01 Jan 1993

Modeling of Supersonic Combustor Flows using Parallel Computing

Bruce M. McMillin
Missouri University of Science and Technology, ff@mst.edu

Eric Jui-Lin Lu

Larry Reeves

MODELING OF SUPERSONIC COMBUSTOR FLOWS USING PARALLEL COMPUTING

B. McMillin, E. Lu and L. Reeves

CSc-93-14

Department of Computer Science

University of Missouri - Rolla

Rolla, MO 65401 (314)341-4491

Computing Systems in Engineering Vol. 3, Nos 1–4, pp. 217–229, 1992
Printed in Great Britain.

0956-0521/92 $5.00 + 0.00

MODELING OF SUPERSONIC COMBUSTOR FLOWS USING PARALLEL COMPUTING

D. RIGGINS,† M. UNDERWOOD,† B. MCMILLIN,‡ L. REEVES‡ and E. JUI-LIN LU‡
†Mechanical and Aerospace Engineering Department and ‡Computer Science Department, University of Missouri-Rolla, Missouri, U.S.A.

Abstract—Computational Fluid Dynamics (CFD) has matured rapidly in the past 20 years and is now an important tool for analyzing and understanding complex fluid flows. Since 1985, CFD has played a vital role in the study of hypersonic flight. It has provided the capability for scientists and engineers to model both internal and external hypersonic flow-fields. Such flows are often impractical or impossible to analyze in laboratory conditions. In particular, the recent application of CFD to the modeling of internal reacting supersonic combustor flows has significantly advanced the understanding of such flows and has increased confidence in the predictive ability of codes. The purpose of these efforts has been to provide the hypersonic propulsion community with realistic large-scale applications of CFD and to use these solutions in direct support of engineering analysis and design of hypersonic vehicles. Although these applications have been successful to date, expectations and requirements are increasing dramatically for both faster turn-around of solutions and for more detailed and accurate solutions (hence requiring greater computational mesh refinement, more complete chemistry and turbulence models, etc.). In order to begin to meet these requirements, a ten-fold or greater increase in computational efficiency is required, relative to current supercomputing capabilities. This increase can be achieved easily by suitably programming existing CFD technology on existing distributed memory parallel computing machines or multicomputers. This paper presents and analyzes the results obtained to date in an investigation aimed at the application of parallel computing to the simulation of scramjet combustor flow-fields.

NOMENCLATURE

Re	Reynolds number
t	time
T_{comm}	total communication time
t_{hop}	time for message to pass from node to node
t_{send}	time to send a 4-byte word
t_{st}	start-up time
u	x-component of velocity
v	y-component of velocity
x, y	Cartesian coordinates
β	$\Delta x/\Delta y$
ψ	stream function
ζ	vorticity
ω	successive over-relaxation factor

INTRODUCTION

The application of Computational Fluid Dynamics (CFD) for modeling realistic supersonic reacting flows is relatively recent. The bulk of work in this area has been done in the past 5 years as part of an ongoing research effort to develop an advanced air-breathing aerospace vehicle. This hypersonic vehicle will rely on hydrogen-fueled supersonic combustion ramjet engines (scramjets) to produce thrust and enable hypersonic speeds in the atmosphere up to Mach 25 with the eventual goal of obtaining a single-stage-to-orbit capability (Fig. 1). Enormous savings from such technology can be easily projected. Although there are numerous experimental facilities engaged in important studies related to the development of the scramjet, CFD must be used in order to evaluate the performance of the vehicle at actual flight conditions. The numerical modeling of reacting flows characteristic of supersonic combustors has been pioneered by Drummond[1] and others.[2,3] Many of the analyses to date have been simplified two-dimensional representations, mainly due to the enormous computer resources required to solve the entire three-dimensional flow field of the scramjet combustor. Significant advances in understanding of the physics of scramjet combustors as well as conceptual contributions to better engine design have resulted from many of these two-dimensional and simplified studies. In addition to this level of work, full three-dimensional simulations of scramjet combustors have been attempted with some success (as will be subsequently described). However, the capability to model details of the flow in these full simulations is limited on conventional supercomputers by memory limitations and CPU time limitations. The purpose of the ongoing work which is discussed in this paper is to begin to utilize proven CFD technology coupled with existing multi-computers to substantially reduce current solution time for complete Navier–Stokes simulations of supersonic reacting flow. This will allow rapid (yet accurate) engineering parametric studies.

The flow in a supersonic combustion ramjet is governed by the Navier–Stokes equations, coupled with a system of species continuity equations which describe individual species production, convection,

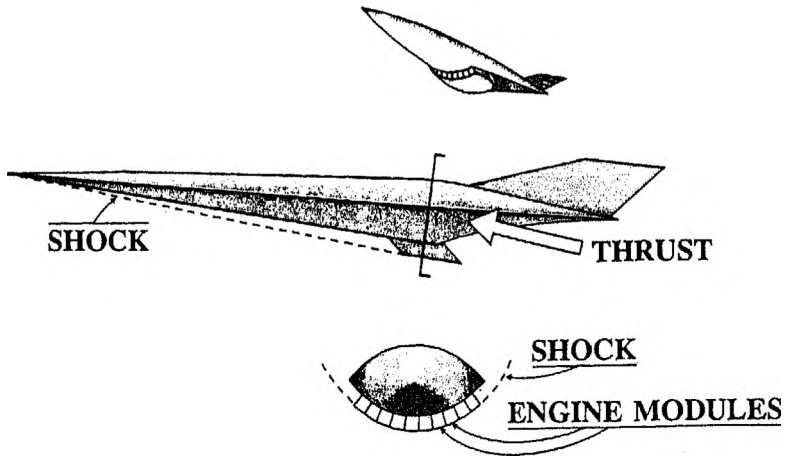

Fig. 1. Generic hypersonic vehicle configuration showing the location of scramjet engines integrated along bottom surface.

and diffusion. A host of techniques exist for the solution of these equations; most are algorithms which rely on either implicit or explicit formulations, are either finite difference or finite volume methods, can be upwind or centrally differenced and are of various order accuracies. Some techniques can be considered to be more accurate, more robust, etc., than others. It is not the purpose of this paper to compare or discuss these issues. It has been concluded from extensive studies of very large-scale CFD applications that the CPU time requirements (and, less restrictively, memory) for all current or foreseeable strategies for solving the Navier–Stokes equations are all deplorable on conventional computers. One notable exception to this situation is when the governing Navier–Stokes equations are reduced in complexity such that the solution can be obtained by space marching (i.e. a parabolized formulation). Unfortunately, the flow in a supersonic combustor is highly elliptical, particularly around the injection ramps or wall jet orifices. Although the downstream section of the combustor can be reliably solved using marching codes (since the Mach number is fairly high and upstream interaction is limited), the zone of the combustor itself should be solved elliptically for input into the downstream region. Some workers have attempted to "model" the elliptic regions so that the fast parabolized codes can be used throughout the combustor. These efforts have been moderately successful for engineering predictions of downstream bulk parameters. However, details of modeling techniques are problem-specific such that reliability is questionable when these techniques are extrapolated to other flight conditions or geometries. An additional incentive for developing high performance computing techniques for use in the research and design of scramjet engines is the characteristically small time step which must be used in the solution procedure due to the presence of finite-rate reaction (this is due to the chemistry time scale being much smaller than the fluid time scale).

This paper presents a typical solution of a scramjet combustor flow-field, discusses the current resource requirements on conventional supercomputers, then explores the feasibility for substantially reducing these resource requirements using parallel computing. A successful preliminary investigation of a simple classic fluids problem (the so-called driven cavity problem) is presented utilizing straightforward domain decomposition techniques. Timings obtained on a 16-node Intel iPSC/2 (a multi-computer with hypercube architecture) indicate an asymptotic approach to linear-speed-up as the grid size increases. Also investigated in this part of the study was the utility of multi-grid in which processors work appropriately on different grid refinements. Finally, a three-dimensional compressible Navier–Stokes research code has been written for multi-processor hypercube architectures with an emphasis toward the iPSC/2. The formulation of this code and some preliminary results for test cases are presented. The intermediate goal of this work is to eventually include both multi-species capability as well as finite-rate reaction mechanisms in this code in order to rapidly solve combustor applications on existing massively parallel machines. The overall objective is to be able to produce fast and accurate solutions of complex supersonic reacting flows for design support and configuration evaluation.

SCRAMJET COMBUSTER CALCULATIONS

The following sections detail a typical scramjet CFD solution obtained on a CRAY-YMP. The purpose of presenting these results is to review current solvers, show the solution techniques used for these types of flow, illustrate the flow physics involved, and finally to discuss the resources required. Following this will be the sections involving parallel computing.

Experimental configuration

The experiment which has been modeled numerically for this work was conducted in the

GASL(HYPULSE) facility. This is a high energy pulse facility utilizing an expansion tube (previously the Langley Expansion Tube) with about 0.5 ms test duration.[4] The combustor section which has been modeled in this work has two side-by-side swept-sided ramp injectors. A schematic of these injectors is shown in Fig. 2. Hydrogen is injected from the bases of these ramps. The 10° compression ramps are somewhat similar to the ramp configurations tested and described by Northam *et al.*[5] and studied numerically by Riggins *et al.*[6] The combustor duct is rectangular in cross-section (2.54 cm × 5.08 cm wide) and is 70 cm long. The base of the ramps (the injection location) is approximately 18 cm from the duct inflow. The sides of the ramps are swept back 10° in order to generate additional vorticity which will contribute to fuel mixing. The enthalpy of the flow corresponds to a Mach 13.5 flight condition, although the actual Mach number at the duct entrance is around 6. The hydrogen is injected at Mach 1.7 with a fuel equivalence ratio of 1.0.

Computational approach

The computational approach taken in this work has been fully described in previous studies.[7] Upstream (from the test section entrance to the leading edge of the ramp injector) the parabolized version of the code was used to generate a fixed inflow for the near-field of the near-field ramp region. The outflow plane from this upstream PNS solution is used as the fixed inflow into the elliptic (jet) region where the full Navier–Stokes code is used. Due to the low downstream angle of the injection, the exit of the elliptical region was chosen relatively close to the ramp base (about 1 ramp length downstream) since flow separation around and behind the ramp was minimal. The flow in the near-field is quasi-steady at best and solution convergence is defined as a matrix of criteria which includes unchanging mean values of wall pressures and mixing efficiencies, mass flow conservation (both fuel and air), and relatively unchanging values of one-dimensionalized parameters describing the flow.[8] Overall mass flow at the elliptical region exit is monitored to ensure a level of less than 0.5% conservation error. When these criteria are met, the elliptic region is considered converged and the outflow is then passed to the parabolized code for the downstream computation. This technique works well for both mixing and reacting high-speed combustor modeling and has successfully simulated numerous experimental flows. The ability to run the parabolized code in the downstream section vastly decreases computer requirements. It should be noted that shock-boundary layer separation cannot be predicted in this region. If such interactions occur, this method is not appropriate. However, results to date indicate that boundary layer separation is not a significant issue in these high Mach number turbulent flow fields.

No grid convergence studies were explicitly undertaken as part of this work due to resource limitations. However, previous work[9,10] has shown convergence for parameters such as fuel mixing and wall pressures when using similar spatial resolution and geometries.

Code descriptions

The NASA Langley Research Center (LaRC) SPARK family of codes were used in this part of the work. Drummond *et al.*[11] formulated the initial

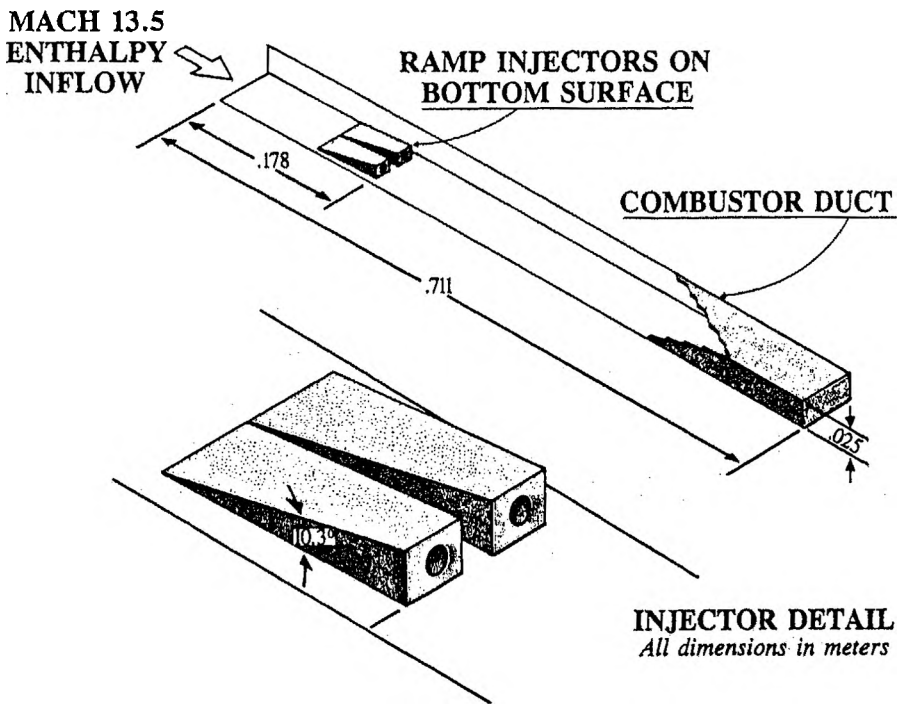

Fig. 2. Schematic of scramjet combustor duct modeled with swept ramp injectors.

two-dimensional elliptic explicit finite difference version of the code with hydrogen–air chemistry capability. This full Navier–Stokes code was extended to three dimensions by Carpenter[12] and has been further modified by Kamath[13] into a separate code which solves the parabolized Navier–Stokes equations. There has been extensive validation of these codes as detailed by McClinton[14] for supersonic combustor flow-fields. Although the codes have choices of either a second-order accurate MacCormack based solver or a fourth-order accurate compact MacCormack solver, the fourth-order solver has been used exclusively in this part of the work. The solution presented here has been run in a local time-stepping mode in order to facilitate the effort. Also, the codes included an internal grid generation capability developed by Smith and Weigel[15] which was used in this analysis.

The SPARK codes have a variety of turbulence models including the recent addition of two-equation models. For this work, however, the Baldwin–Lomax algebraic model has been used to generate the turbulent viscosity. This model is easy to use and has been successfully applied for mixing prediction in injector flows, although it is not physically accurate in the near-field of the injector. A turbulent Schmidt number of 0.5 was used for all cases in this study. Jet-vortex induced viscosity was limited to 1000 times the local laminar viscosity in order to prevent unphysically high diffusion in the jet structure. The Baldwin–Lomax model is used through the jet itself in order to model the turbulent diffusion in this region. It should be emphasized that this method, as modified and used in these injection studies, only represents a modeling technique to predict downstream mixing and mean flow. Details of the highly complex physical turbulence field are not accurately predicted by this model. The SPARK codes have a generalized chemistry package wherein the source term in the species continuity equations can be treated implicitly. The capability exists in these codes to include any number of reactions. The reacting portion of this work was performed using the seven-reaction seven-species finite rate model, which is a subset of the model used by Drummond *et al.*[11] For mixing cases, the source term is simply set to zero in the species equations. For all calculations, SPARK simply carries the nitrogen as an inert species, with its value determined by subtracting the summation of all other species from unity (on a mass fraction basis).

Grid and boundary conditions

Figure 3 shows views (slices) of the grid used for the elliptic near-field of the combustor. Figure 3a is a lateral view of the grid (looking down at the

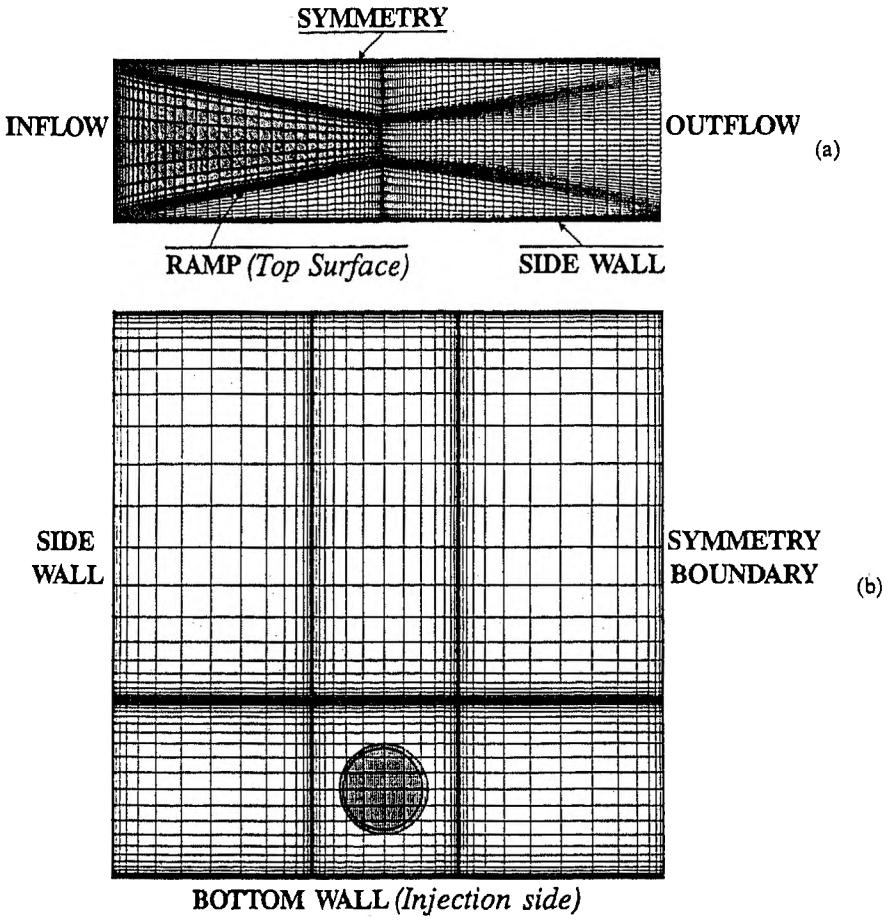

Fig. 3. Views of grid used in combustor calculations. (a) Lateral view (ramp shaded). (b) Cross-flow plane at injector ramp bases (ramp shaded).

injector). Figure 3b is a cross-section of the grid at the ramp base. The grid dimensions for this region are 81 (axial), 61 (vertical) and 61 (horizontal). Grid nodes which fall inside the ramp itself are shaded in this figure. The domain extends only from the duct side wall to the duct centerline and from top wall to bottom wall such that only one half of the duct is actually modeled. Flow symmetry is applied at the duct centerline. No slip is applied on the top, bottom, and side of the duct as well as on the interior ramp surfaces. Note that the grid implementation is such that all boundary conditions are smoothly implemented (i.e. ramp boundaries are described by single grid lines); although there are some grid nodes blocked out or overwritten inside the ramp, this technique has proved successful for this type of flow-field in many previous studies. The jet orifice is modeled as nearly rectangular (due to the Cartesian grid). Approximately 50 points define the injector, which had fixed injection velocity, pressure and temperature conditions. Although modeling the injector as rectangular and with uniform outflow is a source of error (in reality there is considerable three-dimensionality in such flows), this technique has resulted in good agreement with experimental data for lower Mach number flows. The grid was clustered on all four sides in order to resolve the boundary layers as well as the symmetry boundary. Wall temperature was held constant at 300 K. The flow was considered to be laminar upstream of the ramp but was tripped to turbulent through the ramp region on the injection wall. The flow was tripped at the approximate location of the bow shock impingement at both non-injection and side walls.

Flow-field description for supersonic combustor

Figure 4 shows a color view of a swept ramp solution. Hydrogen mass fraction contours on the ramp base and three downstream cross-flow planes are shown. In addition, pressure contours on the inflow plane, symmetry plane and bottom wall are shown. The leading edge shock is plainly seen. The warping of the fuel jet by counter-rotating vortices shed from the ramp surface is evident. This near-field vorticity-driven flow distortion contributes to a rapid mixing of fuel and oxidizer within the elliptic zone. Figure 5 shows cross-flow velocity vectors on the same plane. The locations of the large vortices are seen to be influenced by the presence of the viscous side wall; the left vortex (near the side wall) is beginning to roll above and over the (right) vortex. The influence of the wall, then, is to push the jet (and subsequent downstream reaction) toward the duct centerline. This effect has been observed experimentally in lower Mach number studies performed at Langley; computations following these experiments at the time did not illustrate this phenomenon due to the lack of side wall viscous modeling. The large influence of the walls is evident from the distortion of the vortices.

Figure 6 depicts the reaction-generated water contours on the same downstream cross-flow plane. Notice that the stoichiometric zone of water production is located around the periphery of the hydrogen core. Figure 7 shows hydrogen contours on a

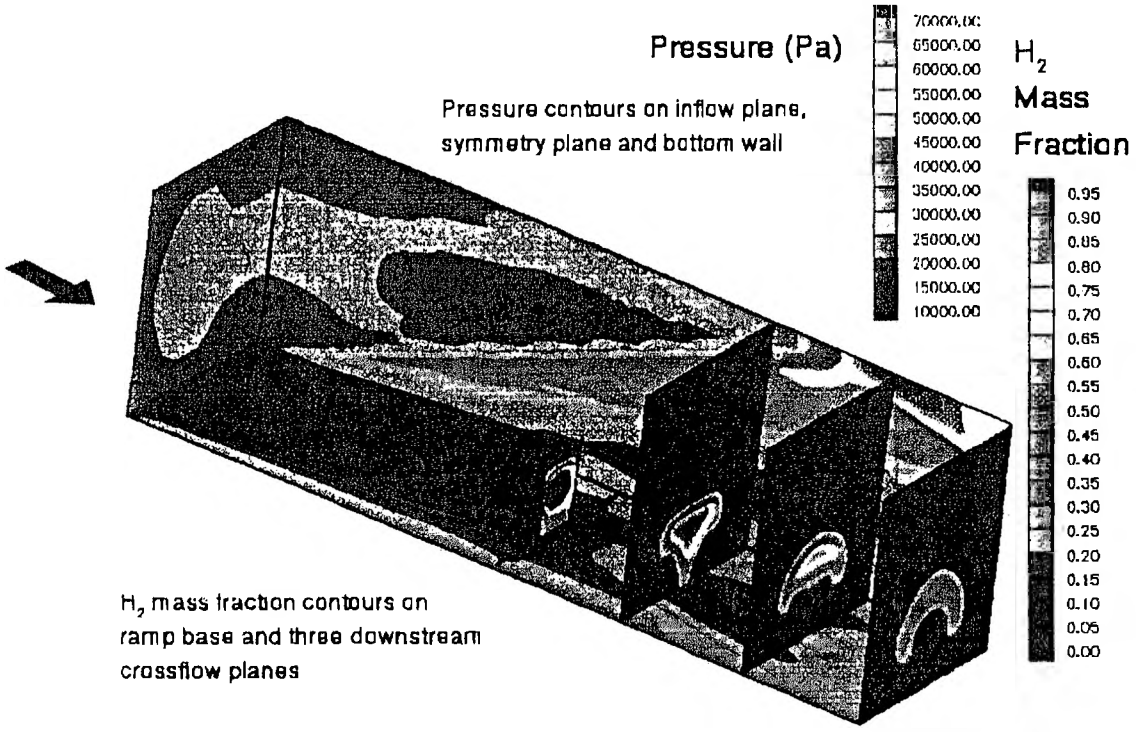

Fig. 4. Hydrogen mass fraction contours on ramp base and three downstream cross-flow planes; pressure contours on inflow phase, symmetry plane and bottom wall.

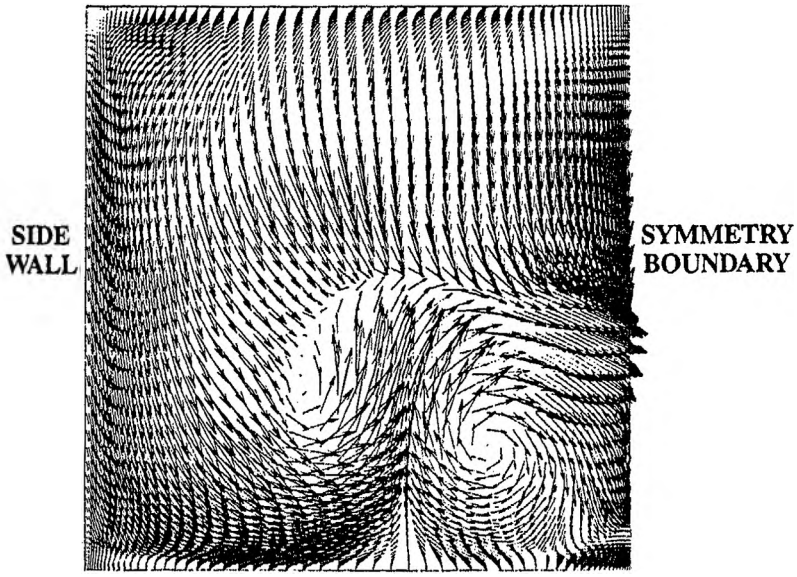

Fig. 5. Cross-flow velocity vectors on plane at end of elliptic domain.

longitudinal (side) plane through the ramp centerline. The bow shock from the leading edge of the ramp intersects the top wall, as shown by the ramp centerline pressure contours in Fig. 8, reflects down, and passes through the plume of the jet, thereby increasing fuel mixing (via vorticity generation). The core of the jet and associated shock and expansion structure can also be seen. A wall pressure comparison with experimental data for a related swept ramp case (Mach 17 flow enthalpy) is shown in Fig. 9. Agreement is excellent, considering the modeling issues involved such as turbulence and kinetics. Although no explicit experimental data exist for describing mixing for these swept ramp injectors, numerous and extensive experimental and computational studies have been performed for simple flush-wall jets in high Mach number flows. A comparison of mixing (i.e. decay rate of maximum concentration of injectant) is taken from earlier work by Riggins and McClinton,[16] and is shown in Fig. 10. Agreement is seen to be excellent, especially in the far-field. The degree of mixing of such flows at the combustor exit is an important design consideration.

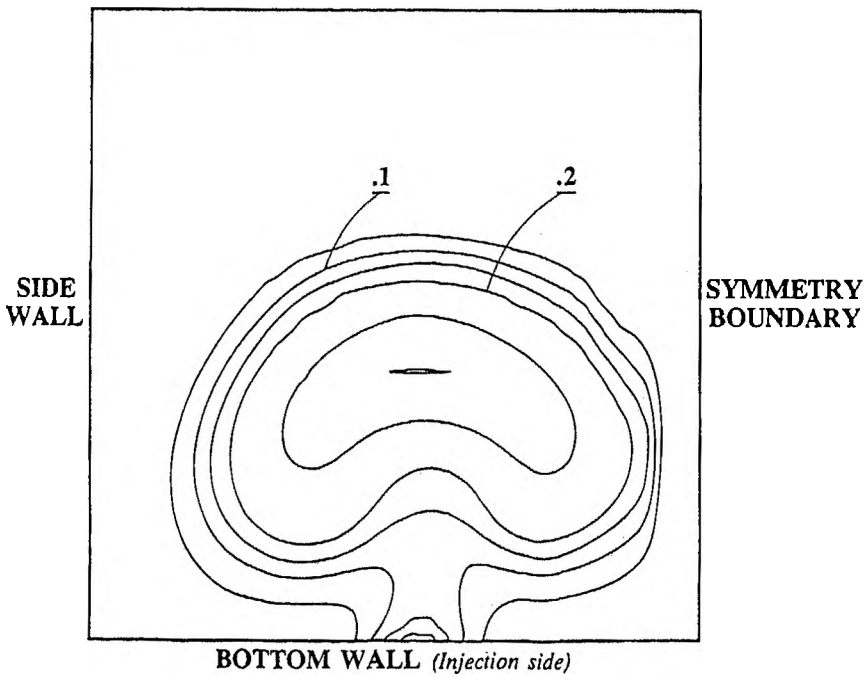

Fig. 6. Reaction-generated water contours on cross-flow plane at end of elliptic domain.

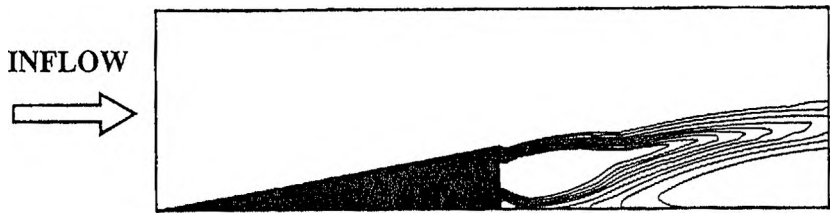

Fig. 7. Hydrogen contours on ramp centerline longitudinal plane.

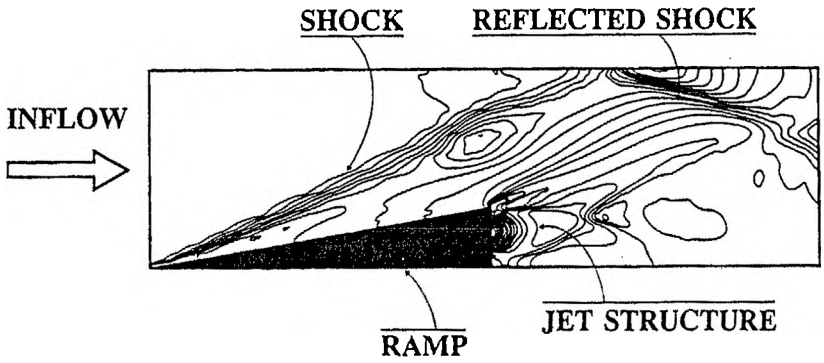

Fig. 8. Pressure contours on ramp centerline longitudinal plane.

Grid and computer resource requirements (CRAY YMP)

The total number of grid points needed to perform a near-field elliptic solution such as that described above is over 300,000 nodes. The number of explicit iterations required varies from 20,000 (mixing) to 60,000 or more (reacting). Memory requirements for mixing cases are of the order of 12 million words; for reacting cases they are about 20 million words. The CPU time required varies from 15 to 20 h for mixing cases up to ten times that (100–200 h) for reacting cases. Due to resource limitations and other users on supercomputers, realistic computations for one configuration can take up to one month or more to complete. Also, left unresolved in such a solution are (1) grid convergence (far more grid resolution would be required to adequately predict skin friction and/or wall heat transfer), and (2) low frequency oscillations in the flow-field itself which can be significant but are difficult to capture due to the length of time necessary to establish a period. Clearly, more time-efficient means of computing these flows are necessary for useful design-oriented computational applications. Current CFD technology, including (to a degree) modeling techniques, can suffice for engineering calculations of design information (wall pressure, mixing, combustion, etc.). Hence, enormous savings can be realized if current CFD techniques can take advantage of multi-processor machines such that work can be equitably split between many processors to speed up the calculations and use individual node memories to ameliorate the storage requirements. A simple strategy for such an application is to divide the domain into a number of sections (either strips or

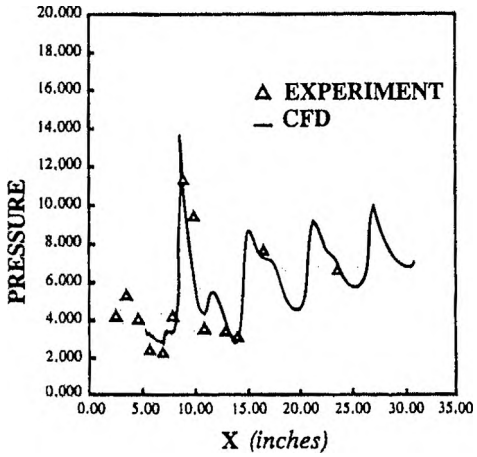

Fig. 9. Experimental versus numerical wall pressure comparison for swept ramp calculation.

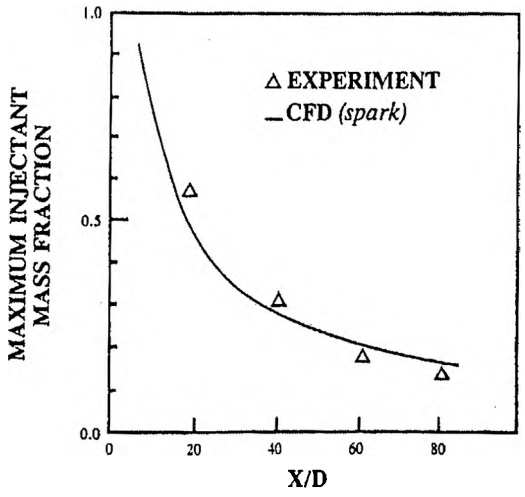

Fig. 10. Decay rate of maximum concentration of injectant for flush-wall injection.

blocks) and allow each processor (node) to perform computations on its given block while passing and receiving information to and from its neighboring nodes. Ignoring the loss inherent in message passing between nodes, this system should conceptually be able to exhibit linear speed-up of a complete solution, i.e. a 128 processor machine should ideally perform approximately 128 times faster than a single processor. Since CFD problem decompositions scale upward, larger problems can be made to use more processors. As the availability of these multiprocessor machines increases, CFD codes which are structured to take advantage of these speed-ups should prove very useful in all areas of research and design.

EFFECT OF DOMAIN DECOMPOSITION ON PARALLEL SPEED-UP OF VORTICITY TRANSPORT–STREAM FUNCTION EQUATIONS (DRIVEN CAVITY)

To illustrate the parallel implementation of flow solvers, a simple pair of non-linear differential equations was solved using an explicit method on an Intel iPSC/2. The iPSC/2 has an MIMD (multiple instruction, multiple data) hypercube architecture, where each of the processors consists of an Intel 80386 cpu with an Intel 80387 math co-processor and 8 Mb (megabytes) of memory per processor. The effects of domain decomposition on parallel speed-up were examined. The results indicate that the speed-up observed seemed to be relatively insensitive to the method of decomposition. Possible reasons include the relative message sizes due to the methods of decomposition and the fact that an explicit solution method was used.

Problem description

In order to study the effects of domain decomposition, the vorticity–stream function formulation of low speed flow was chosen. The physical domain chosen was a driven cavity, as shown in Fig. 11. The pair of non-linear coupled differential equations that describe this flow are easily solved sequentially using a standard second-order central differencing scheme. Central differencing calculates the new values at a particular point by taking a weighted average of the values of the nearest neighbors, as shown in Fig. 12, where the weights are dependent on the flow patterns.

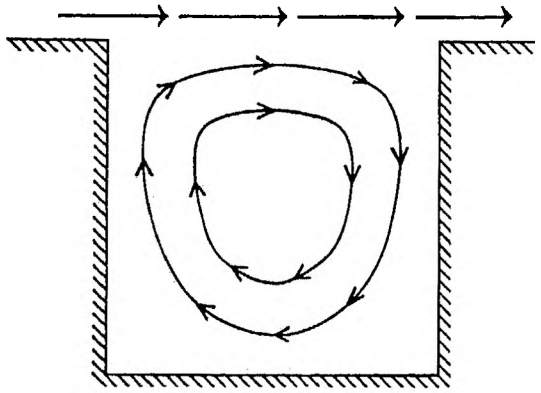

Fig. 11. Physical domain for an Re = 10 flow.

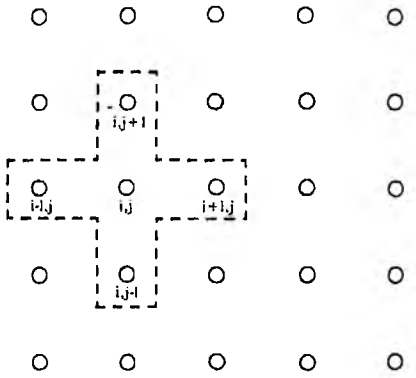

Fig. 12. Points whose values are needed for calculation of values at point (i, j).

The governing equations for this simple flow are as follows:

$$\zeta = -\Delta^2\psi \tag{1}$$

$$\frac{\partial\zeta}{\partial t} + \frac{\partial}{\partial x}(u\zeta) + \frac{\partial}{\partial y}(v\zeta) = \frac{1}{\mathrm{Re}}\Delta^2\zeta \tag{2}$$

where

$$u = \frac{\partial\psi}{\partial y} \quad \text{and} \quad v = -\frac{\partial\psi}{\partial x}.$$

These two equations represent the flow conditions in the physical domain. Lines of constant stream function, ψ, value are parallel to the local flow, while the vorticity, ζ, is a measure of the local shearing rate, or swirl, in the flow.

These equations were solved using the Extrapolated Liebmann's Method with SOR. The resulting discrete equations are as follows:

$$\frac{\partial\psi}{\partial y} = 0$$

$$\psi_{ij}^{k+1} = \psi_{ij}^{k} + \frac{\omega}{2(1+\beta^2)}[\psi_{i+1,j}^{k} + \psi_{i-1,j}^{k+1} + \beta^2(\psi_{ij+1}^{k} + \psi_{ij-1}^{k+1}) - 2(1+\beta^2)\psi_{ij}^{k} - \zeta_{ij}\Delta x^2] \tag{3}$$

$$\zeta_{ij}^{n+1} = \zeta_{ij}^{n} + \Delta t\left[\frac{-u_{i+1,j}^{n}\zeta_{i+1,j}^{n} - u_{i-1,j}^{n}\zeta_{i-1,j}^{n}}{2\Delta x} + \frac{-v_{ij+1}^{n}\zeta_{ij+1}^{n} - v_{ij-1}^{n}\zeta_{ij-1}}{2\Delta y} + \frac{1}{\mathrm{Re}}\left(\frac{\zeta_{i+1,j}^{n} + \zeta_{i-1,j}^{n} - 2\zeta_{ij}^{n}}{\Delta x^2} + \frac{\zeta_{ij+1}^{n} + \zeta_{ij-1}^{n} - 2\zeta_{ij}^{n}}{\Delta y^2}\right)\right], \tag{4}$$

where ω is the over-relaxation factor and $\beta = \Delta x/\Delta y$. The superscript k indicates the current iteration value and n is the value at the current time. The boundary values for ζ are calculated by using first-order accurate away-from-the-wall equations:

$$\zeta_{i,w} = -\frac{2}{\Delta y^2}(\psi_{i,w} - \psi_{i,w+1})\left[+\frac{u_{i,w}}{\Delta y}\right] \tag{5}$$

$$\zeta_{w,j} = -\frac{2}{\Delta x^2}(\psi_{w,j} - \psi_{w+1,j}). \tag{6}$$

In Eqs (5) and (6), w is the location of the boundary, and the term enclosed in brackets is only valid at the top of the cavity.

The standard solution method is to take an initial guess of the values of u, v and ζ, along with a Δt appropriate for the fineness of the grid, and iterate Eq. (4) once. These values are then used to iterate Eq. (3) to convergence, update the values of u and v, calculate the boundary values for ζ, then repeat the process until the values of ζ and ψ have both met the desired convergence criteria.

Domain decomposition and effects on communication

The two different domains examined are illustrated in Figs 13 and 14. In both cases, a particular domain section is assigned to a separate processor for parallel execution. Figure 13 illustrates the strip method, where the domain is split horizontally into equal height regions and the width of the strip is the same as the total width of the domain. Figure 14 shows the grid method, where the domain is split into an $n \times n$ grid and each element of the grid has an $n/P \times n/P$ array of domain points, where P is the width of the domains and P^2 processors are used.

From examining Figs 12–14 it can be seen that if the current values being calculated belong to a point on the edge of the local section, the local node will need to communicate with its neighbor nodes to obtain the values needed for central differencing. All of the affected points are indicated in Figs 13 and 14, along with an example of the communication interactions.

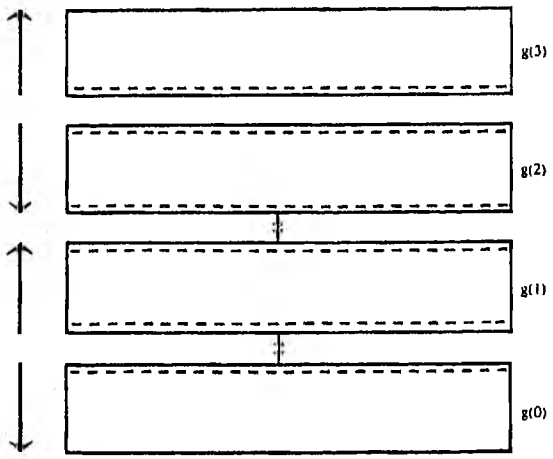

Fig. 13. Example of communication directions between adjacent strips, where g0 is the ring grey code ordering on a hypercube. The arrows to the left of the figure indicate the iteration direction on a given processor.

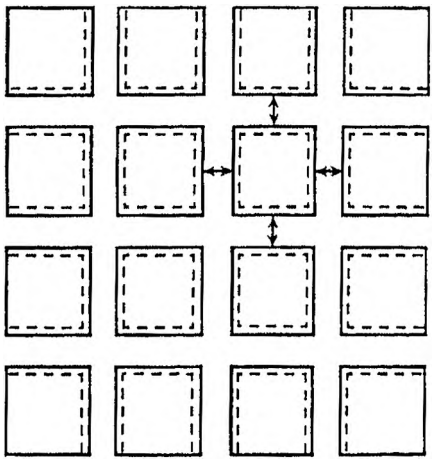

Fig. 14. Example of communication directions between adjacent grid sections, where adjacency is decided with a grey code ordering.

In the strip method, each node updates its values row by row in the direction indicated on the left of Fig. 13. As an edge row is calculated, an asynchronous message is initiated with its neighbor to send/receive the new values for the next iteration. Since the messages are asynchronous, the nodes can continue to calculate values while the I/O processor handles the messages. Since alternate strips are iterating in different directions, as long as the problem size is sufficiently large, a node will not have to wait for its neighbor's values after it finishes calculating its own. In order to check convergence, each node checks its own convergence and then the nodes exchange values to determine the status of global convergence.

In the grid method (Fig. 14), all communications are again asynchronous. The major difference between this method and the strip method is that most of the nodes will transmit more messages. Each node begins by calculating its local boundary values and begin transmissions while it computes the values for the interior points. Since this particular problem is relatively stable, the order in which the values are calculated does not greatly affect convergence rates, so that the calculation order does not significantly affect the results.

The general equation for the time to pass messages between nodes on the iPSC/2 is:

$$T_{\text{comm}} = t_{\text{st}} + k \times t_{\text{send}} + n \times t_{\text{hop}} \tag{7}$$

where T_{comm} is the total communication time for k 4-byte words and n is the number of nodes the message passes through between source and destination.

For this project, all substantial messages were between nearest neighbors, so $n = 0$. According to

Table 1. Runtimes for the strip method

Strip method Domain		Time	
Width	Nodes	min	s
6	1		2.65
12	2		33.48
	1		43.50
24	8	5	15.56
	4	7	59.43
	2	11	40.04
	1	20	29.51
36	8	21	46.92
	4	29	16.19
	2	48	37.17
	1	91	5.63
48	16	34	53.36
	8	56	20.45
	4	101	3.21
	2	178	10.55

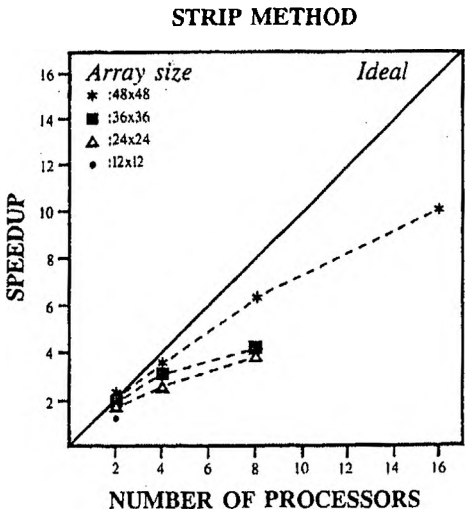

Fig. 15. Relative speed-up of multiple processors versus one processor using the same algorithm. Decomposition was by strips.

Boman and Roose,[17] for messages under 100 bytes, $t_{st} = 350\,\mu s$. and $t_{send} = 0.8\,\mu s$, while for longer messages, $t_{st} = 660\,\mu s$ and $t_{send} = 1.44\,\mu s$. Hence, the major overhead of message passing is the actual start-up of the message, when messages are approximately the same length. With this consideration, it would appear that the strip method would show gradually increasing speed-up as the size of the problem increases.

Results (driven cavity)

The execution times of several runs are given in Tables 1 and 2. Since the solution method is explicit, in order to meet the convergence criteria, Δt must decrease proportionally with the number of points. Also, due to the structure of the physical domain, the number of points required increases as the square of the width. Thus, an exponential increase in runtime is expected.

Figures 15 and 16 show the values for parallel speed-up for the strip and grid methods, respectively. These values are not true speed-up in that the sequential values are assumed to be approximately the same as running the particular method on a single processor.

According to Eq. (7), the strip method should outperform the grid method since most of the grid nodes send $1\frac{1}{2}$–2 times as many messages as the strip nodes. This is moderated, however, since during the actual sending time, the individual nodes continue to do computations, thus avoiding idle time. Also, since the grid method sends smaller messages, it falls in the shorter message group, which has a lower start-up time. Due to the exponential increase in running time,

Table 2. Runtimes for the grid method

Grid Method Domain		Time	
Width	Nodes	min	s
6	1		2.91
12	4		26.39
	1		44.52
24	16	5	57.30
	9	5	9.38
	4	6	56.45
	1	20	56.22
36	16	11	18.12
	9	16	21.36
	4	28	27.64
	1	92	54.30
48	16	34	20.97
	8	59	6.35
	4	99	33.41
	1	349	22.22
96	16	303	32.92

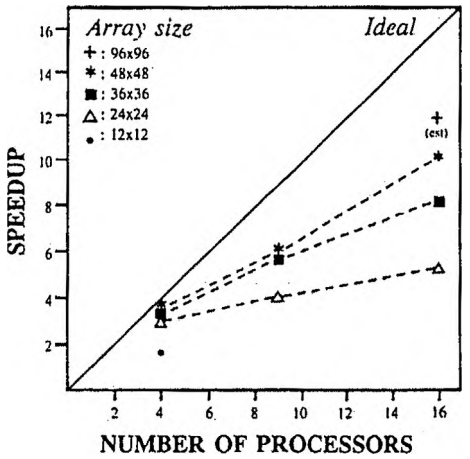

Fig. 16. Relative speed-up of multiple processors versus one processor using the same algorithm. Decomposition was by grid. The speed-up of the estimated point was extrapolated from the times for a single processor.

it is not feasible to run cases large enough for the two methods to have messages that fall in the same start-up group.

An additional possible explanation is the numerical method. Since the computation method is based on explicit updating, the nodes can rearrange the computation order in order to allow more communication/computation overlap. If the numerical method were implicit, this would not be as simple.

Multi-grid

From the results of this section, it is obvious that for an explicit method to be optimized, additional methods of speed-up need to be formulated. One possible acceleration is the multi-grid method.[18] In standard single-grid methods, the low frequency errors are very slow to damp out of the solution, since adjacent points have nearly the same error value. However, multi-grid methods send error values to coarser grids where the frequency of the error is magnified. Table 3 shows the results of the speed-up associated using a standard V cycle multi-grid on this problem with 16 processors. The preliminary results are encouraging and suggest that multi-grids should be useful in parallel computing.[19]

THREE-DIMENSIONAL PARALLEL COMPUTING RESULTS (COMPRESSIBLE FLOW)

As the next step in this effort, a three-dimensional compressible Navier–Stokes code has been written in the C programming language (with parallel extensions). This code has been written specifically to take advantage of multi-computer capability and is fully portable. The code is modular and has been validated to date for three-dimensional Euler flow. It is primarily a research tool and not a production code. As more modules are added and as the code or its derivatives are more fully optimized, the code will be able to solve complex internal reacting flows. For rapidity and ease of programming and due to the relatively successful record of current explicit finite difference codes in the area of supersonic combustion, this code relies on a finite difference formulation and uses a simple explicit second-order unsplit MacCormack algorithm. As performed in the driven cavity (incompressible) problem, based on the computer architecture and the characteristics of the equations, the parallel implementation involves simple domain decomposition with explicit message passing between processors. In the domain decomposition, each processor iterates upon the same number of grid points. The interior boundary conditions (those between processors) were handled by overlapping the boundaries. Consider two processors, one on the left and the other on the right. Moving from left to right, the right-most interior points of the left processor correspond to the left boundary of the right processor. Likewise, the right boundary of the left processor corresponds to the left-most interior points of the right processor. In this case, whenever boundary conditions need to be set, the interior points of each processor are sent to the corresponding boundary points of the other processor.

Table 3. Runtimes for multi-grid cases

Timing comparisons (times in min)		
Domain size	Red–black method	Multi-grid method
24 × 24	6.170	3.711
36 × 36	16.790	11.410
48 × 48	48.902	29.132
72 × 72	171.937	120.769
96 × 96	413.728	342.398

Figure 17 shows the pressure contours for Mach 2.5 airflow over a 10° compression ramp (two-dimensional) generated using this code. The grid for this test case is 122 (vertical) × 62 (axial). The shock is somewhat smeared due to the lower order of accuracy of the code and grid resolution. However, in general, the limited physics of this compressible and inviscid flow are captured satisfactorily. Figure 18 shows timings for solutions obtained on the iPSC/2 for two different cases. The first case is a return of free-stream conditions (for an initially large skewing of the flow). The grid for this case was 62 by 62. The second case shown is the shock problem discussed above. Note that nearly linear speed-up has occurred (based on a four-node to 16-node relationship). These results are very encouraging and provide the basis for continuing development of the parallel three-dimensional code for use in modeling supersonic reacting flows.

CONCLUSIONS

The ability of current three-dimensional CFD codes and modeling techniques to successfully provide engineering-level information for complex scramjet flow-fields has been demonstrated. However, the usefulness of CFD in this area is currently severely limited by (primarily) CPU time requirements for solutions on adequate grids and (secondly) memory requirements. In addition, grid and modeling requirements are only projected to increase as higher fidelity flow solutions are demanded by the design community. Although supercomputers are increasing in speed-per-processor, the major increase in machine efficiency is projected to occur due to the availability of large numbers of processors per machine and the scalability of the problems on these machines. The purpose of the ongoing effort documented here is to specifically provide engineering CFD tools with parallel computing capability for the analysis of supersonic reacting flows. As a first step, a simple incompressible CFD problem (utilizing the streamfunction–vorticity transport equations) was programmed and performance analyzed on up to 16 processors of an Intel iPSC/2 (utilizing a simple domain decomposition). Results were promising: an approach toward linear speed-up as grid size was

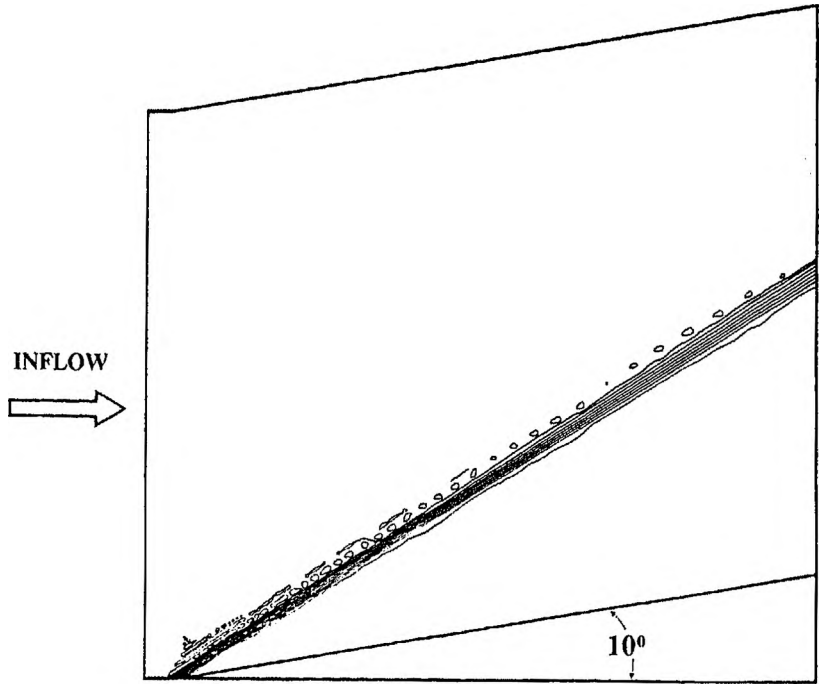

Fig. 17. Pressure contours for 10° compression ramp (free-stream Mach number = 2.5).

increased was observed. Additionally, multi-grid capability was added to this code such that various processors worked on various grid levels. This multi-grid study also resulted in a significant decrease in computing time required.

A compressible three-dimensional Navier–Stokes code was then written specifically for parallel hypercube architecture machines and has been tested for several sample cases. The results for a simple shock problem (generated by a 10° compression wedge) exhibit nearly linear speed-up. This indicates that the domain decomposition method used is very efficient, with little loss due to communication relative to the overall time required for a solution on a realistic grid. This code is currently being expanded to include viscous terms, multiple species, chemistry, and higher-order algorithms. It is highly portable to other machines (being entirely written in C). The ultimate goal of this continuing investigation is to provide very large increases in computational efficiency for internal reacting computational simulations.

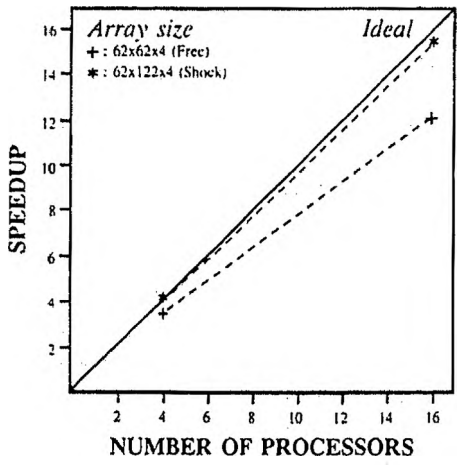

Fig. 18. Relative speed-up of multiple processors versus one processor using the same algorithm for return-of-free-stream (Free) and compression shock (Shock).

Acknowledgements—The authors would like to thank the Hypersonic Technology Office at NASA Langley Research Center for supporting a portion of this work. In particular, thanks are due to Charles McClinton for his continuing support, interest and encouragement.

REFERENCES

1. J. P. Drummond, M. H. Carpenter and D. W. Riggins, "Mixing and mixing enhancement in supersonic reacting flowfields", in *High-Speed Flight Propulsion Systems* (edited by S. N. B. Murthy and E. T. Curran), pp. 383–455, Vol. 137, Progress in Astronautics and Aeronautics, American Institute of Aeronautics and Astronautics, Washington, DC, 1991.
2. P. A. McMurty, W.-H. Jou, J. J. Riley and R. W. Metcalfe, "Direct numerical simulations of a reacting mixing layer with chemical heat release", AIAA Paper 85-0143, January 1985.
3. S. Menon, J. D. Anderson, Jr and S. I. Pai, "Stability of a laminar premixed supersonic free shear layer with chemical reactions", *International Journal of Engineering Science* 22 (4), 361–374 (1984).
4. R. J. Bakos, J. Tamango, O. Rizkalla, M. V. Pulsonetti, W. Chinitz and J. I. Erdos, "Hypersonic mixing and combustion studies in the GASL HYPULSE facility", AIAA Paper 90-2095, July 1990.
5. G. B. Northam, I. Greenberg and C. S. Byington, "Evaluation of parallel fuel injector configurations for supersonic combustion", AIAA Paper 89-2525, July 1989.
6. D. W. Riggins, C. R. McClinton and J. P. Drummond, "A numerical study of mixing enhancement in a supersonic combustor", AIAA Paper 90-0203, January 1990.

7. D. W. Riggins and C. R. McClinton, "A computational investigation of flow losses in a supersonic combustor", AIAA Paper 90-2093, July 1990.
8. D. W. Riggins and C. R. McClinton, "Analysis of losses in supersonic mixing and reacting flows", AIAA Paper 91-2266, July 1991.
9. G. Bobskill, R. Bittner, D. W. Riggins and C. R. McClinton, "CFD evaluation of Mach 17 HYPULSE scramjet combustor data", AIAA Paper 91-5093, December 1991.
10. M. Mao, D. W. Riggins and C. R. McClinton, "Numerical simulation of transverse fuel injection", CFD Symposium on Aeropropulsion, April 1990.
11. J. P. Drummond, R. C. Rogers and M. Y. Hussaini, "A detailed numerical model of a supersonic reacting mixing layer", AIAA Paper 86-1427, June 1986.
12. M. H. Carpenter, "Three-dimensional computations of cross-flow injection and combustion in a supersonic flow", AIAA Paper 89-1870, June 1989.
13. H. Kamath, "Parabolized Navier–Stokes algorithm for chemically reacting flows", AIAA Paper 89-0386, January 1989.
14. C. R. McClinton, "CFD support of NASP design", AIAA Paper 90-3252, September 1990.
15. R. E. Smith and B. L. Weigel, "Analytic and approximate boundary fitted coordinate systems for fluid flow simulations", AIAA Paper 80-0192, January 1980.
16. D. W. Riggins and C. R. McClinton, "A computational investigation of mixing and reacting flows in supersonic combustors", AIAA Paper 92-0626, January 1992.
17. L. Boman and D. Roose, "Benchmarking the iPSC/2 hypercube multiprocessor", *Concurrency: Practice and Experience*, Vol. 1, pp. 3–18, September 1989.
18. A. Brandt, *Multigrid Adaptive Solution to Boundary Value Problems*, Math. Comp., Vol. 31, pp. 333–390, 1977.
19. S. Franks, B. McMillin and R. Khanna, "PAFMV—Pairwise asynchronous multigrid", Proceedings of the 1990 International Conference on Parallel Processing, pp. I-388–I-392, August 1990.