24.
9. Z. Johan et al., "A Data-Parallel Finite-Element Method for Computational Fluid Dynamics on the Connection Machine System," *Computer Methods in Applied Mechanics and Eng.*, Vol. 99, 1992, pp. 113-134.
10. M. Behr et al., "Computation of Incompressible Flows with Implicit Finite-Element Implementations on the Connection Machine," to appear in *Computer Methods in Applied Mechanics and Eng.*, 1993. See update U1 below.
11. A. Huerta and W.K. Liu, "Viscous Flow with Large Free-Surface Motion," *Computer Methods in Applied Mechanics and Eng.*, Vol. 69, 1988, pp. 277-324.
12. Z.C. Feng and P.R. Sethna, "Symmetry-Breaking Bifurcations in Resonant Surface Waves," *J. Fluid Mechanics*, Vol. 199, 1989, pp. 495-518.
13. D. Coles, "Transition in Circular Couette Flow," *J. Fluid Mechanics*, Vol. 21, 1965, pp. 385-425.

Tayfun Tezduyar is a professor in the Department of Aerospace Engineering and Mechanics at the University of Minnesota, Minneapolis, Minnesota. His research interests are high-performance computing, finite-element analysis, and computational fluid dynamics. Tezduyar received his MS and PhD degrees in mechanical engineering from the California Institute of Technology in 1978 and 1982, respectively.



Shahrouz Aliabadi is a PhD student in the Department of Aerospace Engineering and Mechanics at the University of Minnesota. His research interests are high-performance computing and finite elements in fluids. Aliabadi received his BS in aeronautical engineering from the Middle East Technical University, Ankara, Turkey, in 1989.

Marek Behr is a postdoctoral fellow at the University of Minnesota Army High-Performance Computing Research Center, Minneapolis, Minnesota. His research interests are high-performance computing and finite elements in fluids. Behr received his BS and PhD degrees in aerospace engineering and mechanics from the University of Minnesota in 1988 and 1992, respectively.

Andrew Johnson is a PhD student in the Department of Aerospace Engineering and Mechanics at the University of Minnesota. His research interests are high-performance computing and finite elements in fluids. Johnson received his BS and MS degrees in aerospace engineering and mechanics from the University of Minnesota in 1990 and 1991, respectively.

Sanjay Mittal is a postdoctoral fellow at the University of Minnesota Army High-Performance Computing Research Center, Minneapolis, Minnesota. His research interests are high-performance computing and finite elements in fluids. Mittal received his BS in aeronautical engineering from the Indian Institute of Technology, Kanpur, India, in 1988, and MS and PhD degrees in aerospace engineering and mechanics from the University of Minnesota in 1990 and 1992, respectively.

Readers may contact Tezduyar at 107 Akerman Hall, University of Minnesota, Minneapolis, MN 55455; e-mail tezduyar@aem.umn.edu.

U1. M. Behr, A. Johnson, J. Kennedy, S. Mittal and T. Tezduyar, "Computation of Incompressible Flows with Implicit Finite Element Implementations on the Connection Machine", *Computer Methods in Applied Mechanics and Engineering*, **108** (1993) 99-118.