IN-59
14927
p20

NASA Technical Memorandum 104446

Parallel Computing Using a Lagrangian Formulation

(NASA-TM-104446) PARALLEL COMPUTING USING A LAGRANGIAN FORMULATION (NASA) 20 pCSCL 12A

N91-24745

Unclas
G3/59 0014927

May-Fun Liou and Ching Yuen Loh
Lewis Research Center
Cleveland, Ohio

Prepared for the
Parallel CFD Conference
cosponsored by CONVEX, CRAY, debis, GAMNI-SMAI, IBM, INTEL, NASA, nCUBE, Siemens-Nixdorf, and Thinking Machines
Stuttgart, Germany, June 10-12, 1991

NASA

Parallel Computing Using a Lagrangian Formulation

May-Fun Liou and Ching Yuen Loh*
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

Abstract

This paper adopts a new Lagrangian formulation of the Euler equation for the calculation of 2-D supersonic steady flow. The Lagrangian formulation represents the inherent parallelism of the flow field better than the common Eulerian formulation and offers a competitive alternative on parallel computers. The implementation of the Lagrangian formulation on the Thinking Machines Corporation CM-2 Computer is described. The program uses a finite volume, first-order Godunov scheme and exhibits high accuracy in dealing with multidimensional discontinuities (slip-line and shock). By using this formulation, we have achieved better than six times speed-up on a 8192-processor CM-2 over a single processor of a CRAY-2.

1. Introduction

For decades, CFD has been showing increasing value in the studies of basic fluid mechanics as well as in aircraft design. At the same time, purely experimental approaches have become very costly. In some cases (e.g. hypersonic flight vehicles), experimental requirements go beyond the capability of the existing wind tunnels. It

*National Research Council - NASA Research Associate at Lewis Research Center.

is generally agreed that orders of magnitude improvement in computer performance will be required to utilize CFD as a design tool for future aerospace vehicles. But the performance of vector supercomputers has been approaching a plateau. Consequently, parallelism in computer architecture has been sought as a viable alternative for delivering the needed high speed performance.

To this end, considerable research and resources have been directed to take advantage of the raw power of the parallel computer. Two different approaches can be taken. The first one is to suitably (optimally) break a large system into many smaller subsystems (e.g. domain decomposition), so that maximum parallelism can be realized. This approach has been the primary focus of past and current research [3]. The second approach is to choose (devise) a formulation (set of equations) that is most suitable for parallel computing. An example in CFD is the Lagrangian formulation. Since the Eulerian formulation does not entail following the streamlines, the convective velocity will cross the cell boundary and result in the transfer of mass, momenta, and total enthalpy between cells. In other words, in addition to the pressure wave interaction at the cell boundary, there is also a cross communication via the convective fluxes in the Eulerian approach. To minimize the cross communication, a better choice is to take advantage of the inherent parallel property of the streamlines by adopting the Lagrangian approach. Thus, the Lagrangian approach reduces numerical operations of fluxes across the boundary of cells and gives rise to much more crisp solution because of eliminating the numerical mixing of fluids. This is the approach adopted in the present research and this paper will detail the formulation, its parallel implementation, and the resulting benefit in accuracy and solution speed.

In supersonic flow , the steady Euler equations are of hyperbolic type and a "time-like" variable can be used to reduce the number of independent variables by one. By further combining the Lagrangian concept, where a "time-like" variable is defined to be along the streamline, a 2-D steady supersonic/hypersonic flow problem is reduced to that of 1-D unsteady flow. There are two important characteristics in the Lagrangian

formulation. First, it results in higher accuracy in dealing with multidimensional discontinuity (slip-line and shock), as demonstrated by Loh and Hui [1,2]. The other characteristic of the Lagrangian formulation is the embedded parallelism. This is due to the fact that we can compute each streamline independently with only the pressure wave interacting between streamlines. To implement the Lagrangian formulation on the Thinking Machines Corporation CM-2 computer, the numerical procedure is programmed in CM FORTRAN. Several Riemann problems in different configurations are computed and compared with exact solutions.

2. The New Lagrangian Formulation of Euler Equations

It is well known that there exists two basic methods of specifying fluid motion : the Eulerian and the Lagrangian. One dimensional unsteady flow and problems of free boundaries composed of the same set of fluid particles are often studied using the Lagrangian formulation. However, most of the theoretical and numerical studies of fluid are based on the Eulerian formulation. In particular, the Eulerian formulation is usually prefered for 3-D steady flow problems because the number of independent variables is reduced from four to three. The conventional Lagrangian formulation still requires four independent variables for 3-D steady flows.

Hui and Van Roessel [4] have introduced a Lagrangian time, τ which plays a dual role (i.e. both a fluid particle label and a measure of time). In this way the number of independent variables for 3-D steady flow is also reduced from four to three, placing the Lagrangian formulation on the same ground as the Eulerian one for steady flow. Thus, for two-dimensional steady flow the independent variables are the stream function ξ and the "Lagrangian time" τ . The conservation form, based on this variable transformation, is given as follows [1]:

$$\frac{\partial \mathbf{E}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \xi} = 0 \tag{1}$$

with

$$\mathbf{E} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{pmatrix} = \begin{pmatrix} K \\ H \\ Ku + pV \\ Kv - pU \\ U \\ V \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} 0 \\ 0 \\ -pv \\ pu \\ -u \\ -v \end{pmatrix}$$

where

$$U = \frac{\partial x}{\partial \xi}, \quad V = \frac{\partial y}{\partial \xi}$$

are the geometrical quantities representing fluid particle deformation during marching forward. The variable

$$K = \rho(uV - vU) \tag{2}$$

is the mass flux and H is the total enthalpy per unit mass,

$$H = \frac{1}{2}(u^2 + v^2) + \frac{\gamma}{\gamma - 1}\frac{p}{\rho} \tag{3}$$

where p, ρ are the pressure and density respectively. The first four equations in (1) represent the physical conservation laws of mass, energy and momentum respectively. The last two equations arise from the compatibility condition between the τ- derivatives and the ξ- derivatives, representing the deformation of a fluid particle.

In the new Lagrangian formulation the coordinate lines are the streamlines and time lines. Consequently, the flow tangency condition on a solid boundary is satisfied exactly on a coordinate line (e.g. $\xi = \xi_0$). We further remark that since slip lines are also streamlines, they must be coordinate lines.

To find a steady flow solution, we need to solve equation (1) subject to the flow tangency condition on the solid boundaries, as prescribed, or free stream condition, as given, and the Rankine-Hugoniot jump conditions across a shock.

3. Godunov Scheme

For supersonic/hypersonic flow, the system of equations (1) is of hyperbolic type.

As indicated by Ortega and Voigt [5], explicit methods tend to be more attractive on parallel computers than on serial computers for solving hyperbolic type equations. However, the explicit methods are constrained by stringent stability requirements. Other considerations when selecting a numerical method for parallel computing include the computer architecture, inter-processor communication, the boundary conditions, the form of the coefficients, and the number of space dimensions, etc. Based on these considerations, we believe that the explicit scheme is more appropriate for our formulation and for the massively parallel computer which we are using. We apply the standard first-order Godunov scheme in a manner similar to that for one-dimensional unsteady flow. The computational domain in the τ-ξ plane is illustrated in Fig. 1 with superscript n refering to the time step number and subscript j as the cell number. The marching step, $\Delta\tau^n = \tau^{n+1} - \tau^n$ is chosen to satisfy the CFL stability condition. The mesh divides the computational domain into control volumes or cells with center at (τ^n, ξ_j) and height of $\Delta\xi_j = \xi_{j+1/2} - \xi_{j-1/2}$ in the ξ-direction. The procedures of solving (1) by the Godunov scheme have been described in detail in [1]. Here, to avoid repetition, we shall only give a brief description.

After integrating eq.(1) over the shaded rectangle in Figure 1 and applying the divergence theorem, the difference equations for the j^{th} cell at time step n are

$$\mathbf{E}_j^{n+1} = \mathbf{E}_j^n - \lambda(\mathbf{F}_{j+1/2}^{n+1/2} - \mathbf{F}_{j-1/2}^{n+1/2}) \tag{4}$$

where $\lambda = \frac{\Delta\tau^n}{h_j}$, and $\Delta\tau^n = \tau^{n+1} - \tau^n$ is the time step size. For $\tau = \tau^n$ the flow variables $\mathbf{Q} = (p, \rho, u, v)$ and geometric quantities (U, V) are assumed to be given and constant within each cell j, denoted as $\mathbf{Q}_j$, U_j and V_j. A sequence of Riemann problems with initial data:

$$\mathbf{Q} = \begin{cases} \mathbf{Q}_{j+1}, & \xi > \xi_j \\ \mathbf{Q}_j, & \xi < \xi_j \end{cases} \qquad j = 1, 2, ..., N-1. \tag{5}$$

are solved to determine the interaction between flows in adjacent cells and subsequently the interface fluxes $\mathbf{F}_{j+1/2}^{n+1/2}$ at the cell interfaces.

The solution to the Riemann problem yields a flow consisting of the Prandtl-Meyer expansion, oblique shock, and slip line. A Newton iterative method is employed for solving the Riemann problem.

If there is a solid boundary, the boundary Riemann solver is employed. Details about the boundary Riemann solver are described in [1] . If there is any slope discontinuity at the solid boundary, the same special treatments as described in [1] are applied to minimize numerical errors. As a matter of fact, these special treatments amount to applying a local exact solution at the sudden turns of the solid body surface.

4. Massively Parallel Computer

Researchers are now finding that many problems in nature, human society, science, and engineering are naturally parallel, and can be effectively solved by using mathematical and computational methods that work in parallel. Therefore if a computer "looks" like the problems and can exhibit thousands degrees of parallelism, we should be able to solve those computation-intensive problems on such a computer and achieve a rapid speedup, relative to a uniprocessor. As a result, we are now experiencing a paradigm shift (i.e. a shift from sequential to massively parallel computing).

The most massively parallel computer built so far is the Thinking Machines Corporation's Connection machine. The Connection machine is of the SIMD (Single Instruction, Multiple Data path) type. The current model CM-2 contains up to 65,536 physical processors ($64K$) in blocks of $8K$(where K stands for 1024) and has a peak speed of approximately 5 GFLOPS (64-bit precision) [6]. Each processor has its own local memory of 8192 32-bit words. In addition, every 32 processor share a floating accelerator chip. Thus, a $64K$ CM-2 has 2048 floating chips. Parallel data sets are spread across the processors, with one single parallel data element stored in each processor's memory. When the number of parallel data elements exceeds the number of

physical processors, the hardware operates in the virtual processor mode by splitting each processor into several equal subprocessors to generate more prosessors, each with correspondingly smaller memory. The ratio of virtual to physical processors is known as the virtual-processor ratio (*VP ratio*). The CM-2 processors all operate in lockstep on data stored in their local memories and execuate a single stream of instructions. The instruction is directed down from a front-end computer which can be a VAX, a Symbolics, or a SUN/4. And the system can connect up to four front-end computers.

The CM-2 system software is designed to utilize the existing programming languages and environments as much as possible. Parallel versions of LISP, C, and FORTRAN are available. PARIS is a low-level instruction set that provides a facility to optimize the execution speed of critical parts of a program. The parallel version of FORTRAN (CM FORTRAN), which incorporates the proposed FORTRAN 8X array extension into the ANSI FORTRAN 77 standard, is employed in this research.

Another feature of the CM-2 system is the interprocessor communication mechanism. There is a dynamical router mechanism. That allows any processor to communicate in parallel to any other processor. There is also a NEWS grid which is a more structured local communication mechanism, and it allows processors to pass data in parallel in a circular multidimensional pattern. The average time to send data through the router is equivalent to 60 integer-add operations while it takes about 6 integer-add operations to communicate with an adjacent processor by NEWS grid [7].

5. Computational Performance and Test Problems

The Lagrangian method is tested on the Thinking Machines Corporation CM-2. The numerical results of several test problems are compared with the available exact solutions to demonstrate the accuracy. The efficiency of this approach is presented by the performance comparisons between the sequential and parallel computing. The

serial version of the same Lagrangian formulation is vectorized and run on a single processor of a CRAY-2. The parallel version has been run on the $8K$ machine with 64-bit floating point hardware.

Results (accuracy)

Based on a detailed comparison of the distribution of all flow variables, the results from the CM-2 parallel version are identical, as they should be, to that of the vectorized serial version on the CRAY-2. The first example is a pure initial value problem, namely a Riemann problem. The top and bottom states are shown in Figure 2. The ratio in pressure, density, and Mach number across the two streams is 4, 2, and $\frac{5}{3}$ respectively. The resulting interaction between the top and bottom streams produces an oblique shock in the low-pressure stream and a Prandtl-Meyer expansion on the high-pressure stream side. Obviously, the numerical results agree well with the exact solution (solid lines) for the slip line and shock wave. Especially, the slip line is resolved with essentially no intermediate points, reflecting the strength of the Lagrangian formulation. On the other hand, the Eulerian formulation will resolve the slip line typically in 5-6 points. Also, the shock resolution is slightly better than the Eulerian calculation for the first-order results, but significantly better for the second-order results (not shown in this paper). The accuracy for the expansion fan is about the same as that of the Eulerian results.

The next example is an initial-boundary value problem. Two shocks are generated at the slope discontinuity on both the upper and lower wall in a converging channel. Subsequently, the shocks collide with each other, resulting in two new shocks and a slip line between them. The upper and the lower wall wedge angles are 10^o and 20^o respectively. The flow variables of the incoming free stream are:

$$p = 2, \quad \rho = 1, \quad u = 13.1483, \quad v = 0$$

Special treatments at the sudden body turns, as decribed in [1], are applied to reduce the local numerical errors. Figure 3 (a) and (b) illustrate the pressure and density along

a typical time line after the shock collision (the line $B - B'$), with the exact solutions represented by solid line. The exact solution to this problem is obtained from the oblique shock theory which predicts the strength and location of these induced shocks. The numerical results agree well with the exact solutions, the shocks are seen to be resolved in 4-5 points and the slip line in 2 points.

cpu time (efficiency)

To utilize CM-2 to its maximum efficiency, there are two general rules [7] to follow. First, avoid too many processors sitting idle. Since CM-2 is a SIMD type machine, it takes the same amount of time to perform operations either for a single grid or for the entire grids (i.e. try to use all processors simultaneously). Second, the communication through the general purpose communication network (router) is very slow and should be minimized. Following these two rules, we let all the fluid cells, with each processor representing one cell, on the same time line (τ) march forward simultaneously. In addition, the communication in our program is limited to the nearest neighbor cells due to the numerical scheme we adopt (i.e. omitting the useage of costly router but still achieving high accuracy). However, treating the boundary points separately results in the decrease of the CM-2 efficiency and is inevitable.

A built-in facility, called the Paris timer, in CM-2 is used to measure both the total elapsed time and the time during which the CM was active. The UNIX front-end has some degree of multiprocessing activity which results in interference between processes even when only one user is logged in. Such interference can lead to timing distortion introduced by other processes. Another factor affecting the accuracy of timing is that the UNIX real time clock has lower resolution than the CM cycle time. To be more accurate in timing, we run each case three times and report the average of those three values.

The following timings are based on the overall time for a run, including both execution time on the front-end and the CM-2. Execution on the front end consists of the following operations: reading the input, writing the output, doing scalar calculations,

setting the logical masks, and transferring arrays to the CM-2. The overhead due to the front end operation can weaken the CM-2 performance and is a function of n, the size of problem. The comparison of overall performance of parallel and serial versions is obtained for three situations: (1) the "embarrassingly parallel" situation, where the size of the problem, n, is very much less than the number of processors, p; (2) the problem's size is equal to the number of processors, $n = p$; and (3) the large problem whose size $n >> p$. Because of the structure of the CM-2, the problem size has to be dimensionalized to match the number of available virtual processors. For example, the grid of 60 K points will run as long as a grid of 64 K points. Furthermore, the higher the VP *ratio* the better. Since the grid is automatically generated as a part of the solution, we like to point out that the execution time we are discussing above includes the time spent on grid generation too.

Figure 4 shows the overall performance as a function of the number of cells in the flow field of the initial value Riemann problem. *Cpu* times are shown for the CRAY-2, CM-2 only, and CM-2 plus front end. The horizonal axis represents the total number of cells along the ξ direction while the vertical one denotes the *cpu* seconds. The curve of the CM-2 performance (dashed line) is in steps distribution and is function of the number of available virtual processors only. The solid line represents the performance of the sequential computation on a single processor of CRAY-2. The overall performance denoted by the symbol Δ shows the deviation from the ideal CM-2 distribution because of the overhead from the front end. Similarily, Fig. 5 presents the performance comparison between the CRAY-2 and CM-2 for the converging channel flow. A substantial decrease in the performance efficiency of the CM-2 is noticed. Since this test case needs more numerical operations in handling the boundary condition, Fig. 5 shows that the execution time spent on the CM-2 processors dominates and weakens the overall performance. The effect played by the boundary conditions on the CM-2 is explained in the following.

As we mentioned previously, the computation of boundary points is a source of

computational inefficiencies on the CM-2. Many processors, assigned to interior points, are sitting idle while some processors execute operands of boundary conditions. At the boundary of interest, preset masks are employed to design specific boundary condition. The mask is a logical array of bits, each bit associated with a single processor, whose context (false or true) determines whether or not the result of the operation of the processor is actually used.

There are two types of boundary : free boundary and wall boundary. Case 1 in our testing is a free boundary problem to which we apply the zero gradient condition at the far field boundaries. In CM-FORTRAN an intrinsic function CSHIFT which is the circular shift of the data in the specified array along a specified axis of the array for a specified displacement handles this kind of boundary (i.e. extrapolation from interior point) easily. As wall boundary is concerned, such as the case 2, the velocity normal to the wall is zero and the body surface is one of the streamlines. Also special treatments are employed to impose exact solution locally around the sudden turn of boundary[1]. Case 2 is an example of the more complicated boundary condition we will encounter in the applications of Lagrangian formulation. The relative performance of two different types of boundary condition is shown in Fig. 6. It indicates that both case 1 and 2 have the same degree of computatioal intensity on the CRAY-2. Due to the special treatments along the boundaries, we observe that there is approximately 2.4 times cost of *cpu* seconds on CM-2 for the converging channel (initial-boundary value problem) over the Riemann problem (initial value problem). Though Fig. 5 indicates that the CM-2 still outperforms the CRAY-2 for the more complicated boundary condition as the case of converging channel, minimizing the inefficiency caused by boundaries will be undertaken in the future.

Tables 1 and 2 display the relative performance in terms of the overall *cpu* time per cell with respect to *VP ratio*. Item *cpu* ratio/cell represents the ratio of the *cpu* time spent on the $8K$ CM-2 to the time on a single processor of the CRAY-2 for each cell. And the CM utilization indicates how efficiently the CM processors are utilized.

It is calculated as the percentage of the CM-2 executing time over the elapsed real time. Using the formulation and procedures described in this paper, we achieve the speed-up approximately 2 to 7 times faster for the problems/conditions studied. Thus, the expectation of the inherent parallelism existing in this Lagrangian formulation has been confirmed.

6. Conclusions

The inherent parallelism that exists explicitly in the Lagrangian formulation has been exploited on a massively parallel computer. The parallel processing of this Lagrangian formulation has shown its good efficiency and offers an alternative to the conventional Eulerian formulation; the best performance on a 8192-processor CM-2 machine was shown to be approximately seven times over that of a single processor CRAY-2. In this formulation the grid is automatically generated to follow the streamlines, as a part of the solution. In addition, it achieves higher accuracy than the Eulerian formulation.

It is suggested that using the combination of the Lagrangian formulation and the massively parallel computer should result in efficient calculations of those computation-intensive problems with complex configurations. Thus, a 3-D viscous code based on this research will be developed to deal with those computation-intensive problems, including the real gas calculation for hypersonic problems. Furthermore, the implementation of this Lagrangian formulation on a MIMD machine (i.e. iPSC/860) will be investigated.

References

1. C.Y. Loh and W.H. Hui, "A new Lagrangian method for steady supersonic flow

computation Part I: Godunov scheme," *J. Comput. Phys.* **89**(1990).

2. C.Y. Loh and W.H. Hui, "A new Lagrangian Random Choice method for Solving the Steady Euler Equations," submitted to *SIAM J. of Scientific and Statistical Computing.*
3. U. Schnendel, *Introduction to Numerical Methods for Parallel Computers,* B. W. Conolly, Trans., Halsted Press, New York (1984).
4. H. J. Van Roessel and W. H. Hui, "A New Lagrangian Formulation for Steady Three Dimensional Hypersonic Flow," *J. Appl. Math. Phys.* **40**(1989).
5. J. M. Ortega and R. G. Voigt, *Solution of Partial Differential Equations on vector and parallel Computers* , SIAM, Philadelphia (1985).
6. "Connection Machine Model CM-2," Thinking Machines Technical Report HA87-4, Thinking Machine Corp., Cambridge, Mass., 1987.
7. B. E. Wake and T. A. Egolf, "Implementation of a Rotary-Wing Navier-Stokes Solver on a massively Parallel Computer," *AIAA J.* **29** (1991).

	CM-2				Cray-2
cpu ratio/cell	0.145	0.154	0.25	0.265	1
CM utilization	71%	62%	58%	31%	N/A
VP* ratio	16	8	4	2	

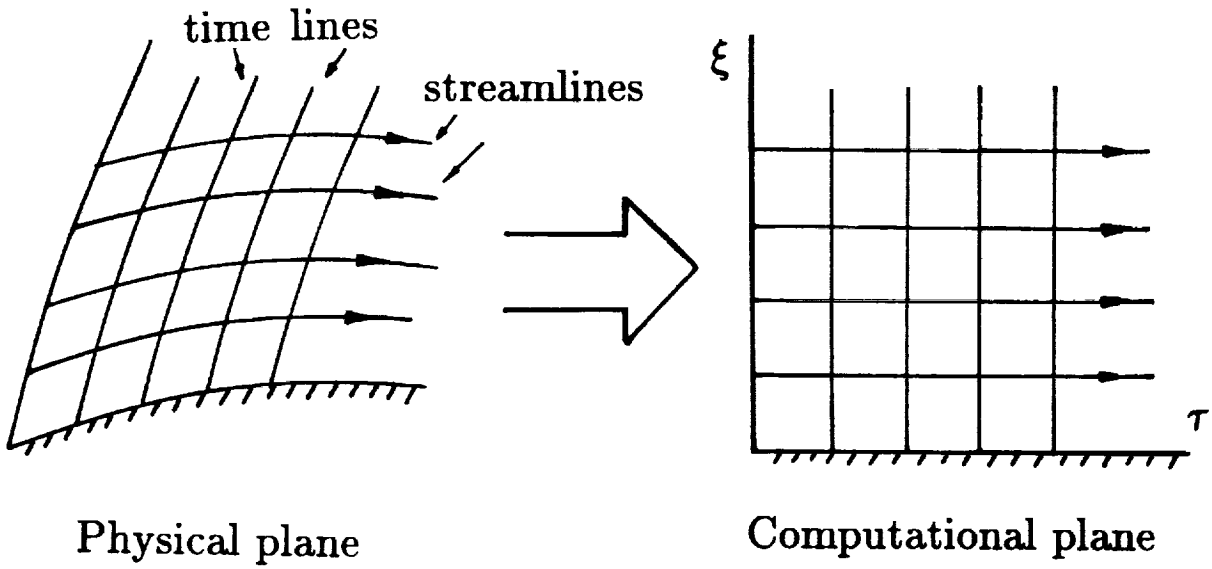
* VP: virtual processor

Table 1. Performance comparison for Riemann problem

	CM-2				Cray-2
cpu ratio/cell	0.24	0.27	0.30	0.51	1
CM utilization	92%	86%	79%	54%	N/A
VP* ratio	16	8	4	2	

* VP: virtual processor

Table 2. Performance comparison for converging channel



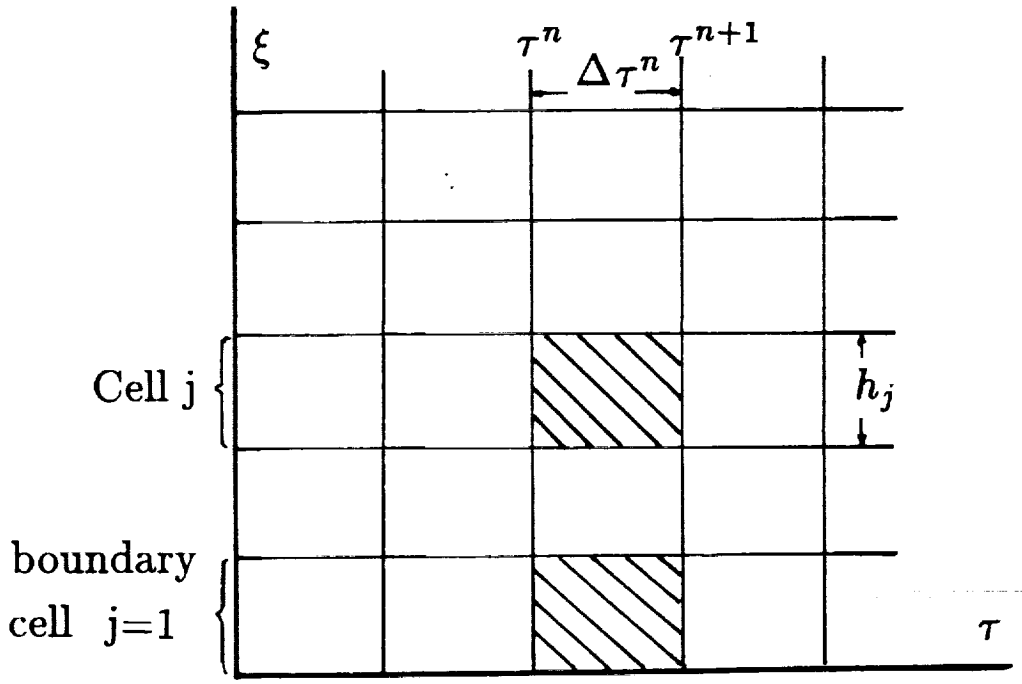
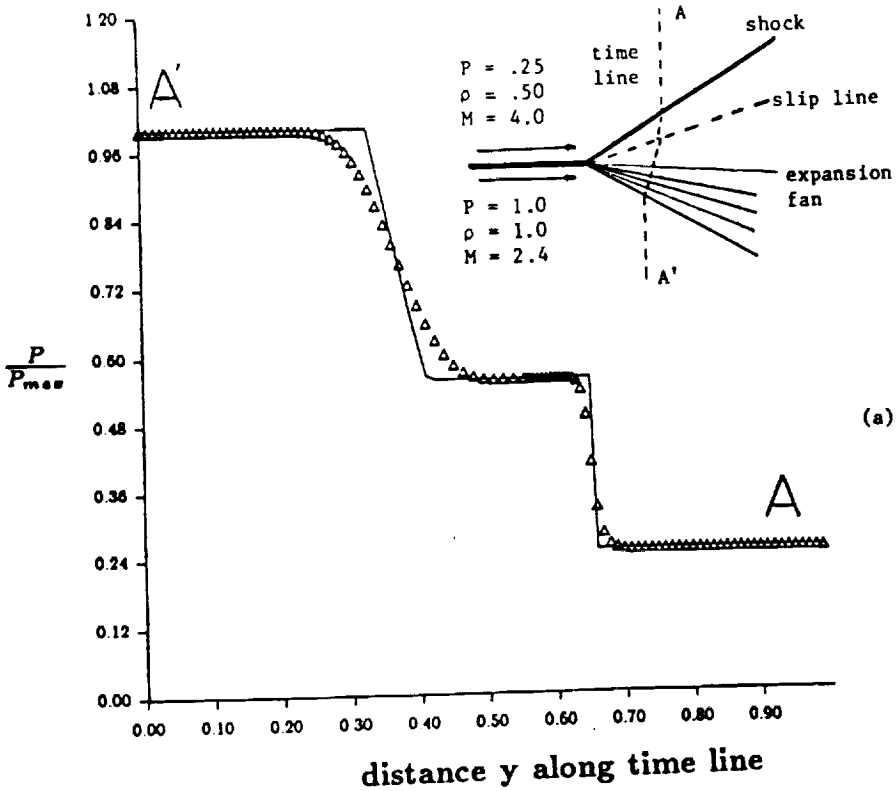


Fig. 1. Computational Domain and Mesh



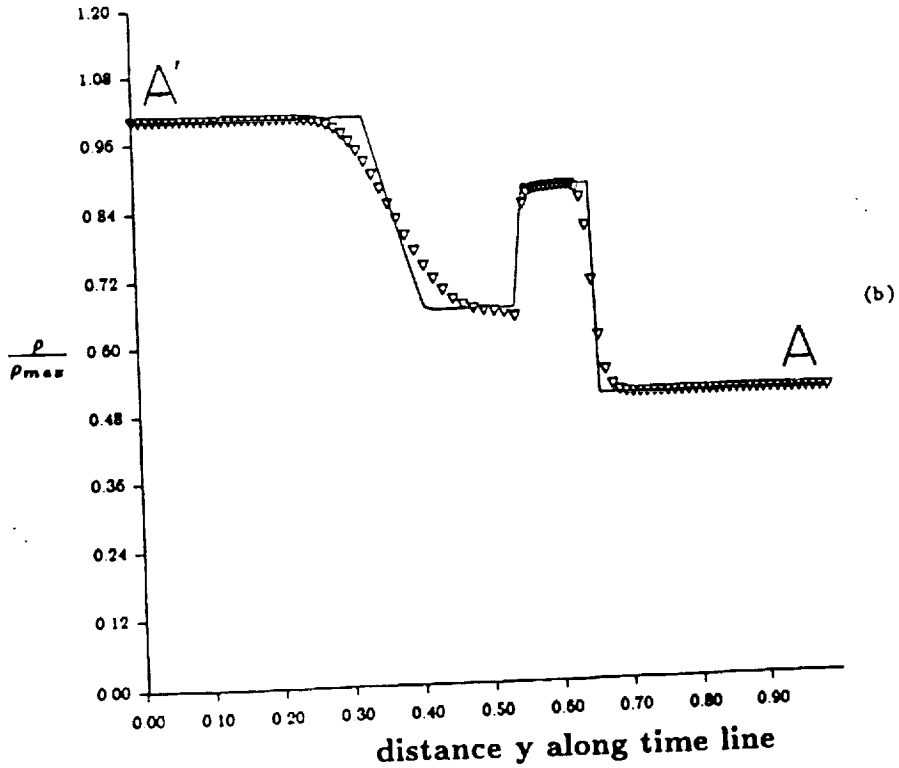
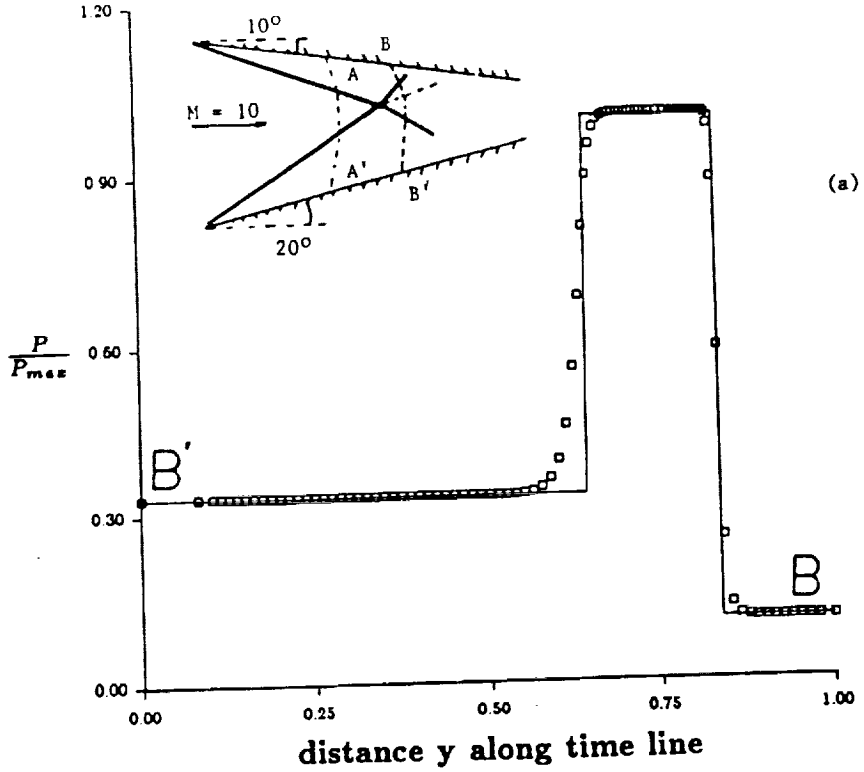


Fig. 2. Flow variables vs. normalized Eulerian coordinate y along time line A-A' (τ=.125). Solid line denotes the exact solution



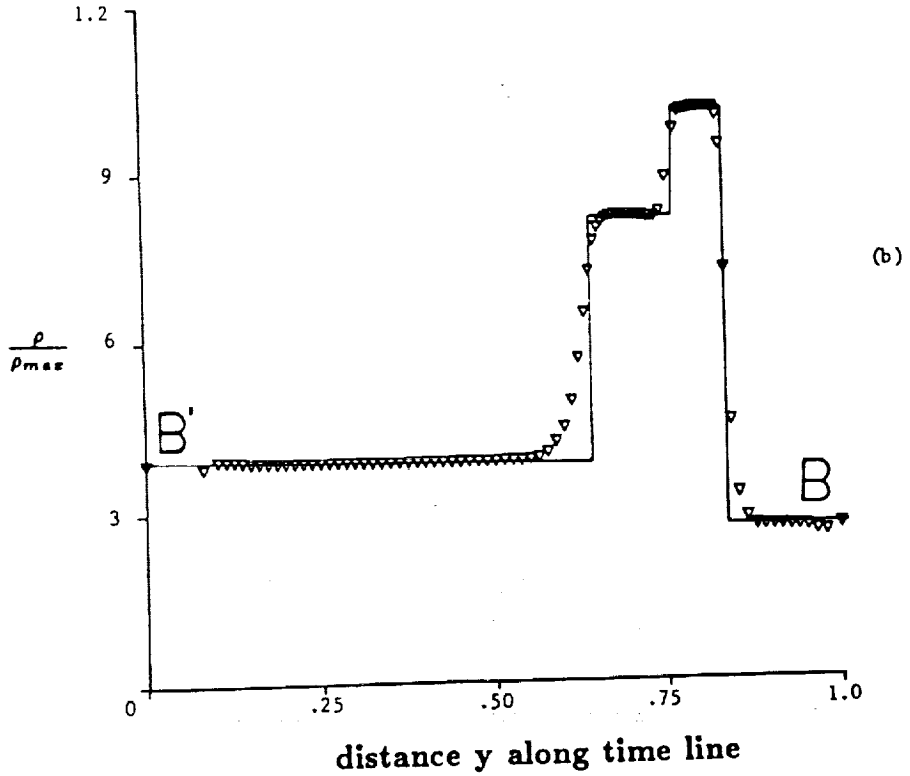


Fig. 3. Flow variables vs. normalized Eulerian coordinate y along time line B-B'. Solid line denotes the exact solution

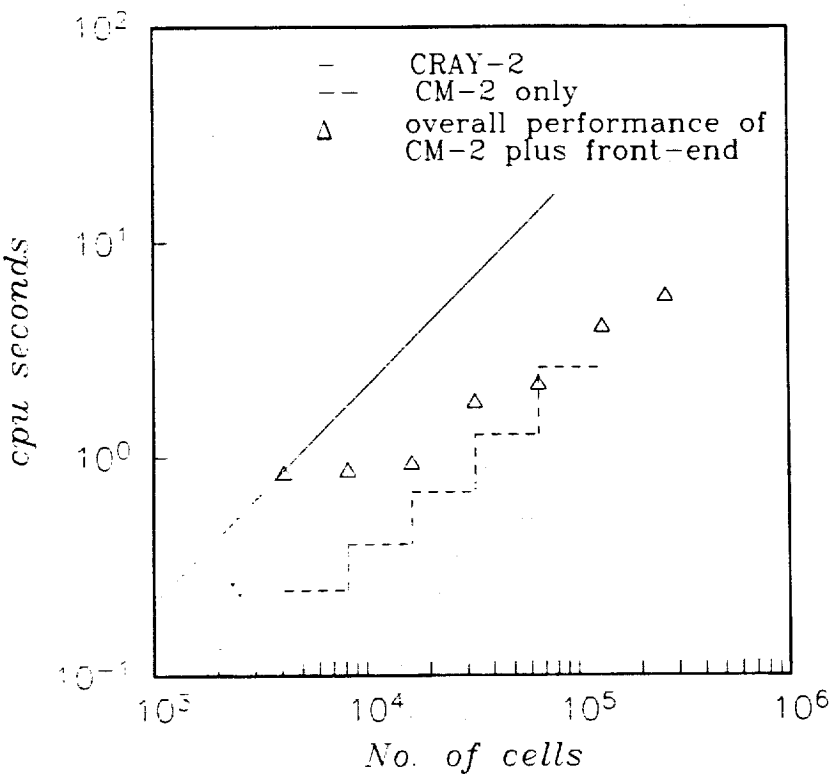
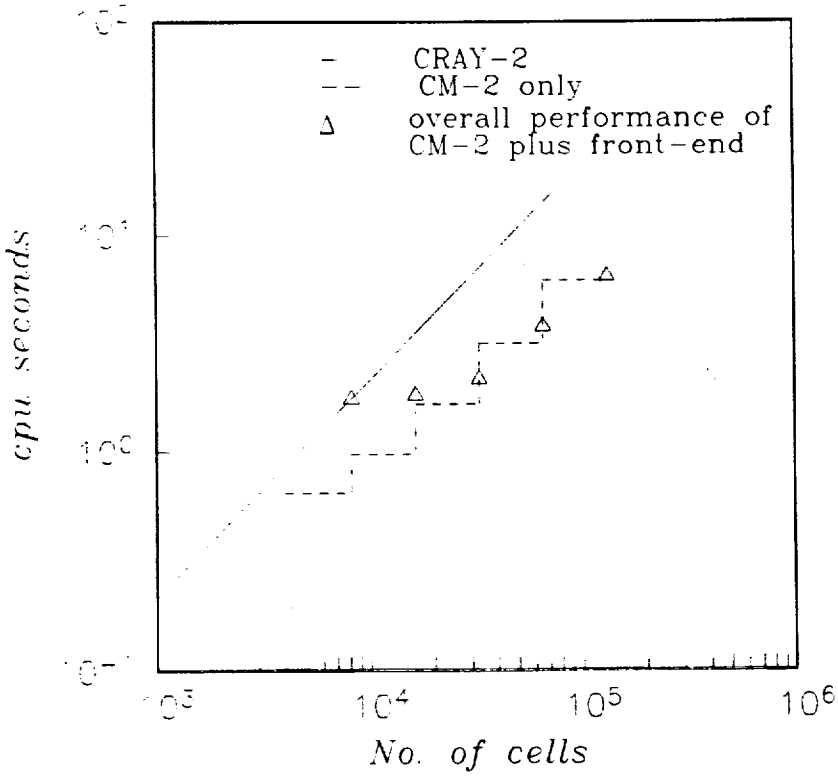


Fig. 4 Performance comparison for Riemann problem



ORIGINAL PAGE IS
OF POOR QUALITY

Fig. 5 Performance comparison for converging channel

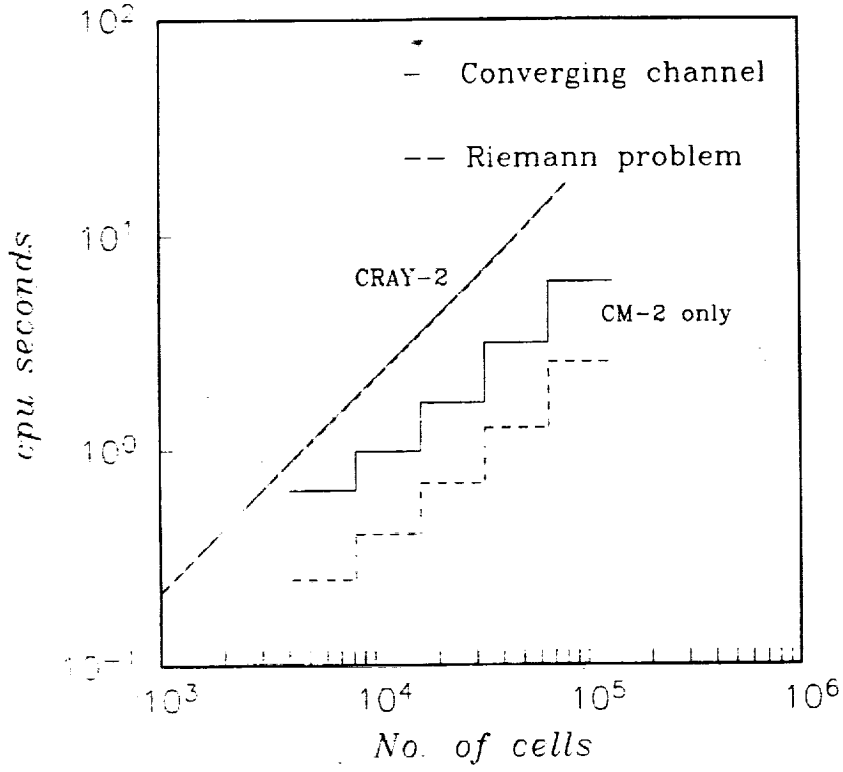


Fig. 6 Performance comparion for different problems/conditions

ORIGINAL PAGE IS
OF POOR QUALITY

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-104446		

4. Title and Subtitle	5. Report Date
Parallel Computing Using a Lagrangian Formulation	
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
May-Fun Liou and Ching Yuen Loh	E-6286
	10. Work Unit No. 505-62-52

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

Prepared for the Parallel CFD Conference cosponsored by CONVEX, CRAY, debis, GAMNI-SMAI, IBM, INTEL, NASA, nCUBE, Siemens-Nixdorf, and Thinking Machines, Stuttgart, Germany, June 10-12, 1991. May-Fun Liou, NASA Lewis Research Center. Ching Yuen Loh, National Research Council-NASA Research Associate at Lewis Research Center. Responsible person, May-Fun Liou, (216) 433-3600.

16. Abstract

This paper adopts a new Lagrangian formulation of the Euler equation for the calculation of 2-D supersonic steady flow. The Lagrangian formulation represents the inherent parallelism of the flow field better than the common Eulerian formulation and offers a competitive alternative on parallel computers. The implementation of the Lagrangian formulation on the Thinking Machines Corporation CM-2 Computer is described. The program uses a finite volume, first-order Godunov scheme and exhibits high accuracy in dealing with multidimensional discontinuities (slip-line and shock). By using this formulation, we have achieved better than six times speed-up on a 8192-processor CM-2 over a single processor of a CRAY-2.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Computational Fluid dynamics Euler-Lagrange Parallel processing (computers) SIMD (computers)	Unclassified - Unlimited Subject Category 59

19. Security Classif. (of the report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	20	A03

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161

PAGE 19 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED