Massively Parallel Three-Dimensional Euler/Navier-Stokes Method

Lyle N. Long,* M. M. S. Khan,† and H. Thomas Sharp‡
Lockheed Aeronautical Systems Company, Burbank, California 91520

A method for solving the three-dimensional Euler and Navier-Stokes equations using the massively parallel Connection Machine computer is described. The program uses a finite-volume, Runge-Kutta time-marching scheme and can accept structured or unstructured grids. The computer program is written entirely in *LISP. Significant computational speedups were obtained over similar codes developed for vector supercomputers.

Introduction

COMPUTATIONAL fluid dynamics (CFD) will play a major role in the design of the next generation of aerospace vehicles. This reliance on CFD has been driven by the rapidly decreasing cost of computers and the rapidly increasing cost of wind tunnel tests. CFD will be used to help reduce the cost, development time, and risk of many aerospace designs. In addition, it may lead to innovative new concepts as our understanding of the flow phenomena is improved.

CFD may well be required for the design of future aircraft since they will be highly complex vehicles involving nonlinear areodynamics, strong aeroelastic interactions, and stringent observability requirements. As these vehicles become more and more complex, design iterations and optimization become extremely difficult. Increased computer power will allow more iterations to be performed and a better design achieved. Some of the detailed simulation studies normally done late in the design cycle may have to be performed earlier to satisfy the design requirements.

Unfortunately, the power of conventional serial or vector computers is inadequate to allow CFD to be used routinely and quickly. The enormous advances in vector processing supercomputers have begun to level off due to physical limitations such as the finite propagation speed of electromagnetic signals. For instance, the Cray-2 and ETA-10 offer rather modest improvements over the Cray-XMP and Cyber-205. This leveling off of supercomputer performance poses a major problem because orders of magnitude increases in computer speed will be required before CFD can achieve its true potential as a design tool.

Recently, there has been a proliferation of new computer architectures that are different from classical serial computers. Most of these incorporate a few processors working in parallel. However, another class of machines uses a massive number of processors and will be the focus of this paper.

Computer Architectures

The wide variety of computer architectures available today are summarized in Fig. 1. The two main variations are in the number of processors and the speed of each processor. From this figure one can detect the two types of computers: coarse grain and fine grain. There are computers that have a few very powerful processors (coarse grain) and computers with very many simple processors (fine grain).

Aside from having different levels of parallelism, parallel computers also have many architectural differences. Most coarse-grain machines are multiple-instruction, multiple-data (MIMD) computers. Each processor can perform different operations on different data. Single-instruction, multiple-data (SIMD) computers have each processor perform exactly the same task but on different data. Nearest-neighbor type aerophysics problems are ideally suited to SIMD computers, as are image processing problems.

If one tries to put a problem that is actually made up of many, many small problems on a coarse-grain computer, one can encounter load-balancing problems. That is, unless each processor has exactly the same computational load, the execution time will correspond to the time required by the slowest processor. On coarse-grain computers, opportunities for significant algorithm improvements are often missed because the focus is on the wrong level of parallelism. However, they do offer flexibility and are more general purpose computers.

There are very few computers on the market that can be classified as massively parallel. Examples of this class of machine are the Geometric Arithmetic Parallel Processor (GAPP), the Distributed Array Processor (DAP), the Massively Parallel Processor (MPP), the NCUBE, and the Connection Machine (CM-2). These are summarized in Fig. 2,

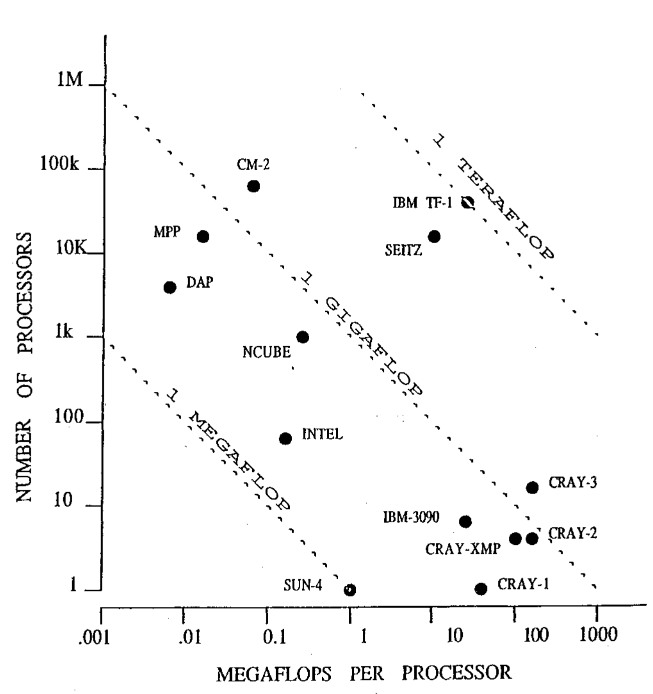


Fig. 1 Performance comparisons of existing and planned computers.

Received Aug. 7, 1989; revision received March 21, 1990. Copyright © 1990 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Senior Research Scientist; currently Assistant Professor, Pennsylvania State University. Member AIAA.

†Research Scientist. Member AIAA.

‡Research Scientist; currently at Arete′ Associates, Sherman Oaks, CA. Member AIAA.

which compares the number of processors, communication architectures, memory, and floating point capabilities of the five machines. Compared with the DAP and the MPP, the CM-2 represents a significant improvement due to its communication system and its Weitek floating point chips. The numbers in parentheses for the CM-2 correspond to the parallelism represented by the Weitek chips. The GAPP, made by Martin Marietta, is used primarily for image processing and has only 128 bits of memory per processor. The NCUBE has fairly modest floating point performance and because of its MIMD design can be more difficult to program than the CM-2.

There are several other massively parallel machines under development. One of these is being developed by Athas and Seitz[1] and their associates at the California Institute of Technology. This machine will have 16K MIMD processors and an expected peak performance of 0.2 trillion operations per second (0.2 Teraflops). Another example is the IBM TF-1,[2] which is planned to have up to 32K processors and to be capable of 1.0 Teraflops. There may also be machines built using the new Intel 860 chip,[3] which could have a peak speed of 50 Mflops (from a single chip). A machine using 20,000 of these chips would have a peak speed of 1.0 Teraflop.

One of the main problems with massively parallel MIMD computers is that with current operating system and compiler technology they are extremely hard to program effectively (in anything other than minor deviations from a SIMD mode). Major breakthroughs in software technology will be required before the full power of MIMD computers can be realized. Due to the advantages and disadvantages of each system, the ultimate machine will most likely have both SIMD and MIMD features.

One of the key parameters for SIMD machines is the level of sophistication to use for the processors. Many SIMD machines currently use very simple 1-bit processors. The higher the floating point precision required, the less advantage there is to using these simple processors. For example, solving very large systems of linear equations requires more precision than an explicit CFD algorithm. On the other hand, some image processing applications require very few bits per cell, and thus 64-bit machines are not cost effective for them. Also, in discussing the optimum processor speed, one must consider the communication speed to have a good balance of I/O and processing.

Quite often when people discuss coarse-grain versus fine-grain computers, they refer to Amdal's law (law of diminishing returns):

$$\text{Speedup} = \frac{1}{R + (1-R)/N}$$

where N is the number of processors and R is the fraction of the computer code that cannot be performed in parallel. For a code with an explicit algorithm, R represents primarily the fraction of time spent applying boundary conditions. Amdahl's law is often used to show why coarse-grain computers are more effective than fine-grain computers. It predicts that a one-million-processor machine with $R = 0.1$ would not perform much better than a 128-processor machine.

While Amdahl's law does apply to some situations, researchers at Sandia Laboratories[4] have proposed that in practice R is usually a function of N. That is, as the problem size is increased, the fraction of serial code usually decreases. Amdahl's law assumes that R is a constant, independent of N. To illustrate the effect this has, if you assume R is proportional to $1/N$, then the speedup is linear with number of processors N.

This empirical rule also does not take into account the fact that problems such as CFD are very well suited to massively parallel computers, and significant algorithm improvements are made possible by the machine architecture. In addition, this rule does not include memory access effects. Many numerical problems will require the use of asynchronous "out-of-core" memory such as parallel disk drives or solid-state memory devices. A machine that balances the computation and memory requirements will be essential.

In summary, it should be emphasized that there is currently no clear consensus as to what are the best techniques for solving the fluid dynamic equations on the various types of computers. The fastest approach will be the one that most effectively matches the algorithm and the computer. The best approach must also consider the many other factors that make up a computing system: software, user friendliness, reliability, and cost.

Connection Machine

The CM-2[5,6] can have up to 65,536 physical processors and has a peak speed of approximately 5 Gflops (64-bit precision). These are extremely simple 1-bit processors, each with 32K bytes of local memory. In addition, it is equipped with a floating point chip (Weitek) for each group of 32 1-bit processors. The CM-2 therefore has two levels of parallelism, and when viewed from this perspective, a 2048-Weitek CM-2 and a 1024-processor NCUBE are of similar size. However, the CM-2 is an SIMD machine and is very easy to program compared to an MIMD machine such as the NCUBE. The CM-2 processors all operate in lockstep on data held in their local memories, executing a single stream of instructions. The instruction stream is generated by a host computer that can be a Symbolics, a VAX, or a SUN/4. A single user can be allocated a portion of the available processors, the smallest portion normally being 8192 processors.

Another feature of the CM-2 is the virtual processor mechanism. This mechanism allows each physical processor to automatically act as a number of virtual processors. This is accomplished by slicing the memory of a physical processor into equal parts that then simulate additional processors. For instance, a CM-2 with 65,536 processors can be made to act as a 1,048,576 processor machine [virtual processor (VP) ratio of 16] where each virtual processor has 2048 bytes of memory. This feature is very useful for finite-difference codes where each cell in the computational space requires roughly 500 bytes of memory for storing the relevant grid and flow properties.

Currently, the user can program the CM-2 in four languages: *LISP, C*, Fortran-8X, and PARIS (Parallel Instruction Set). The first three are parallel supersets of LISP, C and Fortran-77, respectively, designed to give programmers access to the parallel capabilities of the machine from a high-level

	GAPP	DAP	MPP	NCUBE	CM-2
Number of Processors	72 / chip	4 K	16 K	1 K	64 K (2 K)
Architecture	2-D Grid	2-D Grid	2-D Grid	Hypercube	Hypercube
Memory/Processor (bytes)	16 / chip	500	125	64K	32K (1024K)
Floating Point Hardware	NO	NO	NO	WEAK	YES
Processor Type	SIMD	SIMD	SIMD	MIMD	SIMD

Fig. 2 Comparison of GAPP, DAP, MPP, NCUBE, and CM-2.

language. PARIS is a low-level instruction set that provides a facility for optimizing the execution speed of critical parts of a program.

All software development for the CM-2 is done on the host computer. Consequently, the user has access to all the development tools provided on these computers, including editors and debuggers. In addition, Thinking Machines, Inc., has developed a *LISP simulator.[7] These features make it quite easy to use the CM-2 since most engineers are familiar with either a VAX, a Sun, or a Symbolics.

Software

The new generation of parallel computers will require a new generation of software. For some time to come, this will be the only way to achieve order-of-magnitude improvements in performance. While this means one must spend the necessary time and money doing this, it is often worthwhile to rewrite code to incorporate software and hardware advances. In addition, a wide variety of modern tools—new computer languages, software development workstations, and parallel constructs—make this task much less costly.

In terms of programming languages, Fortran-77 is not necessarily the best scientific language. It is the most common language today primarily because of the momentum generated by its widespread use in the 1960s and 1970s. However, familiarity with a language should not be the only criteria by which a language is judged. The future of Fortran really rests on how well Fortran-8X is implemented.

One of the reasons for people turning to new computer languages (such as C or Lisp) is the availability of very advanced software development workstations such as the Symbolics. These computers allow one to write good quality code in about a tenth the time it would require using minicomputers and Fortran. This may be due to the AI influence on these machines.[8] They have a well-developed and easy to use windowing system, combined with high-quality interactive debugging, incremental compiling, and interpreted mode of operation. Most of these features would be more difficult to implement in a non-AI-influenced machine. In addition, Symbolics supplies large amounts of prewritten software (in Lisp) that can be incorporated into the user's code. Another reason for turning to new languages is to use object-oriented programming.

One disadvantage to Lisp is its current inability to use infix notation, which makes the code more difficult to read. For example, the equation

$$x = (y*6 + 2)/4$$

	CONTINUUM	COMPRESSIBLE	NONLINEAR	ROTATIONAL	VISCOUS	RAREFIED
Boltzmann	o	o	o	o	o	o
Navier-Stokes	o	o	o	o	o	
Euler	o	o	o	o		
Full Potential	o	o	o			
Prandtl-Glauert	o	o				
Laplace	o					

Fig. 3 Capabilities of gas dynamic equations.

would be programmed as

(setf x (/ (+ (y 6) 2) 4))*

in Lisp.

Software development time can also be significantly shorter on the CM-2 than on a classical serial computer. This can be attributed to the fact that the SIMD formulation of many problems often leads to compact and simple code. The majority of the nested do-loops that appear in serial or vector code do not appear. In fact, the sequential formulation of many problems becomes complex because the problem being tackled is inherently of an SIMD nature. However, the vector computers will also be using array constructs to remove the manual treatment of arrays.

Fluid Dynamics

The basic equations of fluid dynamics, and their capabilities, are summarized in Fig. 3. These equations range from the simple Laplace equation ($\nabla^2\phi = 0$) to the Boltzmann equation of kinetic theory. The linear equations (Laplace and Prandtl-Glauert) can be solved quite effectively numerically. These are routinely used for aerodynamic design methods. However, as one moves up the chain of equations, less and less is known about the behavior of the equations, analytical solutions become rare, and numerical solutions are more and more difficult to obtain.

The Navier-Stokes equations include molecular and turbulent dissipation and diffusion, but, due to computer limitations, this is usually only possible when one "models" the turbulence. The Euler equations are obtained by going to the limit of zero viscosity and conductivity. While vorticity transport can be modeled by the Euler equations, the effects of molecular dissipation of momentum and the conduction of heat are not included in the Euler equations.

At the extreme upper end of the spectrum of gas dynamic models is kinetic theory that is governed by Boltzmann's equation. This approach has a wider range of validity than the continuum approach, but numerical simulation required an enormous number of computations and storage. When the mean free path is not much smaller than a characteristic length, one must resort to the Boltzmann equation.

Grid Generation

There are really two parts to CFD codes: the flow solver algorithm and the grid generation. Grid generation is one of the main bottlenecks in using CFD for complex three-dimensional shapes in an aircraft design environment. One must be able to rapidly generate grids about a vehicle to have an effective design tool. For complex three-dimensional configurations such as complete aircraft, there is no single grid-generation scheme that is adequate. One must decide what type of grid-generation scheme to use based upon the flowfield, the geometry, the computer resources, and the engineering resources.

Grid-generation codes all have their advantages and disadvantages. The most common CFD codes use structured grids, and the grid generation schemes are based upon solving hyperbolic, parabolic, elliptic, or algebraic equations for the nodal points of the grid given the surface of the vehicle. The simplest codes to use are those based on conformal mapping techniques, but these techniques are only effective for relatively simple wing-body combinations. Some of the more difficult codes to use are those based on the algebraic methods such as transfinite interpolation.[9] These methods require a great deal of user interaction and effort, but they are capable of generating grids around very complex three-dimensional configurations.

Another alternative is to use programs such as PDA's PATRAN or SDRC's IDEAS software , which were originally developed for structural analysis methods and can produce unstructured grids. Unstructured grids are also produced by a few CFD grid-generation methods.[10,11] For complex geometries, it appears that unstructured grids are much easier to generate than structured grids.

However, there are a number of uncertainties regarding the use of unstructured flow solvers. Of particular concern is the accuracy possible on unstructured grids, especially tetrahedral grids. Solving the fluid dynamic equations on unstructured grids can also take more computer time since indirect addressing makes it more difficult to vectorize the code (on vector supercomputers) or to exploit the nearest neighbor communication (on parallel computers). In addition, it is difficult to implement a turbulence model on unstructured grids.

Euler/Navier-Stokes Algorithm

This section will describe a three-dimensional Euler/Navier-Stokes method[12] written for the Connection Machine using the *LISP computer language. The form of the Navier-Stokes equations being solved is

$$\frac{\partial}{\partial t}\iiint_V \boldsymbol{P}\,\mathrm{d}V = -\iint_S \boldsymbol{F}\,\mathrm{d}S$$

where

$$\boldsymbol{P} = \begin{bmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho E \end{bmatrix}$$

$$\boldsymbol{F} = \begin{bmatrix} \rho u_n \\ \rho \boldsymbol{u} u_n + p\boldsymbol{n} - \boldsymbol{T} \\ \rho H u_n - \boldsymbol{T}\cdot\boldsymbol{u} + \boldsymbol{q}\cdot\boldsymbol{n} \end{bmatrix}$$

and ρ, $\boldsymbol{u}$, E, $\boldsymbol{T}$, and $\boldsymbol{q}$ are the density, velocity, total energy, traction, and heat transfer, respectively. The traction[13] is defined as $\boldsymbol{T} = n_i\tau_{ij}$, where

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + (\mu_b - ⅔\mu)\frac{\partial u_i}{\partial x_i}\delta_{ij}$$

where n_i is the unit normal to the surface, p the pressure, μ is the dynamic viscosity, and μ_b is the bulk viscosity. The above equations simply state that the rate of change of the mass, momentum, and energy of the fluid within the volume V is equal to the flux of these quantities through the surface S bounding the region V. They are applied to each cell in the computational domain, and V and S refer to the volume and surface area of each cell. The discretized form of the equations for tetrahedral or hexahedral cells are

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{P}\Delta V = -\sum_{\text{Faces}=1}^{4\text{ or }6}\boldsymbol{F}\Delta S$$

The velocity and temperature gradients are obtained using Gauss' divergence theorem:

$$\iiint_V \frac{\partial u_i}{\partial x_j}\,\mathrm{d}V = \iint_S u_i n_j\,\mathrm{d}S$$

which is approximated by

$$\frac{\partial u_i}{\partial x_j} = \frac{1}{\Delta V}\sum_{\text{Faces}=1}^{4\text{ or }6} u_i n_j \Delta S$$

where ΔV and ΔS are the cell volume and cell-face area, respectively. The above equations are integrated in time using a three-, four-, or five-stage Runge-Kutta scheme. The flow quantities on the left-hand side are cell-centered quantities, and the flow quantities on the right-hand side are cell-face quantities.

The multistage Runge-Kutta scheme[14,15] is schematically represented as

$$Q^{(1)} = Q(t) - \frac{\alpha_1\Delta t}{\Delta V}\sum_{\text{Faces}} F[Q(t)]$$

$$Q^{(2)} = Q(t) - \frac{\alpha_2\Delta t}{2\Delta V}\sum_{\text{Faces}} F[Q^{(1)}]$$

$$Q(t+\Delta t) = Q(t) - \frac{\Delta t}{\Delta V}\sum_{\text{Faces}} F[Q^{(n)}]$$

where $Q = [\rho, \rho\boldsymbol{u}, \rho E]^T$ and the superscripts refer to the stages in the time-marching scheme. Typically, a three-, four-, five-stage scheme is used. This is the same algorithm that is used in the Lockheed/USAF Three-Dimensional Euler/Navier-Stokes Aerodynamic Method (TEAM).[9]

As mentioned before, the program can use structured or unstructured grids. The structured grid code uses C-H topology grids and is limited to wing-body, wing-alone, or airfoil geometries. The unstructured code is highly suited to solving for the flow around complex three-dimensional bodies, such as complete aircraft. The unstructured grid approach will also allow inclusion of adaptive grid refinement in a straightforward manner in the near future, thus simplifying the grid-generation task.

The cell-face quantities are approximated by averaging the two cell-centered values on either side of each face. On a uniform mesh, this is equivalent to a second-order-accurate central-difference scheme that has to be augmented by numerical dissipation to maintain stability. Jameson et al.[14] developed an adaptive dissipation scheme that adjusts the dissipation depending on the strength of the pressure gradient. This is implemented slightly differently on an unstructured grid, since each cell only knows its nearest neighbors. That is, no connectivity information is provided for cells that are two cells away. This means fourth-order differences must be performed using a nested Laplacian approach.[16] A Laplacian operation is performed twice to obtain a fourth-order derivative.

In addition to the above dissipation scheme, this code also includes Roe's approximate Riemann solver,[17] which is a characteristic-based upwind scheme. This is usually required at high Mach numbers. So far only a first-order accurate scheme has been programmed. A *Lisp function was written to calculate dissipative fluxes using a form of Roe's scheme described by Gnoffo.[18] The user can choose between the adaptive dissipation or the upwind scheme.

Boundary Conditions

The boundary conditions used are the same as those used in the TEAM code.[9] For subsonic flow, a locally one-dimensional Riemann invariant problem is solved to achieve the correct propagation of the disturbances. For any cell face that lies on the far-field boundary, one assumes there is an imaginary ghost cell. The flow variables in this ghost cell are set using the local Riemann invariants for incoming and outgoing waves:

$$R_\infty = q_{n_\infty} - \frac{2}{\gamma - 1}c_\infty$$

$$R_e = q_{n_e} + \frac{2}{\gamma - 1}c_e$$

The normal velocity and sound speed on the cell face are then

$$q_n = (R_e + R_\infty)/2$$

$$c = (R_e - R_\infty)(\gamma - 1)/4$$

At an outflow boundary, the flow variables in the ghost cell become

$$\rho = [\rho_e^\gamma c^2/p_e\gamma]^{1/(\gamma-1)}$$

$$q = q_e + (q_n - q_{n_e})n$$

At an inflow boundary, the flow variables in the ghost cell become

$$\rho = [\rho_\infty^\gamma c^2/p_\infty\gamma]^{1/(\gamma-1)}$$

$$q = q_\infty + (q_n - q_{n_\infty})n$$

In both cases, the aforementioned density is used to find pressure according to

$$p = \rho c^2/\gamma$$

The preceding boundary conditions were evaluated by Raj and Sikora[19] and were shown to be very important and effective.

Residual Smoothing

The maximum allowable time step at which a scheme is stable is defined in terms of a Courant-Friedrichs-Lewy (CFL) number. If the CFL number is too large, the difference scheme will be unstable. It is possible to replace the residual at each point by a weighted averaged of neighboring values, thus smoothing the residuals.[20] This results in a larger permissible value for the CFL number. Two types of residual smoothing were implemented here: implicit and explicit.

In a one-dimensional case, the averaging is given by

$$(1 - \epsilon\delta_i^2)\overline{R} = R$$

where δ_i^2 is a central-difference operator, ϵ is the smoothing coefficient, R is the computed residual, and $\overline{R}$ is the smoothed residual.

The implicit method comes from rewriting the equation for the residuals as

$$R_i = -\epsilon\overline{R}_{i-1} + (1+2\epsilon)\overline{R}_i - \epsilon\overline{R}_{i+1}$$

Stability can be maintained for any CFL number λ, if

$$\epsilon \geq 0.25[(\lambda/\lambda_*)^2 - 1]$$

and λ_* is the stability limit of the unsmoothed scheme. Hence, for the implicit scheme a tridiagonal system must be solved at each point in the flowfield. A three-dimensional implementation basically uses three one-dimensional schemes.

The explicit smoothing scheme is written as

$$R_i = -\epsilon R_{i-1} + (1+2\epsilon)\overline{R}_i - \epsilon R_{i+1}$$

where $0<\epsilon<0.25$. The explicit method uses the above equation in an iterative fashion using a point Jacobi method.

The implicit scheme requires solving a tridiagonal system of equations. Simply converting existing code designed for vector or scalar computers to run on the CM-2 results in very poor utilization of the CM-2. This occurs because most of the CM-2 processors are shut off (since the schemes usually work on a single column at a time), and the scheme results in prohibitively long computational times. Several groups are working on effectively solving larger systems of linear equations on the CM-2. This will require some effort but should eventually result in performance competitive with vector computers.

On an 8K CM-2 (at a VP ratio of 8), the explicit and implicit smoothing schemes ran at roughly 100 and 2 Mflops, respectively. This simply shows that not all schemes can be ported directly onto a parallel processor without modification. It does not mean that implicit schemes cannot run efficiently on massively parallel computers. This is an example of one of the greatest challenges in programming massively parallel computers—the need to match the algorithm and the architecture. Existing schemes that have been very successful with serial and vector machines may not map directly onto massively parallel machines.

On massively parallel computers, one should not be limited by algorithms developed for scalar machines. Instead of trying to map a particular algorithm onto a massively parallel computer, one should try to find the algorithm that most effi-

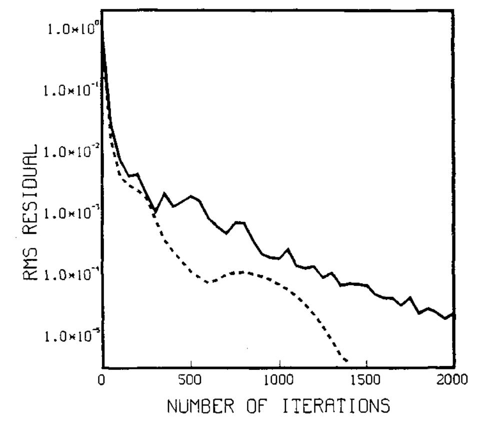


Fig. 4 Effect of enthalpy damping on convergence history NACA airfoil $M = 0.8$ and $\alpha = 0°$, 8192 cells (—— undamped, ---- damped).

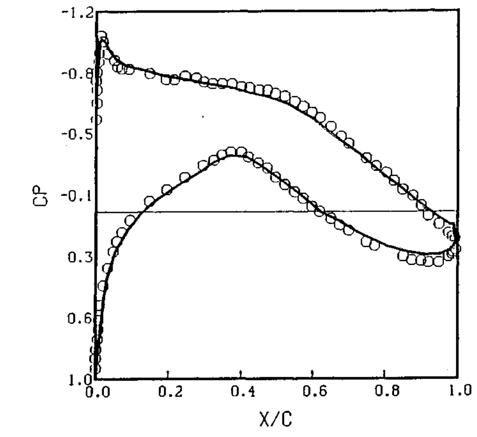


Fig. 5 Surface pressure distribution for RAE 2822 airfoil, $M = .676$, $\alpha = 2.4$ deg, and $Re = 5.7 \times 10^6$ (—— turbulent, 0 experiment).

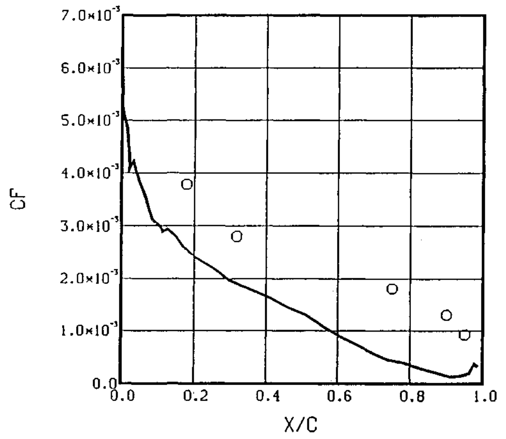


Fig. 6 Surface skin friction distribution on RAE 2822 airfoil, $M = .676$, $\alpha = 2.4$ deg, and $Re = 5.7 \times 10^6$ (—— turbulent, 0 experiment).

ciently uses the machine architecture for each particular application.[21]

Enthalpy Damping

For steady flow, the Euler equations can be modified by introducing forcing terms proportional to the difference between the local total enthalpy and the freestream total enthalpy as follows:[22]

$$\frac{\partial}{\partial t}\iiint_V P\,\mathrm{d}V = -\iint_S F\,\mathrm{d}S + \iiint_V Q\,\mathrm{d}V$$

where

$$Q = -(H - H_\infty)\alpha P + \begin{bmatrix} 0 \\ 0 \\ \beta\rho \\ \gamma \end{bmatrix}$$

where the coefficients α and β are user specified and were taken to be equal to 0.1. The forcing terms do not alter the steady-state solution if the problem satisfies a constant total enthalpy condition. The net effect of these additional terms is that there is an added damping of waves that results in significantly greater convergence rates (see Fig. 4 and Ref. 20).

Algebraic Renormalized Group (RNG) Method for Turbulence

It can be quite challenging to incorporate turbulence models into unstructured grid codes, especially algebraic models. Some of the k-ϵ models may work quite well, however. So far no turbulence model has been incorporated into the unstructured grid code.

However, in the structured grid code we have incorporated a turbulence model based on the work of Yakhot and Orzsag.[23] This model has shown a great deal of promise by accurately predicting a number of very difficult flows and theoretical constants. The mathematics behind this technique are very complicated, but the final formulae can be presented quite easily. The form of the model being used here is

$$\tilde{\nu}^4 + (C_c - 1)\tilde{\nu} - A = 0$$

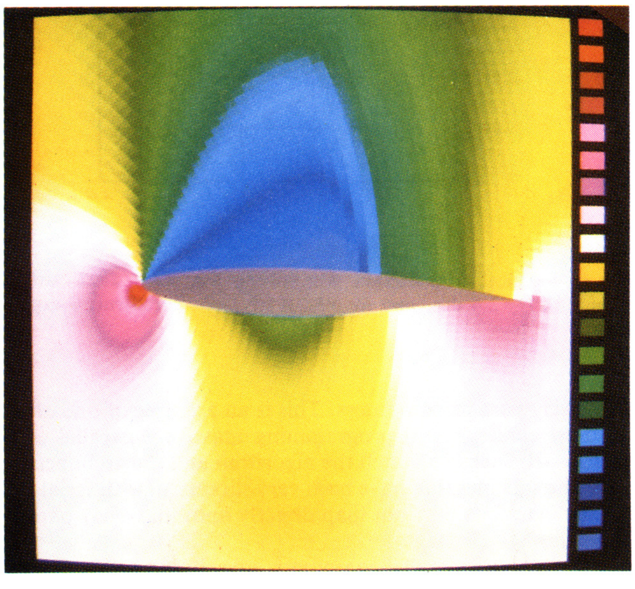


Fig. 7 Flowfield color-shaded according to pressure for RAE 2822 airfoil $M = .676$ and $\alpha = 1.89$ deg (blue: low, red:high).

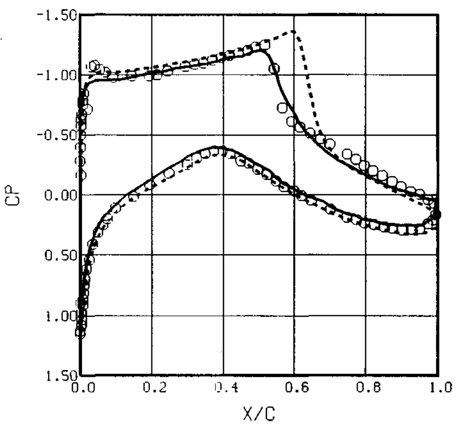


Fig. 8 Surface pressure distribution for RAE 2822 airfoil, turbulent, $M = .75$, $\alpha = 2.8$ deg, and $Re = 6.2 \times 10^6$ (—— RNG, ---- Baldwin-Lomax, 0 experiment).

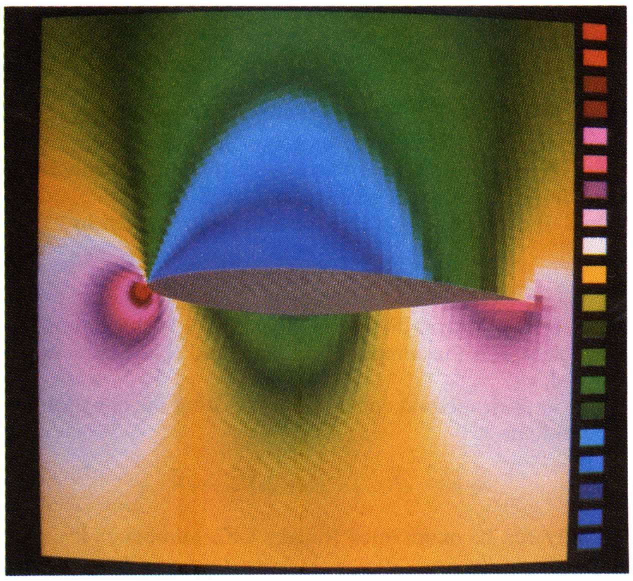


Fig. 9 Flowfield color-shaded according to pressure for RAE 2822 airfoil $M = .725$ and $\alpha = 2.4$ deg (Blue: low, Red: high).

where

$$A = C_k \frac{\tau^2}{\rho^2 \nu_o^4}\left(\frac{1}{y} + \frac{1}{\delta^*}\right)^{-4}$$

and

$$C_k = 0.019, \qquad 50 < C_c < 100$$

and $\tilde{\nu} = (\nu_t + \nu_0)/\nu_0$, y is the distance to the wall, and δ is the boundary-layer thickness. If $A \le C_c\tilde{\nu}$, then $\tilde{\nu} = 0$. Sutherland's formula is used to predict ν_0. The above equation is a quartic equation with two imaginary, one negative, and one positive roots for most values of A. We solve it by using a Newton iteration scheme, which usually converges in 2 or 3 steps.

The biggest drawback to the above method is that it required calculating the displacement thickness δ^* of the boundary layer. This is not easy to do accurately for complicated flowfield and geometries.

While these formulae show a great deal of promise (as do the k-ϵ formulations of RNG), a great deal of testing and validating remain. We have begun programming these for the Connection Machine and plan to evaluate their accurary for a number of aerodynamic flows and Mach numbers. Some results for the RAE 2822 and NACA 0012 airfoils are presented in the next section.

Results

In this section a number of numerical results will be presented. The code is based upon the three-dimensional equations. One-dimensional and two-dimensional flows are solved by enforcing symmetry conditions in certain directions. Presented below are some of the two-dimensional and three-dimensional results.

RAE 2822 Airfoil

Two turbulent flow conditions will be presented here for the RAE airfoil: 1) $M = 0.676$, $\alpha = 1.89$, and $Re = 5.7 \times 10^6$; and 2) $M = 0.75$, $\alpha = 2.8$, and $Re = 6.2 \times 10^6$. Each grid had 32,768 cells ($256 \times 128 \times 1$), and the clustering near the surface was such that y^+ was less than 1.0 in the first cell. The surface pressure distributions are shown in Figs. 5 and 8 along with experimental data. Figure 6 shows the surface skin friction prediction along with the experimental data for the first case. Figures 7 ($M = 0.676$ and $\alpha = 1.89$ deg) and 9($M = 0.725$ and $\alpha = 2.4$ deg) show the inviscid flowfield predictions color-shaded according to pressure.

The color images were obtained using the CM-2 frame buffer, which allows the generation of very sophisticated color graphic images. However, at the present time, there is very little graphics software supplied with the CM-2. The software to generate the images shown here was developed by Gary Demos (DemoGraFX) under Lockheed sponsorship. Each image takes from a few seconds to a few minutes to generate. Velocity vectors are drawn very fast, but the color-shaded images require more time. Engineering and scientific productivity is greatly enhanced by integrating graphics hardware with supercomputing as Thinking Machines, Inc., has done.

NACA 0012 Airfoil

Turbulent flow over an NACA 0012 airfoil was also simulated by $M = 0.55$, $\alpha = 8.34$, and $Re = 9.0 \times 10^6$. This airfoil is symmetric with a 12% thickness ratio. This case also had 32,768 cells ($256 \times 128 \times 1$). The surface pressure distribution is compared to experimental data[24] in Fig. 10. The agreement is quite good.

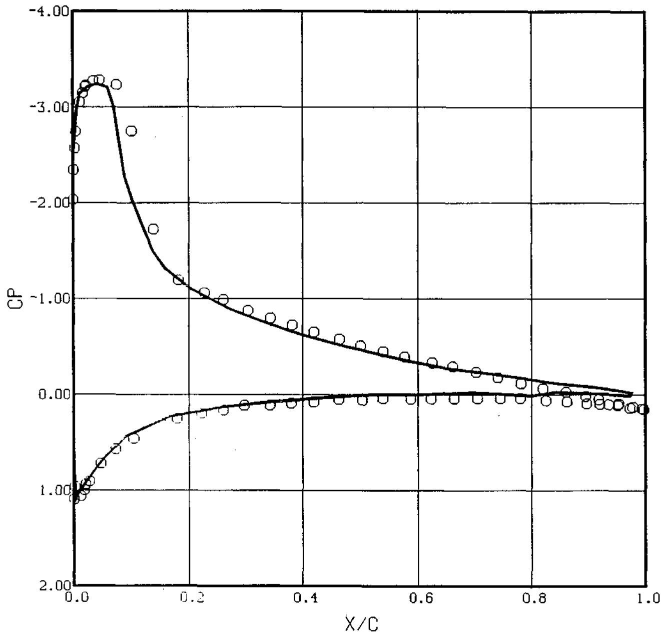
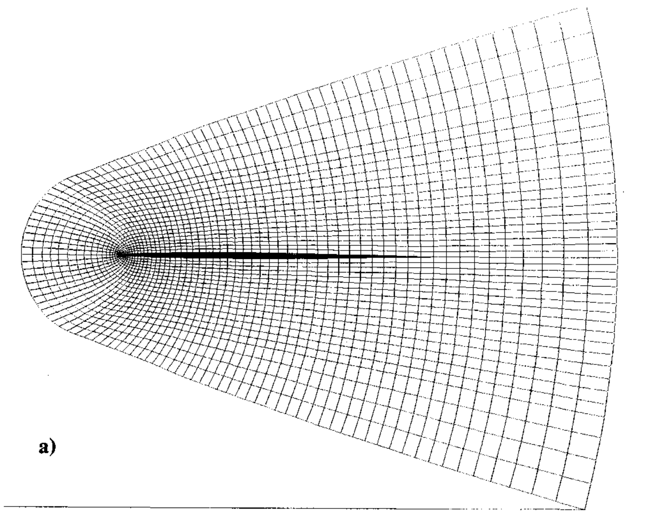


Fig. 10 Surface pressure distribution for NACA 0012 airfoil, turbulent, $M = .55$, $\alpha = 8.34$ deg, and $Re = 9.0 \times 10^6$ (—— RNG, 0 experiment).


a)

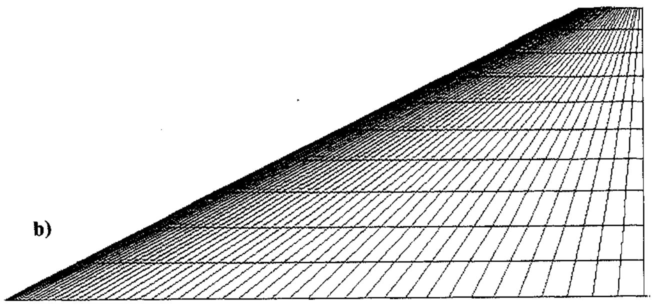
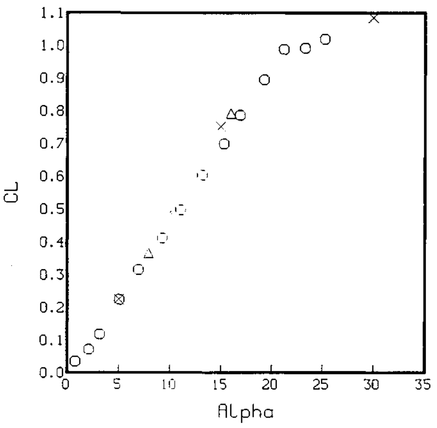

b)

Fig. 11 Grid used for cropped delta wing: a) plane of symmetry and b) planform.



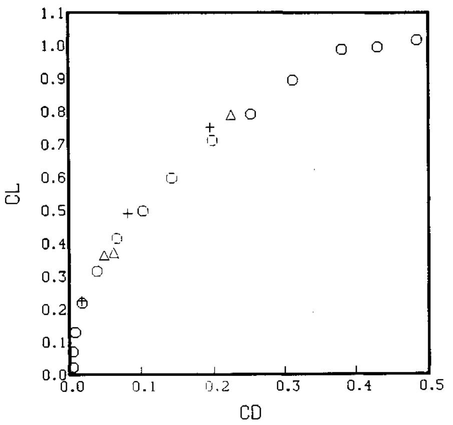


Fig. 12 Predicted and experimental C_L and C_D for a cropped delta wing using CM-2 (0 experiment, Δ CM-unstructured code, X CM-structured code).

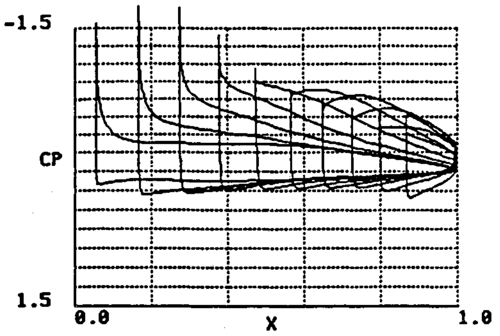


Fig. 13 Surface pressure distribution for cropped delta wing ($M = .6$ and $\alpha = 10$ deg).

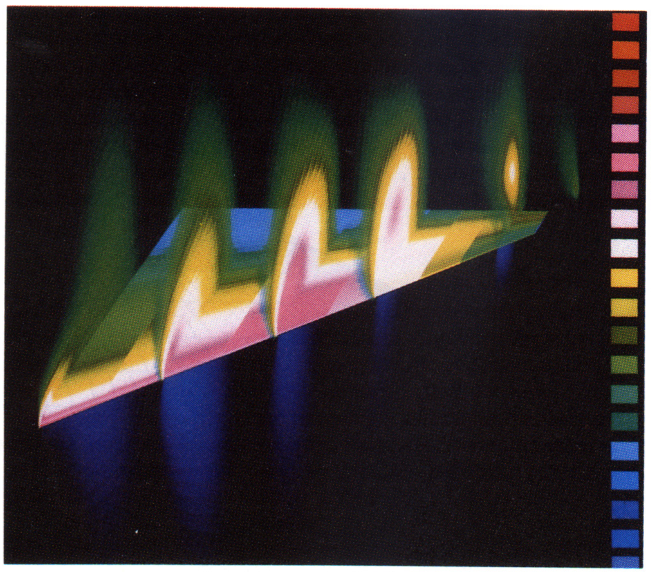


Fig. 14 Flowfield and wing surface color shaded according to pressure for a cropped delta wing, $M = 0.6$ and $\alpha = 15$ deg, (blue: low, red: high).

Cropped Delta Wing

This section will present results for a cropped delta wing with an NACA 63A002 airfoil section, a 63-deg swept leading edge, and an aspect ratio of 1.64. This wing has been evaluated extensively.[25,26] The grid is shown in Fig. 11, which show the plane of symmetry and the wing surface. It has $128 \times 32 \times 16$ cells in the streamwise, normal, and spanwise directions, respectively, which required a VP ratio of 4 on a 16K ©M-2.

Several different flow conditions were evaluated for this wing (all at $M = 0.6$): Structured grid code: $\alpha = 5$, 10, and 15 deg, Unstructured grid code: $\alpha = 8$ and 16 deg, where α = angle of attack. The predicted and experimental values of C_L and C_D are shown in Fig. 12. Also shown is one data point from the Navier-Stokes code, which shows the same lift as the Euler code but slightly higher drag. The agreement between the predictions and experiment is quite good. Figure 13 shows the pressure distribution for several spanwise stations. Figure 14 shows an oblique view of the wing color-shaded according to pressure for $\alpha = 15$ deg. Also shown are selected planes in the flowfield. The vortex core is readily apparent. The planes are set to transparent when the pressure is near to the free-stream value. These solutions can be displayed quite easily while the code is running, including changing the view angle or quantity displayed.

Lockheed/AFOSR Wing C

Wing C is a Lockheed/AFOSR designed low-aspect ratio wing.[27] Its planform is characteristic of a transonic maneuver fighter with varying camber and twist. This wing was analyzed at M = 0.85 and 5 deg angle of attack using a $128 \times 32 \times 32$ grid. Figure 15 shows the planform and pressure distribution along various spanwise stations; the predictions are compared to experiments by Keener.[28] The predictions gave $C_L = 0.561$ and $C_D = 0.0435$.

CPU Time Comparisons

As mentioned earlier, the unstructured grid code dramatically reduces the grid-generation complexities for realistic aircraft configurations. But this also means one must use the

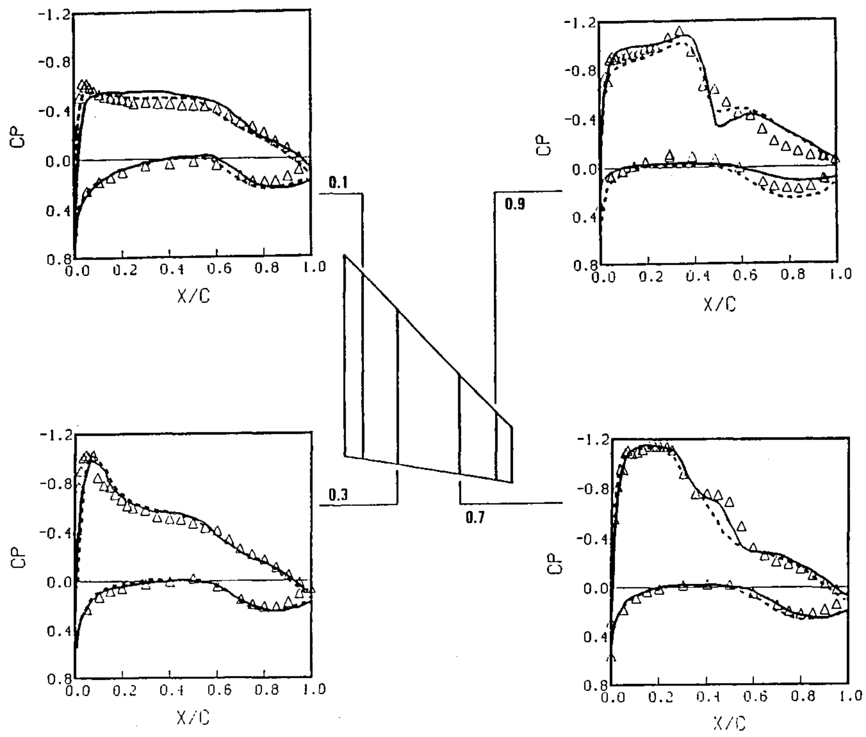


Fig. 15 Pressure distributions for various spanwise locations of the Lockheed/AFOSR wing C, $M = 0.85$ and $\alpha = 5$ deg (—— CM-structured code, ---- TEAM, △ experiment).

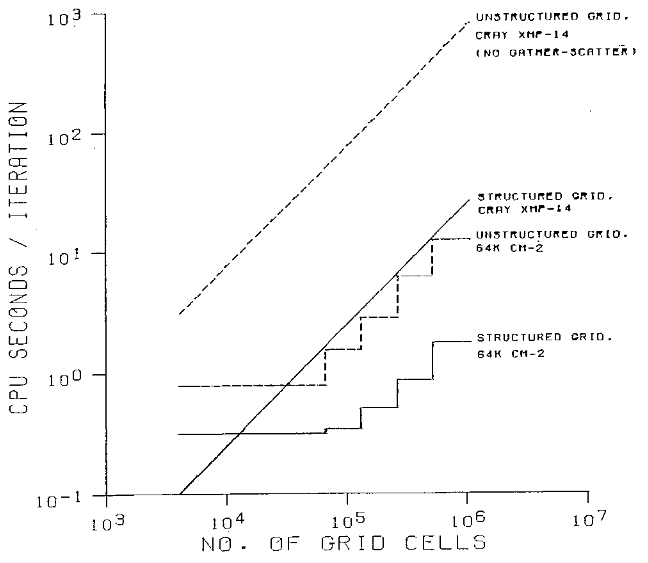


Fig. 16 64K CM-2 and Cray XMP-14 performance comparisons for structured grid and unstructured grid codes.

general purpose communications network (router) on the Connection Machine, which is relatively slow compared to the North-South-East-West (NEWS) network. For the unstructured code, the communications will dominate the CPU time. This type of communication is roughly 10 times slower than two-dimensional nearest-neighbor NEWS communication. It should be noted that these algorithms do not work well on vector computers either. The other code uses structured grids and achieves very high MFLOP rates since communications are very fast for nearest-neighbor type problems on the CM-2.

Figure 16 shows code performance (CPU seconds per iteration) as a function of the number of cells in the flowfield for both of the codes (extrapolated to a 64K processor CM-2 with 8K bytes/processor and 32-bit floating point hardware). The codes have been run on 8K, 16K, and 32K machines, so the extrapolations are expected to be very accurate. The performance of the highly optimized TEAM code on a single processor Cray-XMP is also shown. TEAM requires roughly 25 μs per cell per iteration. The final curve shown is for an unvectorized, unstructured cell per iteration. The final curve shown is for an unvectorized, unstructured grid code on a Cray-XMP (which would require an order of magnitude less of time if one used gather scatter). The three main points to make here are the following:

1) An unstructured grid code on the CM-2 is roughly as efficient as the TEAM code is on a Cray-XMP.

2) A structured grid code on the CM-2 is roughly 15 times faster than TEAM on a Cray-XMP.

3) An unstructured grid code on the CM-2 is roughly 100 times faster than an unstructured Euler code on a Cray-XMP when no gather scatter is used.

The TEAM code uses structured, zonal grids. The structured grid code requires only 1.7 μs per cell per iteration on the CM-2. This means a wing with 524,000 cells can be computed in less than 15 min. With additional optimizing, this code should run at least two times faster.

Conclusions

In conclusion, a very general three-dimensional Euler/Navier-Stokes code has been developed for the Connection Machine in *LISP. This is one of the first examples of a production quality code being written for CM-2. Preliminary predictions show good correlations to experiment and other predictions. The authors are very pleased with the *LISP programming language and prefer it over Fortran, primarily because of the superb Symbolic program development environment and the ability to run in interpreted mode.

The code runs roughly 10 times faster than similar codes on vector supercomputers. Massively parallel computers may offer revolutionary advances, at a time when evolutionary advances are not enough.

Acknowledgments

The authors wish to thank Gary Demos of DemoGraFX for generating the color-graphics renderings of the flowfields. We would also like to thank Pradeep Raj for many constructive comments and suggestions and for his assistance in generating the CFD grids.

References

[1] Athas, W. C., and Seitz, C. L., "Multicomputers: Message-Passing Concurrent Computers," *Computer,* Vol. 21, Aug. 1988, pp. 9–24.

[2]Denneau, M. M., Hochschild, P. H., and Shichman, G., "The Switching Network of the TF-1 Parallel Supercomputer." *Supercomputing Review,* Vol. 1, No. 12, 1988.

[3]"News Pixels & Vectors," *Computer Graphics Review,* Vol. 4, No. 3, 1989, p. 10.

[4]Gustafson, J. L., Montry, G. R., and Benner, R. E., "Development of Parallel Methods for a 1024-Processor Hypercube," *SIAM Journal on Scientific and Statistical Computing,* Vol. 9, No. 4, 1988, pp. 609–638.

[5]"Connection Machine Model CM-2 Technical Summary," Thinking Machines Corp., Cambridge, MA, TR Ha87-4, April 1987.

[6]Hillis, W. D., *The Connection Machine*, MIT Press, Cambridge, MA, 1985.

[7]"*LISP Reference Manual," Version 4.0, Thinking Machines Corp., Cambridge, MA, Oct. 1987.

[8]Dukes, R., "On the Symbolics Artificial Intelligence Programming Environment," AIAA Paper 86-0417. Jan. 1986.

[9]Raj, P., Brennan, J. E., Keen, J. M., Long, L. N., Sikora, J. S., and Singer, S. W., "Three-Dimensional Euler Aerodynamic Method (TEAM)," Vols. 1-3, AFWAL-TR-87-3074, Wright Patterson Air Force Base, OH, Oct. 1987.

[10]Lohner, R., and Parikh, P., "Generation of Three-Dimensional Unstructured Grids by the Advancing Front Method," AIAA Paper 88-0515, Jan. 1988.

[11]Jameson, A., Baker, T. J., and Weatherhill, N. P., "Calculation of Inviscid Transonic Flow Over a Complete Aircraft," AIAA Paper 86-0103, Jan. 1986.

[12]Long, L. N., "A Three-Dimensional Navier-Stokes Method for the Connection Machine," *Proceedings of the NASA-Ames Conference on Scientific Applications of the Connection Machine,* World Scientific, Singapore, 1989, pp. 64–93.

[13]Thompson, P. A., *Compressible-Fluid Dynamics*, McGraw-Hill, New York, 1972.

[14]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259, June 1981.

[15]Agarwal, R. K., and Deese, J. E., "Transonic Wing-Body Calculations Using Euler Equations," AIAA Paper 83-0501, Jan. 1983.

[16]Jameson, A., "Successes and Challenges in Computational Fluid Dynamics," *Proceedings of the AIAA 8th Computational Fluid Dynamics Conference,* Honolulu, HI, June 1987.

[17]Roe, P. L., "Approximate Riemann Solvers, Parameter Vector, and Difference Schemes," *Journal of Computational Physics,* Vol. 43, 1981, pp. 357–372.

[18]Gnoffo, P. A., "Application of Program LAURA to Three-Dimensional AOTV Flowfields," AIAA Paper 86-0565, Jan. 1986.

[19]Raj, P., and Sikora, J. S., "Free-Vortex Flows: Recent Encounters with an Euler Code," AIAA Paper 84-0135, Jan. 1984.

[20]Jameson, A., "Transonic Flow Calculations," Princeton University, Princeton, NJ, MAE Rept. 1651, July 1983.

[21]Parkinson, D., Private communication, 1988.

[22]Moitra, A., Turkel, E., and Kumar, A., "Application of a Runge-Kutta Scheme for High-Speed Inviscid Internal Flows," AIAA Paper 86-0104, Jan. 1986.

[23]Yakhot, V., and Orszag, S. A., "Renormalization Group Analysis of Turbulence. I. Basic Theory," *Journal of Scientific Computing,* Vol. 1, No. 3 1986, pp. 3–51.

[24]Harris, C. D., "Two-Dimensional Aerodynamic Characteristics of the NACA 0012 Airfoil in the Langley 8-foot Transonic Pressure Tunnel," NASA TM-81927, April 1981.

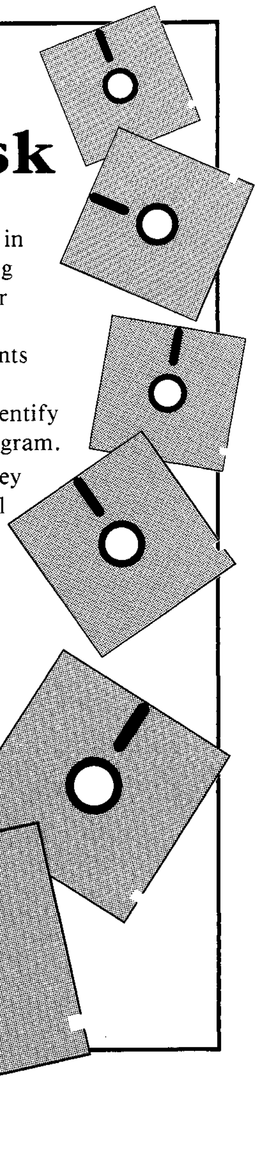
[25]Raj, P., and Long, L. N., "A Euler Aerodynamic Method for Leading-Edge Vortex Flow Simulation," *Vortex Flow Aerodynamics,* Vol. 1, NASA CP-2416, Hampton, VA, Oct. 1985, pp. 263–281.

[26]Emerson, H. F., "Wind-Tunnel Investigation of the Effect of Clipping the Tips of Triangular Wings of Different Thickness, Camber, and Aspect Ratio-Transonic Bump Method," NACA TN-3671, June 1956.

[27]Hinson, B. L., and Burdges, K. P., "Acquisition and Application of Transonic Wing and Far-Field Test Data for Three-Dimensional Computational Method Evaluation," Air Force Office of Scientific Research, AFOSR-TR-80-0421, Washington, DC, March 1980.

[28]Keener, E. R., "Pressure Distribution Measurements on a Transonic Low-Aspect Ratio Wing," NASA TM-86683, Sept. 1985.

Attention Journal Authors: Send Us Your Manuscript Disk



AIAA now has equipment that can convert **virtually any disk** (3½-, 5¼-, or 8-inch) **directly to type**, thus avoiding rekeyboarding and subsequent introduction of errors.

The following are examples of easily converted software programs:

- PC or Macintosh T^{E}X and LAT^{E}X
- PC or Macintosh Microsoft Word
- PC Wordstar Professional

You can help us in the following way. If your manuscript was prepared with a word-processing program, please *retain the disk* until the review process has been completed and final revisions have been incorporated in your paper. Then send the Associate Editor *all* of the following:

- Your final version of double-spaced hard copy.
- Original artwork.
- A *copy* of the revised disk (with software identified).

Retain the original disk.

If your revised paper is accepted for publication, the Associate Editor will send the entire package just described to the AIAA Editorial Department for copy editing and typesetting.

Please note that your paper may be typeset in the traditional manner if problems arise during the conversion. A problem may be caused, for instance, by using a "program within a program" (e.g., special mathematical enhancements to word-processing programs). That potential problem may be avoided if you specifically identify the enhancement and the word-processing program.

In any case you will, as always, receive galley proofs before publication. They will reflect all copy and style changes made by the Editorial Department.

We will send you an AIAA tie or scarf (your choice) as a "thank you" for cooperating in our disk conversion program. Just send us a note when you return your galley proofs to let us know which you prefer.

If you have any questions or need further information on disk conversion, please telephone Richard Gaskin, AIAA Production Manager, at (202) 646-7496.

AIAA