JOURNAL OF THERMOPHYSICS AND HEAT TRANSFER
Vol. 9, No. 2, April–June 1995

Parallel Finite Element Algorithm for Three-Dimensional Inviscid and Viscous Flow

Frank P. Brueckner*
Hughes Missile Systems Company, Tucson, Arizona 85734
and
Darrell W. Pepper†
University of Nevada, Las Vegas, Las Vegas, Nevada 89154

A finite element model is developed and used to simulate three-dimensional compressible fluid flow on a massively parallel computer. The algorithm is based on a Petrov-Galerkin weighting of the convective terms in the governing equations. The discretized time-dependent equations are solved explicitly using a second-order Runge-Kutta scheme. A high degree of parallelism has been achieved utilizing a MasPar MP-2 SIMD computer. An automated conversion program is used to translate the original Fortran 77 code into the Fortran 90 needed for parallelization. This conversion program and the use of compiler directives allows the maintenance of one version of the code for use on either vector or parallel machines.

Nomenclature

c_p = specific heat at constant pressure
c_v = specific heat at constant volume
E = total energy per unit mass
e = specific internal energy
I = identity tensor
k = thermal conductivity
L = reference length
M = mass matrix
M_l = lumped mass matrix
M_∞ = freestream Mach number, $U_\infty/\sqrt{\gamma p_\infty/\rho_\infty}$
N^i = shape function corresponding to node i
$\boldsymbol{n}$ = unit vector normal to boundary
Pr = Prandtl number, $\bar{\mu}c_p/\bar{k}$
p = pressure
$\boldsymbol{R}$ = right-hand-side vector
Re_∞ = freestream Reynolds number, $\rho_\infty U_\infty L/\mu_\infty$
S = Sutherland constant
T = temperature
t = time
U_∞ = freestream velocity
u, v, w = velocity components
$\boldsymbol{u}$ = velocity vector
v_j^i = weighting function for equation j corresponding to node i
v_p, v_u, v_e = weighting functions
x, y, z = Cartesian coordinates
$\boldsymbol{x}$ = position vector
α_j = weighting function parameter for equation j
Γ = boundary of flow domain
γ = ratio of specific heats, c_p/c_v
γ_j = mesh Peclet number for equation j
Δt = timestep
μ = viscosity
ρ = fluid density
τ = deviatoric stress tensor
Φ = solution vector
Ω = flow domain

Subscripts
c = corrected value
e = element quantity
inlet = condition at inlet of nozzle
outlet = condition at outlet of nozzle
p = predicted value
s = stagnation quantity
ξ, η, ζ = element coordinates
∞ = freestream quantity

Superscripts
h = numerical approximation
m = number of timesteps
T = transpose
$\bar{\ }$ = dimensional quantity

Presented as Paper 93-0340 at the AIAA 31st Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 11–14, 1993; received Feb. 8, 1994; revision received Sept. 27, 1994; accepted for publication Sept. 27, 1994.

*Senior Engineering Specialist. Member AIAA.
†Associate Professor, Department of Mechanical Engineering. Associate Fellow AIAA.

Introduction

LARGE computational fluid dynamics (CFD) problems have traditionally been solved using powerful scalar or vector pipelined computers. With the recent developments in software tools and programming environments for massively parallel computers, it is now possible to develop applications that can exploit the computing power of massively parallel architectures with a reasonable program development time. Finite element methods, particularly those that use explicit solution schemes, map very well to massively parallel architectures.

In this article, a finite element algorithm for the numerical simulation of compressible flow is described. Elements are defined using trilinear velocity, density, and energy, with a piecewise constant pressure. An anisotropic balancing diffusion is used via a Petrov-Galerkin weighting. The original vector code is ported onto the MasPar MP-2 family of parallel computers; a simple map of one element per processor in each memory layer is used. CPU timing is obtained on the MasPar MP 2216 (16,384 processors) and compared to the Cray Y-MP (single processor) and the Alliant FX/40.

Governing Equations

The conservation form of the equations that describe the transport of compressible fluid are written in nondimensional

vector form as

Mass (continuity)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{1}$$

Momentum

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot \left(\rho \boldsymbol{u}\boldsymbol{u} + \frac{1}{\gamma M_\infty^2} pI \right) = \nabla \cdot \tau \tag{2}$$

Energy

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [\rho \boldsymbol{u} E + (\gamma - 1) p \boldsymbol{u}] = \nabla \cdot \left[\frac{\gamma}{PrRe_\infty} \mu \nabla e + \gamma(\gamma - 1) M_\infty^2 (\tau \cdot \boldsymbol{u}) \right] \tag{3}$$

The total energy

$$E = e + [(\gamma - 1)/2] \gamma M_\infty^2 (\boldsymbol{u} \cdot \boldsymbol{u}) \tag{4}$$

is the sum of the specific internal e and the kinetic energy per unit mass. The equations have been nondimensionalized to emphasize the relative importance of terms. The nondimensional variables are defined as

$$\begin{aligned} \nabla &= L\bar{\nabla}, & \boldsymbol{u} &= \frac{\bar{\boldsymbol{u}}}{U_\infty}, & \rho &= \frac{\bar{\rho}}{\rho_\infty} \\ \mu &= \frac{\bar{\mu}}{\mu_\infty}, & k &= \frac{\bar{k}}{k_\infty}, & T &= \frac{\bar{T}}{T_\infty} \\ p &= \frac{\bar{p}}{p_\infty}, & e &= \frac{\bar{e}}{c_v T_\infty}, & t &= \frac{\bar{t}}{L/U_\infty} \end{aligned} \tag{5}$$

The dimensionless deviatoric stress tensor τ is

$$\tau = \mu / Re_\infty \, [(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) - \tfrac{2}{3} I (\nabla \cdot \boldsymbol{u})] \tag{6}$$

The kinematic viscosity μ is approximated using Sutherland's formula

$$\mu = T^{3/2}[(1 + S)/(T + S)] \tag{7}$$

in which

$$S = \bar{S}/T_\infty \tag{8}$$

is the dimensionless Sutherland constant. To close this system of equations, a thermally and calorically perfect gas is assumed, leading to the equation of state

$$p = \rho T \tag{9}$$

$$T = e \tag{10}$$

Finally, the Euler equations are derived by letting $1/Re_\infty \to 0$, thereby forcing the right-hand-sides of Eqs. (2) and (3) to vanish.

Finite Element Method

It is well known that the use of equal-order shape functions for the velocity components and the pressure in incompressible flow problems can lead to poor, if not meaningless, solutions.[1] Such behavior is also possible in low subsonic regions of compressible flow. Averting this condition in nearly incompressible regions of compressible flow problems has been an active area of research for some time.[2-4] The algorithm outlined below is developed to accurately simulate these low-speed flow regimes, as well as regions in which compressibility effects are significant.

Weak Formulation and Discretization

The governing equations [Eqs. (1-3)] are rewritten as transport equations for ρ, $\boldsymbol{u}$, and e.[5] The weak forms of this system of equations are derived via the Petrov-Galerkin weighted residual method as

$$\int_\Omega v_\rho \left(\frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho + \rho \nabla \cdot \boldsymbol{u} \right) d\Omega = 0 \tag{11}$$

$$\int_\Omega \left\{ \boldsymbol{v}_u \cdot \left[\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) \right] - \frac{1}{\gamma M_\infty^2} p \nabla \cdot \boldsymbol{v}_u + \nabla \boldsymbol{v}_u : \tau \right\} d\Omega - \int_\Gamma \boldsymbol{v}_u \cdot \left[-\frac{1}{\gamma M_\infty^2} p \boldsymbol{n} + \boldsymbol{n} \cdot \tau \right] d\Gamma = 0 \tag{12}$$

$$\int_\Omega \left\{ v_e \left[\rho \left(\frac{\partial e}{\partial t} + \boldsymbol{u} \cdot \nabla e \right) + (\gamma - 1) p \nabla \cdot \boldsymbol{u} - \gamma(\gamma - 1) M_\infty^2 (\tau : \nabla \boldsymbol{u}) \right] + \frac{\gamma}{PrRe_\infty} \mu \nabla v_e \cdot \nabla T \right\} d\Omega - \int_\Gamma v_e \left[\frac{\gamma}{PrRe_\infty} \mu \boldsymbol{n} \cdot \nabla T \right] d\Gamma = 0 \tag{13}$$

The boundary integrals in Eqs. (12) and (13) arise from the application of Green's identity to the respective flux terms. In addition, the pressure gradient in the momentum equation has also been integrated by parts. Removing all pressure derivatives from these weak forms allows the pressure to be constant within each element. This piecewise constant approximation to the pressure improves the behavior of the algorithm in regions of the flow where the local Mach number is very low, but without degrading the numerical solution at transonic and supersonic Mach numbers.

The density, velocity components, and internal energy are now approximated by the following expressions:

$$\begin{aligned} \rho(\boldsymbol{x}, t) &\approx \rho^h(\boldsymbol{x}, t) = \sum_{i=1}^{n} N^i(\boldsymbol{x}) \rho_i(t) \\ \boldsymbol{u}(\boldsymbol{x}, t) &\approx \boldsymbol{u}^h(\boldsymbol{x}, t) = \sum_{i=1}^{n} N^i(\boldsymbol{x}) \boldsymbol{u}_i(t) \\ e(\boldsymbol{x}, t) &\approx e^h(\boldsymbol{x}, t) = \sum_{i=1}^{n} N^i(\boldsymbol{x}) e_i(t) \end{aligned} \tag{14}$$

utilizing standard trilinear isoparametric shape functions. As discussed above, the pressure is approximated as a constant within each element; nodal values are interpolated from these element quantities when required for postprocessing.

To stabilize the discretized equations, an anisotropic balancing diffusion is introduced into each governing equation via a Petrov-Galerkin weighting function.[6,7] These weights are obtained by perturbing the shape functions N^i such that term i of the v_j weight ($j = \rho, u, v, w, e$) is given by

$$v_j^i = N^i + \alpha_j (h_e / 2|\boldsymbol{u}|)(\boldsymbol{u} \cdot \nabla N^i) \tag{15}$$

The calculation of h_e utilizes the element geometry and three mesh length vectors. For the mesh length along $\boldsymbol{u}$, these vectors, which intersect the element at the midpoints of the sides, are projected in the direction of the local velocity to obtain

$$h_e = (1/|\boldsymbol{u}|)(|\boldsymbol{h}_\xi \cdot \boldsymbol{u}| + |\boldsymbol{h}_\eta \cdot \boldsymbol{u}| + |\boldsymbol{h}_\zeta \cdot \boldsymbol{u}|) \tag{16}$$

The parameter α_j is defined as[5,8]

$$\alpha_j = \coth(\gamma_j/2) - (2/\gamma_j) \tag{17}$$

in which

$$\gamma_j = \begin{cases} \infty, & \text{for } j = \rho \\ \rho Re_\infty |\boldsymbol{u}| h_e/\mu, & \text{for } j = u, v, w \\ \rho Pr Re_\infty |\boldsymbol{u}| h_e/\mu\gamma, & \text{for } j = e \end{cases} \tag{18}$$

In this work, the anisotropic balancing diffusion approach of Kelly et al.[6] and Yu and Heinrich[9] is adopted. In doing so, the functions v_ρ, v_u, and v_e are used to weight the advection terms in Eqs. (11), (12), and (13), respectively, while the remaining terms are weighted by the shape functions N^i. The effect of this weighting is to introduce a diffusion along the streamline in each transport equation. The precise amount of artificial diffusion and the direction in which it must be added for optimizing accuracy is calculated locally for each element.

In the Euler limit $1/Re_\infty \rightarrow 0$, the Petrov-Galerkin weighting functions become the same for each equation, i.e.,

$$v^i = N^i + (h_e/2|\boldsymbol{u}|)(\boldsymbol{u}\cdot\nabla N^i) \tag{19}$$

Finally, the spatial integrations over Ω are computed numerically using a $2 \times 2 \times 2$ Gaussian quadrature in the local element coordinate system. To handle regions where the velocity and the divergence of the velocity are small, reduced integration (i.e., one-point) is used for the pressure terms in the momentum equations and the velocity divergence terms in the continuity and energy equations.

Temporal Integration

The formulation of the previous sections yields a system of equations of the form

$$M\frac{d\Phi}{dt} = \boldsymbol{R} \tag{20}$$

To produce a fully explicit algorithm, the mass matrix is diagonalized by employing lumped mass approximations.[10]

An explicit second-order Runge-Kutta method is used to advance the discretized equations in time. With the lumped mass matrix denoted by M_l, the Runge-Kutta scheme is defined by the following two-step algorithm:

$$\begin{aligned} \Phi^{m+1/2} &= \Phi^m + (\Delta t/2)[M_l^{-1}\boldsymbol{R}]^m \\ \Phi^{m+1} &= \Phi^m + \Delta t[M_l^{-1}\boldsymbol{R}]^{m+1/2} \end{aligned} \tag{21}$$

where the superscript m indicates an expression evaluated at time $m\Delta t$, Δt being the magnitude of the timestep. Since the discretized equations are solved explicitly, a stability constraint on the timestep must be imposed. The Courant limits associated with a forward Euler scheme are calculated over each element, and the timestep adjusted to the minimum value within the computational domain.

Boundary and Initial Conditions

In this work, the primary boundary conditions associated with Eqs. (11–13) are of two types: 1) solid walls and 2) inflow/outflow.

Solid Walls

For the Navier-Stokes equations, a no-slip condition is imposed along solid walls. Either a heat flux or a specified temperature may be imposed as boundary conditions for the energy equation. If no quantity is specified, the homogeneous natural boundary condition of an adiabatic wall is modeled. These conditions may be written as

$$\boldsymbol{u} = 0 \tag{22}$$

$$\nabla T\cdot\boldsymbol{n} = 0 \tag{23}$$

at the boundaries.

In the Euler limit, solid walls are incorporated by restricting the nodal velocity vector to be tangent to the domain boundary. Using a predictor-corrector approach, the expression

$$\boldsymbol{u}_c = \boldsymbol{n} \times (\boldsymbol{u}_p \times \boldsymbol{n}) \tag{24}$$

is applied at each time step to enforce this tangency condition. Here, the subscripts c and p denote corrected and predicted values, respectively.

Inflow/Outflow

Inflow/outflow boundary conditions are implemented based on the nature of the boundary at each time step; i.e., inflow or outflow, subsonic or supersonic.

Supersonic Inflow

All quantities are specified for both the Euler and Navier-Stokes systems of equations.

Subsonic Inflow

As discussed in Rudy and Strikwerda,[11] the specification of all quantities at a subsonic inflow boundary overspecifies the inflow data in the sense that the Navier-Stokes system of partial differential equations requires only three conditions. Therefore, only the density and velocity components are specified for a subsonic inflow. A homogeneous natural boundary condition is used for the internal energy equation that effectively results in no heat flux across the boundary.

The Euler equations are treated in a similar manner, with only the density and velocity components specified.

Supersonic Outflow

For both the Navier-Stokes and the Euler equations, the boundary integrals in the momentum equations are computed using previously calculated quantities for the integrands. Since piecewise constant pressure elements are employed, a single, centroidally located Gauss point is used for the integration over each element boundary face.

Subsonic Outflow

The choice of subsonic outflow conditions has a significant influence on both the accuracy of the solution and the convergence characteristics. In general, the pressure at the outflow boundary needs to be specified. This specified pressure is used to compute the contributions from the boundary integrals in the momentum equations.

Finally, the initial conditions are taken uniformly as the freestream values.

MasPar Architecture

MasPar manufactures a family of massively parallel computer systems capable of attaining peak processing speeds up to 68 KMIPS (32-bit integer adds) and 6.3 GFLOPS (32-bit floating point adds and multiplies). The MP-2 family of computers obtains its performance by using an array of processing elements (PEs). The PE array consists of 1024 to 16,384 processors that operate in a single-instruction multiple-data (SIMD) fashion. There are three major components to the machine[12]: 1) the PE array, 2) the array control unit (ACU), and 3) the UNIX subsystem. Computational power is attained by using a massively parallel array of PEs. Architecturally, each PE is a RISC processor with a 64-bit wide accumulator, 48 32-bit registers, and 64 KBytes of data memory. All PEs execute

instructions in lock step on data stored in its local memory. Each PE can enable or disable itself for part or all of a computation based on a logical expression for conditional execution.

To share data with other PEs, there are three communication mechanisms available: 1) the Xnet, 2) the router, and 3) the global-or tree. Xnet, or eight-way nearest neighbor communication, provides a very fast path for moving data between a PE and its eight neighbors. Xnet provides an aggregate bandwidth of up to 20 GBytes/s (16K PEs). In addition to Xnet, the MP-2 has an alternate, multistage, circuit-switched network for global or random communication patterns. This network router provides a PE the ability to send or fetch data from any other PE in the array. The aggregate bandwidth of the router communications is 1.3 GBytes/s (16K PEs). A global-or tree is used for moving data from the PE array into the ACU.

The ACU performs two functions: execution control and scalar computations, and broadcasting instructions and/or data to the PE array. The ACU is the master and controls all the processing in the MP-2 computer. Programs are written to control the ACU, and hence, the PE array. The UNIX subsystem provides application engineers with a programming and run-time environment.

Porting onto the MP-2

The initial implementation of the finite element code was highly optimized for efficient execution on vector machines such as the Cray Y-MP. To execute this code on the massively parallel MasPar MP-2, the existing Fortran 77 was converted to Fortran 90 or MasPar Fortran.[13] MasPar Fortran is based on Fortran 90 and includes expressions and intrinsics that allow the compiler to map data structures onto the processor array. MasPar Fortran also accesses the system's different functional units, as required, and integrates all communications and I/O operations. The conversion process to MasPar Fortran can be divided into the following tasks:

Converting F77 to F90

A substantial portion of the conversion effort consists of replacing Fortran 77 DO loops with Fortran 90 array syntax. An automated conversion program, VAST-2,[14] was used to accomplish this task. In this application, most of the computationally intensive DO loops are either the number of elements or number of nodes long. On an MP 2216 (16,384 PEs), an instruction is executed using 16,384 different data simultaneously.

Another significant conversion done by VAST-2 was to translate a condition statement (i.e., IF statement) into a WHERE construct in Fortran 90.

Storing Arrays in PE's Memory

To maximize the execution efficiency, all arrays must be stored in the PE's memory. The arrays in this implementation are declared in COMMON blocks and are specified in an include file. The MasPar Fortran ONDPU directive stores either COMMONs or variables in PE memory.

Storing Two-Dimensional Arrays

Two-dimensional arrays could be stored in a cut-and-stack fashion using MasPar Fortran. This storage scheme, however, would not produce the most efficient algorithm. To maximize performance, the MasPar Fortran MAP directive is inserted in the routines that use two-dimensional arrays. For example, the directive maps the first dimension of the array DEL(NELM,8) onto the PE array. The second dimension of DEL is mapped in the PE's memory. In this way, the MAP directive effectively converts a two-dimensional array into a number of one-dimensional arrays; DEL becomes eight one-dimensional arrays, each of which is NELM long.

FORTRAN 77

```
      DO 500 I=1,NELM
        B1(NODE(I)) = B1(NODE(I)) + W1(I)
        B2(NODE(I)) = B2(NODE(I)) + W2(I)
        B3(NODE(I)) = B3(NODE(I)) + W3(I)
  500 CONTINUE
```

FORTRAN 90

```
CMPF COLLISIONS
      B1(NODE(:NELM)) = B1(NODE(:NELM))+W1(:NELM)
CMPF COLLISIONS
      B2(NODE(:NELM)) = B2(NODE(:NELM))+W2(:NELM)
CMPF COLLISIONS
      B3(NODE(:NELM)) = B3(NODE(:NELM))+W3(:NELM)
```

Fig. 1 MasPar COLLISIONS directive.

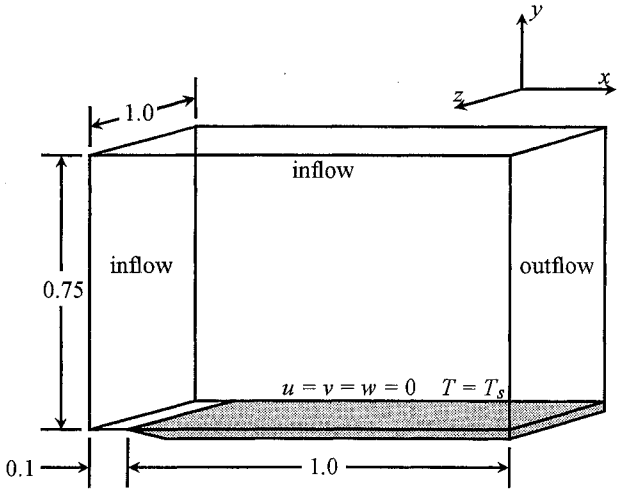

Fig. 2 Geometry and boundary conditions for supersonic flow past a flat plate.

Managing Vector Dependencies

For a general unstructured finite element grid, the DO loop in Fig. 1 has a potential vector dependency. To insure no data conflicts occur while also permitting parallel execution, MasPar Fortran provides a COLLISIONS directive. This directive handles gather operations by adding any data that collides at the nodes. As in any finite element application there are a number of statements that require the COLLISIONS directive in the computational kernel loop. To optimize performance, it is essential that these statements run in parallel.

Computational Performance

The numerical performance of the algorithm is illustrated through the following numerical examples. Results of the porting effort are also discussed by comparing execution time requirements for different platforms.

Supersonic Flat Plate Boundary-Layer Flow

The first example considers the development of a flat plate boundary layer in $M_\infty = 3$ flow. The plate geometry, computational domain, and boundary conditions are depicted in Fig. 2. The reference length L is taken to be the length of the plate, the freestream Reynolds number based on this length is 10^3, $\gamma = 1.4$, and the Prandtl number is 0.72. In addition, the temperature of the plate is held constant at the nondimensional stagnation temperature

$$T_s = 1 + [(\gamma - 1)/2]M_\infty^2 \tag{25}$$

The Sutherland viscosity law is used with the dimensionless freestream temperature of 390°R (205 K). Figure 3 shows the 80 × 40 × 10 mesh used for this computation. The mesh is clustered near the plate wall and uniformly distributed in the z direction.

Since Carter[15] first calculated a numerical solution to the two-dimensional problem in 1972, several investigators have used his results as a benchmark for code validation (e.g., Peraire[16] and Devloo et al.[17]). Here, we use it as an initial validation of the present three-dimensional algorithm.

Steady-state density and pressure contours are presented in Figs. 4 and 5, respectively, for the $z = 0.5$ plane. Each figure clearly shows the generation of an oblique shock at the leading edge of the flat plate that curves toward the outflow boundary. No pressure oscillations are observed in the low-speed region near the wall. Profiles of the various flow properties at the outflow boundary are presented in Fig. 6 along with the finite difference results published by Carter. The agreement between these two solutions is excellent.

Transonic Flow Through a Rectangular Nozzle

In this example, the algorithm is applied to flow through an annular to rectangular nozzle configuration. An understanding of the flow physics through this type of nozzle is of current interest for reasons including thrust vectoring and infrared plume signature reduction. The rectangular nozzle provides a relatively simple means to vector the thrust about a single axis. In addition, the rectangular geometry of the resulting freejet enhances mixing between the jet fluid and the surrounding ambient air.

Figure 7 shows the discretized models used for the inviscid and viscous calculations. Boundary conditions are imposed

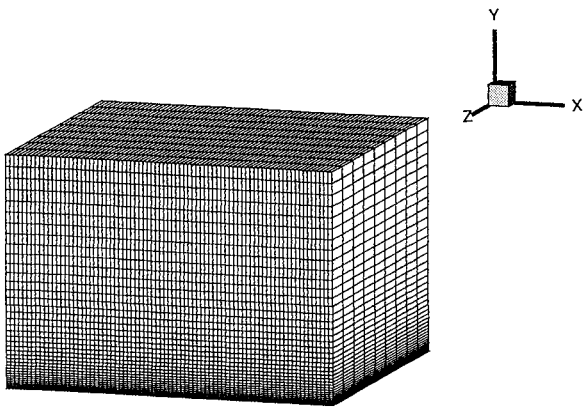

Fig. 3 Computational mesh used for the simulation of supersonic flow past a flat plate.

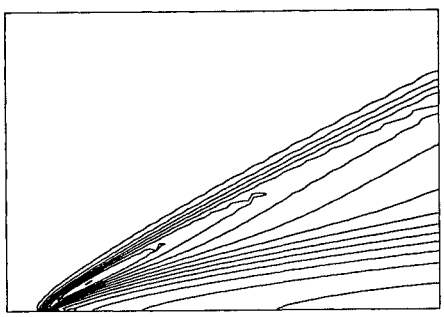

Fig. 4 Density contours for supersonic flow past a flat plate.

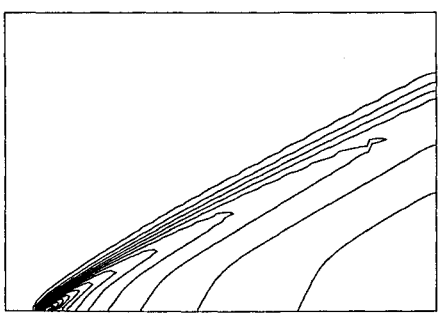

Fig. 5 Pressure contours for supersonic flow past a flat plate.

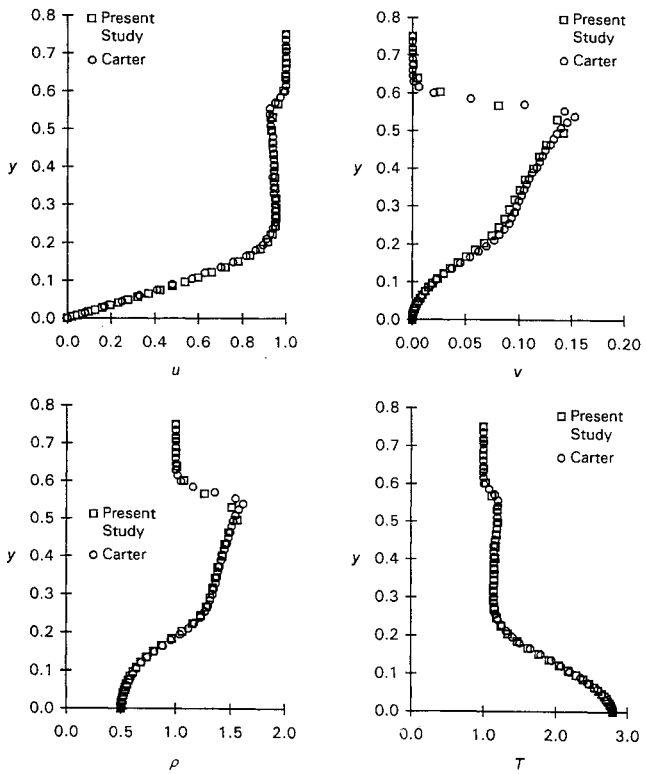

Fig. 6 Comparisons of flow properties at the outflow boundary between the current finite element results and the finite difference solutions of Carter.

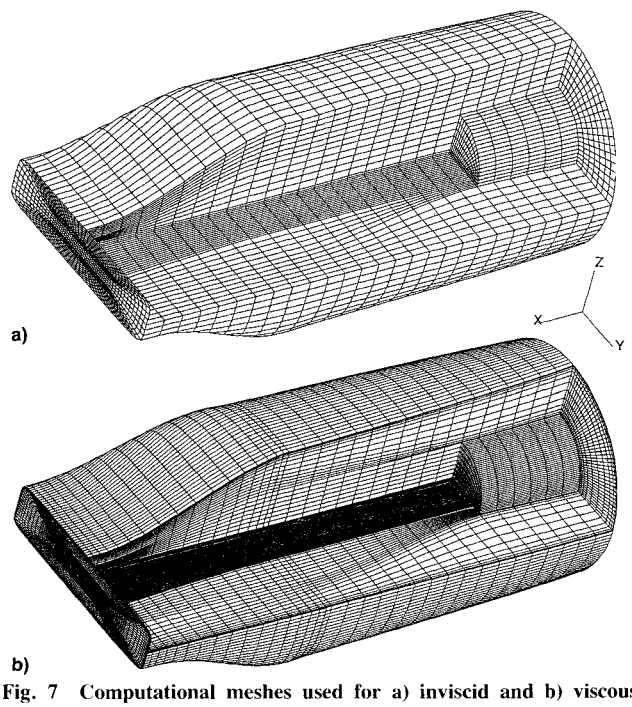

Fig. 7 Computational meshes used for a) inviscid and b) viscous simulations.

Table 1 Execution time comparison (s/timestep)

	15,234 Nodes		77,395 Nodes	
	32-Bit	64-Bit	32-Bit	64-Bit
MasPar MP 2216	0.37	0.55	2.4	3.7
Cray Y-MP (one CPU)	—	0.56	—	5.2
Alliant FX/40	17.2	—	147	—

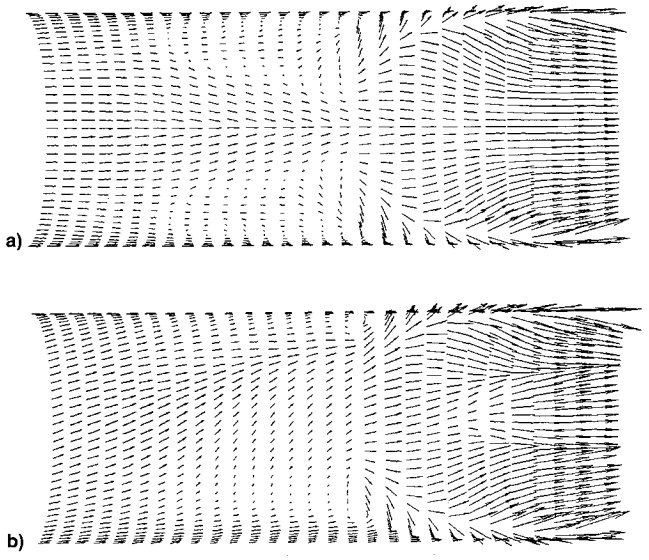

Fig. 8 Velocity field on upper nozzle surface for a) no swirl and b) 20-deg swirl inviscid simulations.

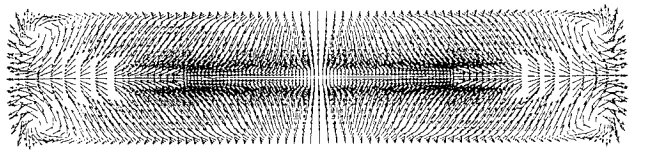

Fig. 9 Transverse velocity vector field at the nozzle exit plane.

along the x-y plane to reduce the model to one-half of the nozzle. The inviscid half-model consists of 15,234 nodes and 13,272 elements; the viscous half-model has 77,395 nodes and 71,512 elements. The reference length L is taken as the half-width of the model in the x-y plane. The freestream Mach number is 0.22, the Prandtl number is 0.72, the reference temperature is 1644°R (865 K), $\gamma = 1.4$, and the pressure ratio $p_{\text{inlet}}/p_{\text{outlet}}$ is 1.83.

Two inviscid solutions are presented; the first with no swirl and the second with a 20-deg inlet swirl angle. A plot of the steady-state velocity vector field is shown in Fig. 8 for both the no swirl and 20-deg swirl simulations. In a viscous simulation at a Reynolds number of 500, axial vortices are formed along the side walls of the nozzle as the fluid progresses through the transition region. A plot of the transverse velocity vectors at the nozzle exit plane is given in Fig. 9, clearly showing a vortex pair at each side wall.

The nozzle simulations described above were run on each of three machines: 1) a MasPar MP 2216, 2) a Cray Y-MP (one CPU only), and 3) an Alliant FX/40. Table 1 compares the execution times per time step for both 32- and 64-bit word lengths. It is seen here that the performance achieved on the MP 2216 was roughly that obtained on a single CPU of the Cray Y-MP. In general, large problems are relatively more efficiently run on the MasPar than smaller problems (in terms of seconds/timestep/node), with maximum efficiency achieved when the number of elements equals the number of PEs. This is nearly the case for the inviscid example, where in excess of 80% of the array of processing elements is utilized. Five memory layers were required for the Navier-Stokes simulations.

Flow in the Nozzle Plume

As a final example, the mixing field downstream of the nozzle exit is computed. Here, the calculated quantities at the outflow of a viscous nozzle simulation with a 20-deg swirl are interpolated onto the inflow boundary of a plume mesh. Figure 10 shows contours of temperature at various cross sections through the computed plume.

Concluding Remarks

A finite element code is presented and used to solve the equations governing compressible fluid flow. The algorithm

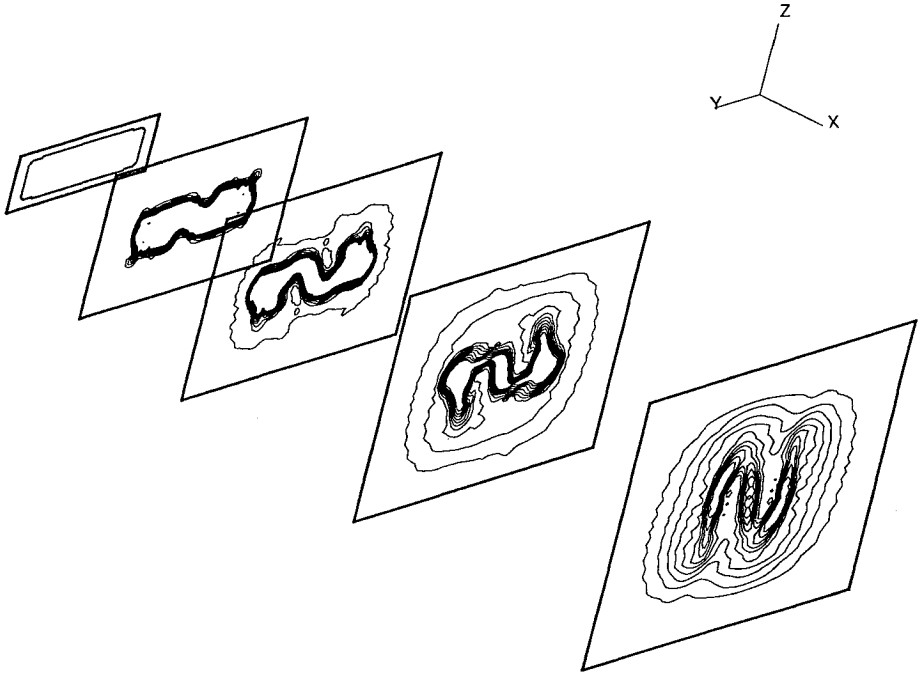

Fig. 10 Contours of temperature at various cross sections through the plume.

is based on hexahedral elements with trilinear density, velocity, and energy, and piecewise constant pressure. The pressure approximation and the selective reduced integration of certain terms in the governing equations improves the behavior of the solution as compared to equal-order, fully integrated (i.e., eight-point), trilinear elements. By converting the Fortran 77 code to Fortran 90, the program is ported onto a MasPar 2216 massively parallel computer (16,384 processors). All of the compiler directives that were added to optimize performance were incorporated into the Fortran 77 code; only the Fortran 77 code needs to be maintained, an important benefit when various computing platforms are utilized. Preliminary efforts incorporating h adaptation also appear promising.

Acknowledgment

The authors would like to thank Roger Chu of MasPar Computer Corporation, Sunnyvale, California, for his assistance in this work.

References

[1]Gunzburger, M. D., *Finite Element Methods for Viscous Incompressible Flows. A Guide to Theory, Practice, and Algorithms*, Academic Press, San Diego, CA, 1989.

[2]Miquel, J., Oñate, E., Quintana, F., Wu, J., and Zienkiewicz, O. C., "Report on the Research Work on Finite Element Analysis of High Speed Compressible Flow," International Center for Numerical Methods in Engineering, Univ. Politècnica de Catalunya, Barcelona, 1989.

[3]Zienkiewicz, O. C., Szmelter, J., and Periare, J., "Compressible and Incompressible Flow—An Algorithm for All Seasons," *Computer Methods in Applied Mechanics and Engineering*, Vol. 78, 1990, pp. 105–121.

[4]Dyne, B. R., "Finite Element Analysis of Incompressible, Compressible, and Chemically Reacting Flows, with an Emphasis on the Pressure Approximation," Ph.D. Dissertation, Univ. of Arizona, Tucson, AZ, 1992.

[5]Brueckner, F. P. "Finite Element Analysis of High-Speed Flows with Application to the Ram Accelerator Concept," Ph.D. Dissertation, Univ. of Arizona, Tucson, AZ, 1991.

[6]Kelly, D. W., Nakazawa, S., Zienkiewicz, O. C., and Heinrich, J. C., "A Note on Upwinding and Anisotropic Balancing Dissipation in Finite Element Approximations to Covective Diffusion Problems," *International Journal for Numerical Methods in Engineering*, Vol. 15, 1991, pp. 1705–1711.

[7]Brueckner, F. P., and Heinrich, J. C., "Petrov-Galerkin Finite Element Model for Compressible Flows," *International Journal for Numerical Methods in Engineering*, Vol. 32, 1991, pp. 255–274.

[8]Pepper, D. W., and Humphrey, J. W., "A Modified Finite Element Method for CFD," AIAA Paper 89-0519, Jan. 1989.

[9]Yu, C.-C., and Heinrich, J. C., "Petrov-Galerkin Methods for the Time-Dependent Convective Transport Equation," *International Journal for Numerical Methods in Engineering*, Vol. 23, 1986, pp. 883–901.

[10]Zienkiewicz, O. C., *The Finite Element Method*, 3rd ed., McGraw-Hill, London, 1977.

[11]Rudy, D. H., and Strikwerda, J. C., "Boundary Conditions for Subsonic Compressible Navier-Stokes Calculations," *Computers and Fluids*, Vol. 9, 1981, pp. 327–338.

[12]Blank, T., "The MasPar MP-1 Architecture," *Proceedings of the COMPCON Spring 90, IEEE Computer Society International Conference*, IEEE Computer Society, Los Alamitos, CA, 1990, pp. 20–24.

[13]MasPar Fortran Reference Manual, Software Version 1.1, MasPar Computer Corp., Sunnyvale, CA, 1991.

[14]MasPar VAST-2 User's Guide, Software Version 1.2, MasPar Computer Corp., Sunnyvale, CA, 1992.

[15]Carter, J. E., "Numerical Solutions of the Navier-Stokes Equations for the Supersonic Laminar Flow over a Two-Dimensional Compression Corner," NASA TR R-385, 1972.

[16]Peraire, J., "A Finite Element Method for Convection Dominated Problems," Ph.D. Dissertation, Univ. College of Swansea, Wales, 1986.

[17]Devloo, P., Oden, J. T., and Pattani, P., "An h-p Adaptive Finite Element Method for the Numerical Simulation of Compressible Flow," *Computer Methods in Applied Mechanics and Engineering*, Vol. 70, 1988, pp. 203–235.