JOURNAL OF COMPUTATIONAL PHYSICS 12, 534–552 (1973)

A Parallel Algorithm for High Subsonic Compressible Flow over a Circular Cylinder

S. RAJAN

Center for Advanced Computation, University of Illinois, Urbana, Illinois 61801

Received October 18, 1972

The development of superfast computers, which operate on data other than on the purely sequential principle, calls for new computational algorithms which make full use of the machine capabilities. Using the inviscid compressible supercritical flow over a circular cylinder as an example, a computational algorithm is developed for a parallel processing computer. All boundary conditions are satisfied explicitly and boundary points are computed in parallel with interior points, thereby greatly increasing the "machine utilization." Treatment of mesh size changes with the parallel algorithm has been accomplished in a novel way. The influence of various finite difference formulations of the wall boundary condition on the computed results is studied and the results are compared with those obtained by other authors. The importance of correlating the formulation of the physical problem with the numerical technique and the computational procedure to secure a high machine utilization in parallel computation is demonstrated.

1. INTRODUCTION

Various numerical techniques have been employed to compute the inviscid transonic flow over airfoil sections [1–4] and cylindrical bodies [5–7]. Problems of this category are numerically complex and computationally time consuming [1]. The finite difference calculations have thus far all been performed on computers operating on the sequential principle and extensions of the problems have been severely limited by the computing capacities of available machines. The parallel computer concept offers distinct advantages in terms of speed, computing capacity and storage for finite difference computations. With a larger class of problems in mind, of which the transonic airfoil problem is one example, a computational technique for a parallel machine is developed and illustrated in the calculation of the inviscid supercritical flow over a circular cylinder. The technique as such is adaptable to more complicated geometries with additional physical features.

In the numerical solution of a given problem three aspects are to be considered.

1. The formulation of the physical problem with the appropriate governing equations, and the relevant initial and boundary conditions.

2. The selection of an adequate numerical technique to ensure a consistent computational procedure. Considerations of stability and accuracy enter the picture here.

3. The actual implementation of the computational procedure on the machine.

With new machines like the Illiac IV which operates on the parallel processor concept, this last consideration takes on added significance. A problem governed by a set of partial differential equations and solved on a parallel machine must take notice of the way a parallel machine computes. The parallel computational algorithm is different from that of the serial machine, and the special features of the machine must be correlated with the formulation and programming of the physical problem, from the very outset of the computation. Otherwise many of the advantages offered by the new computers, such as increased computing capacity, increased storage and speed of computation are lost, by computing the problem in the same manner as with a serial machine. This is a crucial point if the parallel machine is to be used efficiently. It now becomes essential to correlate the above three criteria at every stage of the computation and give each its proper weightage, to ensure the maximum possible machine utilization. This in turn paves the way towards solution of more complex problems which the new generation of computers promises.

The manner in which this is done for the case of the inviscid, compressible, supercritical flow around a circular cylinder, is developed in this paper.

2. Governing Equations

In computing the inviscid, compressible steady state flow around a circular cylinder, it is convenient to formulate the problem in polar coordinates. The conservation laws governing the flow are the following:

continuity:

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial r} + \frac{v}{r}\frac{\partial \rho}{\partial \theta} + \rho\frac{\partial u}{\partial r} + \frac{\rho}{r}\frac{\partial v}{\partial \theta} + \frac{\rho u}{r} = 0; \qquad (1)$$

momentum:

r direction

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + \frac{v}{r}\frac{\partial u}{\partial \theta} - \frac{v^2}{r} + \frac{1}{\rho}\frac{\partial p}{\partial r} = 0; \qquad (2)$$

θ direction

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial r} + \frac{v}{r}\frac{\partial v}{\partial \theta} + \frac{uv}{r} + \frac{1}{\rho r}\frac{\partial p}{\partial \theta} = 0; \qquad (3)$$

energy:

$$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial r} + \frac{v}{r}\frac{\partial s}{\partial \theta} = 0; \tag{4}$$

state:

$$p = \rho \bar{R} T. \tag{5}$$

After some manipulation these equations can be written in vector form as a single equation of the form

$$F_t + AF_r + BF_\theta + C = 0 \tag{6}$$

where,

$$F = \begin{bmatrix} P \\ u \\ v \\ s \end{bmatrix}, \quad A = \begin{bmatrix} u & \gamma & 0 & 0 \\ p/\rho & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{bmatrix}, \quad B = \frac{1}{r}\begin{bmatrix} v & 0 & \gamma & 0 \\ 0 & v & 0 & 0 \\ p/\rho & 0 & v & 0 \\ 0 & 0 & 0 & v \end{bmatrix}, \quad C = \begin{bmatrix} \gamma u \\ -v^2 \\ uv \\ 0 \end{bmatrix}.$$

Here $P = \ln(p/p_\infty)$. Subscripts denote differentiation with respect to the subscripted variable.

In the actual computations, Eqs. (1)–(5) are solved in nondimensional form using

$$(p_\infty/\rho_\infty)^{1/2}, \qquad r_0/(p_\infty/\rho_\infty)^{1/2}, \qquad \text{and} \qquad (p_\infty/\rho_\infty)$$

as reference values for velocity, time and temperature, respectively. Also the normalized entropy is given by

$$S = \ln(p/p_\infty) - \gamma \ln(\rho/\rho_\infty). \tag{7}$$

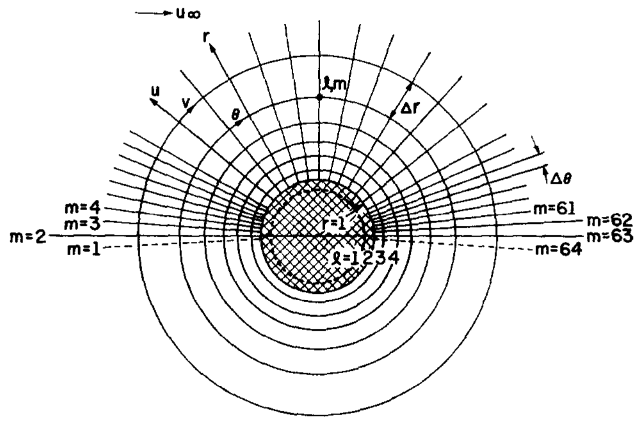
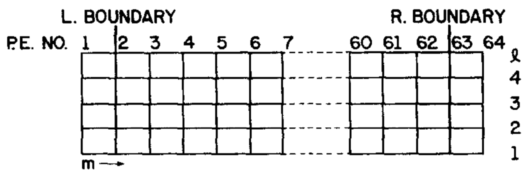


FIG. 1. Computational Mesh system.



FOR ROWS 1 AND 3	FOR P.E. 1,3 AND P.E. 62,64
$p(1,m) = p(3,m)$	$p(\ell,1) = p(\ell,3)$
$u(1,m) = -u(3,m)$	$u(\ell,1) = u(\ell,3)$
$v(1,m) = v(3,m)$	$v(\ell,1) = -v(\ell,3)$
$s(1,m) = s(3,m)$	$s(\ell,1) = s(\ell,3)$
$\rho(1,m) = \rho(3,m)$	$\rho(\ell,1) = \rho(\ell,3)$ ETC.

FIG. 2. Computational arrangement of mesh points.

The conservation equations are quasilinear with respect to F and hyperbolic with respect to t, and are the Euler equations in polar coordinates. The solution of these equations for a given free stream Mach number and specified body geometry, involves the determination of the dependent variables P, u, v, and ρ at each point in the computational space. The steady state solution of the compressible flow field is obtained as the long time limit of the temporal formulation.

3. The Finite Difference Formulation

In the computational space, the above governing differential equations are replaced by equivalent finite difference equations. Discretized values of the variables at individual mesh points are used to formulate the finite differences. A variation of the explicit two-step Lax–Wendroff scheme is employed in this investigation to illustrate the parallel algorithm. A linear stability analysis of the scheme shows that it is stable if the Courant–Friedrichs–Lewy criterion is satisfied. A value of the time step Δt given by

$$\Delta t = \Delta r/1.5(|u| + a) \tag{8}$$

is used, where the factor 1.5 is introduced to be on the safe side.

Intermediate values for the variables P, u, v, and ρ are first calculated at time level $(n + 1/2)\Delta t$ from known values at time level $n\,\Delta t$. For the differential equation

$$F_t + AF_r + BF_\theta + C = 0 \tag{6}$$

the intermediate values are given by the equation

$$\begin{aligned} F_{l,m}^{n+1/2} = (1/4)[F_{l+1,m}^{n} + F_{l-1,m}^{n} + F_{l,m+1}^{n} + F_{l,m-1}^{n}] \\ - (\Delta t/2\Delta r)\, A_{l,m}^{n}[F_{l+1/2,m}^{n} - F_{l-1/2,m}^{n}] \\ - (\Delta t/2\Delta \theta)\, B_{l,m}^{n}[F_{l,m+1/2}^{n} - F_{l,m-1/2}^{n}] - (\Delta t/2)\, C_{l,m}^{n} \end{aligned} \tag{9}$$

where $F_{l,m}^{n}$ is the value of F at the point $(l\,\Delta r, m\,\Delta\theta)$ and time $n\,\Delta t$. Final values of F at time level $(n+1)\,\Delta t$ at the same point are computed from the intermediate values from the equation

$$\begin{aligned} F_{l,m}^{n+1} = (1-\phi)\, F_{l,m}^{n} + (\phi/4)[F_{l,m+1}^{n} + F_{l,m-1}^{n} + F_{l+1,m}^{n} + F_{l-1,m}^{n}] \\ - (\Delta t/\Delta r)\, A_{l,m}^{n+1/2}[F_{l+1/2,m}^{n+1/2} - F_{l-1/2,m}^{n+1/2}] \\ - (\Delta t/\Delta \theta)\, B_{l,m}^{n+1/2}[F_{l,m+1/2}^{n+1/2} - F_{l,m-1/2}^{n+1/2}] - \Delta t\, C_{l,m}^{n+1/2}. \end{aligned} \tag{10}$$

This formulation involves a nine-point grid. Here ϕ is a numerical viscosity parameter which provides additional stability other than that inherent in the truncation viscosity of the finite difference scheme [9].

In this formulation of the finite difference equations, no attempt was made to cast the governing equations into conservation form, as the method illustrated herein for a parallel machine is not affected by the form of the equations. Furthermore, in certain types of applications the final steady state result does not appear to be dependent on the form of the governing equations as demonstrated by Moretti [10]. In addition, the shock strengths in the range of supercritical Mach numbers considered here are such that any inaccuracies introduced by the non-divergence form of the equations are not expected to be greater than the error introduced by the finite difference scheme itself.

4. Parallel Computer Characteristics

We will now describe the essential features of the parallel computer insofar as it helps us to understand the computational algorithm for a parallel machine.

Most computers in use today are sequential machines in which different blocks of data are operated upon one after another, by an identical program. Each operation on a data block corresponds to one iteration in a loop. The mode of

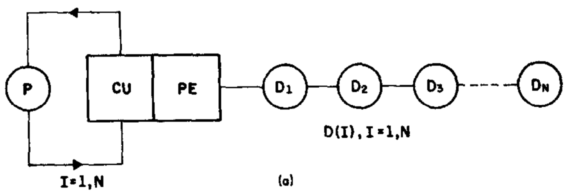


FIG. 3(a). Serial computer concept.

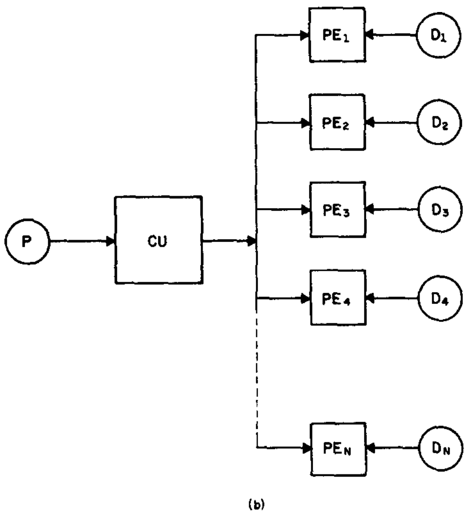


FIG. 3(b). Parallel computer concept.

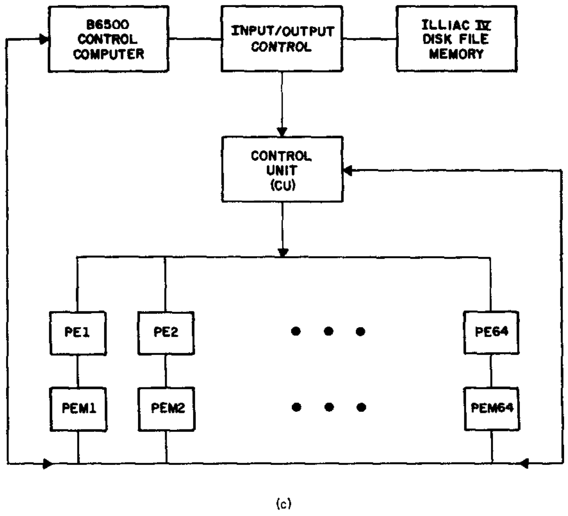


FIG. 3(c). Illiac IV parallel machine

operation is illustrated schematically in Fig. 3(a). There is only one processing unit (PU) performing arithmetic operations, and hence the rate of data throughput in the sequential machine is limited. The parallel concept differs from this in that a number of arithmetic units operate simultaneously on different data blocks, under the direction of a control unit (Fig. 3(b)). For example, in the Illiac IV parallel computer (Fig. 3(c)), there are 64 processing units under the direction of each control unit. Each processing unit contains the relevant hardware for performing the usual arithmetic operations in addition to an associative memory unit capable of storing 2048 sixty-four bit words. The program may be stored in the main disc memory and then transferred in blocks to the array memory.

The arithmetic units are connected in such a way that direct, parallel word transfer paths exist between each processing unit and others having assigned labels differing by plus and minus eight and plus and minus one. For example PU 11 can transfer words directly to PU's 19 and 3 and PU's 12 and 10. In finite difference calculations this feature is very useful. Thus in taking either a forward or a backward difference we can proceed as follows:

forward difference:

Transfer U_i in PU no. i to memory of PU no. $i + 1$
Subtract. $(U_{i+1} - U_i)$ is in PU no. $i + 1$.

backward difference:

Transfer U_i from PU no. i to memory of PU no. $i - 1$.
Subtract. $(U_i - U_{i-1})$ is in PU no. $i - 1$.

Since all the PU's are performing the same transferring (or shifting) operations simultaneously, the appropriate differences at each mesh point reside in the required processing element (PE) memory. This can be ensured by storing the quantities U_i in the appropriate PE memory in the beginning before executing the arithmetic operations.

Central differences may also be readily obtained as the differences of forward and backward differences. Since these differences can be obtained simultaneously in parallel instead of sequentially, the parallel algorithm greatly increases the computing capacity in proportion to the number of arithmetic units operating in parallel. Therefore in finite difference mesh point calculations, the dependent variables at a number of mesh points can be calculated simultaneously, up to a maximum of 64 points.

The arithmetic unit or processing element (PE) is driven by the control unit (CU) to execute the instruction string contained in the CU. Each control unit commands a set of 64 processing units (PU's). It is responsible for the initial processing of instructions, up to and including the generation of instruction microsequences for a step by step control of the processing elements. This control unit manipulates two types of instruction streams: those which it decodes for specifying commands to the PE's and those which command the common registers.

Thus all the processing elements execute the same instruction stream. Since each PE calculates the dependent variables at a given mesh point, the instruction stream for calculating the dependent variables at all the 64 mesh points must be the same. This criterion is especially hard to satisfy at a boundary point in the calculation since in most cases the instruction stream for the boundary point differs from the instruction stream for the interior point. The way in which this difficulty has been overcome in this work, will be described later on.

Other essential features of the Illiac IV parallel computer are the input–output subsystem, the disc file memory and the control computer. These elements are shown schematically in Fig. 3(c).

To fully utilize the parallel concept the maximum possible number of arithmetic units should be kept busy during the calculation. This requires a proper computational mesh layout and problem formulation. In most cases, it is the boundary points that offer the maximum difficulty in parallel computation, and the formulation of the boundary conditions require the greatest attention.

5. The Boundary Conditions

We now examine the specification of appropriate boundary conditions to satisfy the following criteria:

(a) The boundary conditions must be physically realistic so as to provide the correct solution to the initial-boundary value problem.

(b) The computation of mesh points on the boundary surface must be accomplished in such a way as not to reduce the machine utilization.

In this paper the term machine utilization as applied to a parallel processing machine is defined thus:

$$\textit{Machine Utilization} = \frac{\textit{Actual number of PE's performing useful calculations}}{\textit{Total number of PE's capable of useful calculation}}.$$

Since in a serial machine the arithmetic statements are processed one after the other, there is no loss of computing capability when boundary points are calculated from a separate finite difference formulation than that used at interior points. The same strategy applied to a parallel machine results in a serious loss in machine utilization. Let

N_1 = number of boundary points encountered in a row of mesh points.

N_2 = total number of processing units of a parallel machine.

During boundary point calculations ($N_2 - N_1$) of the arithmetic units would be shut off, since these points require a different set of instructions than those used at interior points. Thus machine utilization during boundary point calculations would be

$$(N_2 - (N_2 - N_1))/N_2 = N_1/N_2$$

For a two point boundary calculation on a machine like the Illiac IV which has 64 arithmetic units, the machine utilization is only about 4 %. This is very poor usage of the machine, considering that this loss of computing capacity is suffered at each time step iteration in the calculation. On the other hand if boundary points can be computed with the same instruction stream as interior points, while still satisfying conditions (a) and (b) above, almost all of the N_2 processing units can be simultaneously employed, with consequent increase in machine utilization. We now examine each of the boundary conditions to see how this is done.

A. $\theta = 0$ *and* $\theta = \pi$ *Boundaries*

A study of the finite difference representation of the governing equations obtained with the Richtmyer two-step scheme shows that the boundary points on the $\theta = 0$ and $\theta = \pi$ lines may be computed with the same instruction stream as other interior points, if reflection conditions are specified at these boundaries. This criterion is also physically realistic. The corresponding mesh arrangement is shown in Fig. 1 and 2. The dependent variables at the $\theta = 0$ and $\theta = \pi$ lines on any circumferential row of points are calculated in PE 2 and PE 63, respectively. They are labeled as $m = 2$ and $m = 63$ in the above mentioned figures. The PE's 1 and 64 contain the reflected values of the dependent variables in PE 3 and PE 62, respectively, and do not perform useful mesh point calculations. As a result there is a loss of 2/64 or roughly 4 % of computing capacity. However, as boundary points on the $\theta = 0$ and $\theta = \pi$ lines are now computed simultaneously in parallel with exactly the same instruction stream as interior mesh points 3 to 62, the machine utilization has been brought close to 100 per cent.

B. *Boundary Condition on the Cylinder Surface*

On the cylinder body surface, the radial component of the velocity is zero and is so maintained for all time. At the body surface reflection conditions are again specified. This requires the storing of the dependent variables on a fictitious row of points as illustrated in Fig. 1, and is an increased demand for storage in each PE memory. If a one-sided difference scheme is employed at the body surface, this fictitious row of points is not required. However, calculations show that reflection conditions at the body surface provide smoother results and faster approach to the steady state solution. In this case although there is a change in the instruction stream with the one-sided scheme at the cylinder body surface, this is not crucial because the changed instruction set is fed to all the 64 processing units.

C. *Far Field Boundary Condition*

To carry the computation far afield into the free stream would require a great many grid points. Instead an inverse transformation is used which maps the region external to a given radius in the physical space into another circular space such that infinity in the physical space now transforms to the origin in the new transformed space. The conditions at infinity, namely $P = P_\infty$, $u = u_\infty$, $v = v_\infty$ and $\rho = \rho_\infty$ are applied on a circle of small radius in the transformed space. The transformed governing equations are now applied in finite difference from in the transformed space. Sufficient overlap is maintained between computations in the physical and transformed spaces. The parallel computational algorithm in the transformed space is identical to that in the physical space.

Truncation errors in the far field will be larger than in the near field. However, due to the small variation of the physical variables from the free stream values, this effect is not considered important. Further details may be found in Ref. [7].

6. Computation on a Parallel Machine

The method of solution is to superpose on the computational space a finite difference grid system, as shown in Fig. 1. Initial values of P, u, v, and ρ, equal to the free stream values at infinity, are assigned to each grid point. Values at boundary points are specified to comply with the boundary conditions discussed in the section on boundary conditions.

To compute in parallel on a machine like the Illiac IV, the circumferential row of mesh points at any radius r (Fig. 1) are allotted to a row of 64 processing units, one mesh point per processing unit (Fig. 2). These are labeled from $m = 1$ to $m = 64$ in Figs. 1 and 2. The mesh points are arranged such that at the $\theta = 0$ and $\theta = \pi$ boundaries, the values of m are 2 and 63, respectively. Each circumferential row of mesh points lies at a particular radius r, and is labeled with the letter l. The row of points $l = 1$ lies below the body surface and is required to satisfy the specified reflection boundary conditions. Each processing unit m is engaged in the operations of computing the value of the variables P, u, v, and ρ at the point $m\,\Delta\theta$ on the circumference. Since we have a row of processing units, a circumferential row of mesh points are computed simultaneously in parallel.

The computational space is traversed from the cylinder body surface to the far field via the transformed space in the far field. Beginning at the body surface, the two-step Lax–Wendroff analog of the conservation equations is employed to compute $(n + 1)\,\Delta t$ values from known $n\,\Delta t$ values. In this process two sets of intermediate $(n + 1/2)\,\Delta t$ values of P, u, v, and ρ centered at $(m + 1/2)\,\Delta\theta$ and $(m - 1/2)\,\Delta\theta$ and at $(l + 1/2)\,\Delta r$ and $(l - 1/2)\,\Delta r$ are obtained. The derivatives $\partial p/\partial\theta$, $\partial u/\partial\theta$, $\partial v/\partial\theta$ are obtained by simultaneous transfers across the PE's in the proper directions, while the r derivatives are computed individually in parallel in each processing element. With these known $(n + 1/2)\,\Delta t$ values, all 64 processing elements now calculate $(n + 1)\,\Delta t$ values. However, according to the computational mesh arrangement (Fig. 1), only PE's 2 to 63 contain correct computed values at time $(n + 1)\,\Delta t$. It is therefore necessary to transfer $(n + 1)\,\Delta t$ values from PE 3 to PE 1 and from PE 62 to PE 64 to comply with the reflection boundary conditions at these points. These housekeeping operations, however, occupy only a small percentage of time in one cycle of computing 64 mesh points.

The radius r is now increased by Δr and the procedure repeated for the new row of circumferential mesh points. The computational field is thus traversed from the body surface to the outer edge of the far field for this value of time equal to $(n + 1)\,\Delta t$. Using these values of P, u, v, and ρ, the computational field is next traversed from the body surface to the far field to obtain dependent variables at the mesh points at time $(n + 2)\,\Delta t$. The steady state criterion used to stop the time step iteration is that the dependent variables on the body surface should not change by more than one per cent for ten successive iterations.

Should it be necessary to compute 128 mesh points instead of 64, the procedure would essentially remain the same. In this case mesh points 2 and 127 would be

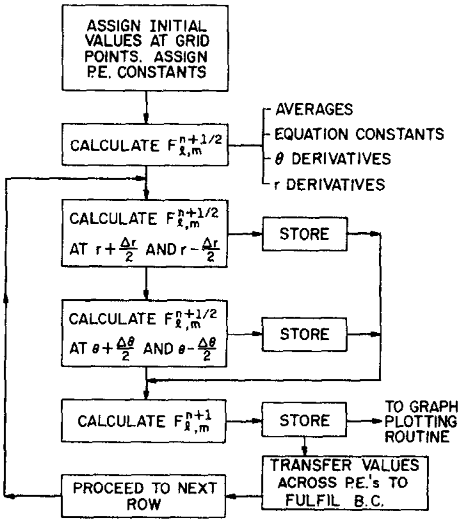

ASSIGN INITIAL VALUES AT GRID POINTS. ASSIGN P.E. CONSTANTS
CALCULATE $F_{l,m}^{n+1/2}$
AVERAGES
EQUATION CONSTANTS
θ DERIVATIVES
r DERIVATIVES
CALCULATE $F_{l,m}^{n+1/2}$ AT $r+\frac{\Delta r}{2}$ AND $r-\frac{\Delta r}{2}$
STORE
CALCULATE $F_{l,m}^{n+1/2}$ AT $\theta+\frac{\Delta\theta}{2}$ AND $\theta-\frac{\Delta\theta}{2}$
STORE
CALCULATE $F_{l,m}^{n+1}$
STORE
TO GRAPH PLOTTING ROUTINE
TRANSFER VALUES ACROSS P.E.'s TO FULFIL B.C.
PROCEED TO NEXT ROW

FIG. 4. Schematic computational block diagram.

the boundary points on the $\theta = 0$ and $\theta = \pi$ lines, respectively. The calculation would be performed in two blocks of 64 mesh points each. Mesh points numbered 64 and 65 would refer to the same point in the physical space, but would be necessary to obtain the correct forward and backward differences as the values are transferred across the PE's. Thus if we have only 64 processing elements, but we wish to compute 128 mesh points instead of 64, a total of 125 points can be accommodated in the physical space. It is also implied in the above description that old values of P, u, v, and ρ are retained long enough in memory to enable new values at a later time to be computed, in addition to being available to provide continuity where there are overlapping mesh regions.

A schematic layout of the computational procedure is shown in Fig. 4.

7. Treatment of Mesh Size Changes

A. *Mesh Size Change in the Tangential Direction*

A location in the cylinder problem, where the rapid change of field variables necessitates a change in the mesh size, is the region around the forward stagnation point where the horizontal component of the velocity is brought to zero. In any region where mesh size changes are accomplished on a parallel machine, it must be done in such a way that the parallelism is not destroyed.

The procedure for changing the mesh size in the tangential direction is illustrated in Figs. 5(a) and 5(b). Suppose that we need to change mesh sizes at point 14 on a row of circumferential points, Fig. 5(a). The points are numbered such that at the next point, point 15, the tangential increment would now be $2\Delta\theta$ instead of $\Delta\theta$. Under the usual procedure, point 12 variables would be calculated in PE 12, point 13 in PE 13, point 14 in PE 14 and point 15 in PE 15 and so on. Now while using the Lax–Wendroff two step scheme to calculate the field variables at point 13 and time level $(n + 1)\,\Delta t$ we need the field variables at points 12 and 14 at time level $n\,\Delta t$. Therefore all sums, differences, derivatives, etc., that are needed for the calculation of P, u, v, and ρ at $(n + 1)\,\Delta t$ and at point 13, just before the mesh size change, are available and can be calculated in parallel with the same instruction stream as that supplied to, say, point 11.

However, when we feed the same instruction stream to point 14, where the mesh size change begins, we run into trouble. For example the number of shifts performed in obtaining forward and backward differences for the finite difference approximations to the derivatives will be different in the process of obtaining the $(n + 1)\,\Delta t$ values of P, u, v, and ρ from the known $n\,\Delta t$ values. The parallelism will therefore be destroyed.

To maintain the parallelism while changing the mesh size in the tangential direction, we allot the calculation of the dependent variables P, u, v, ρ at points 1

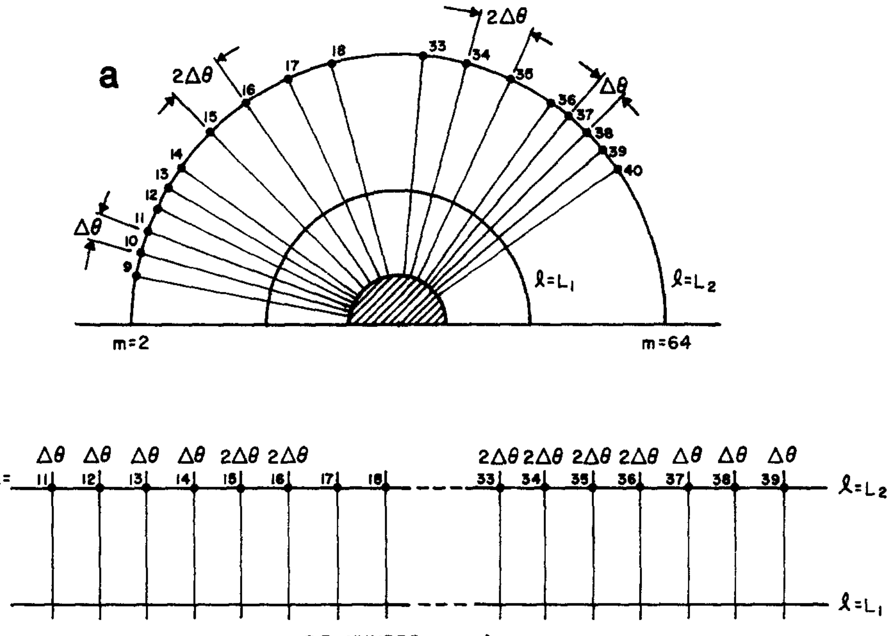


FIG. 5(a). Mesh size change in tangential direction. No overlapping points.

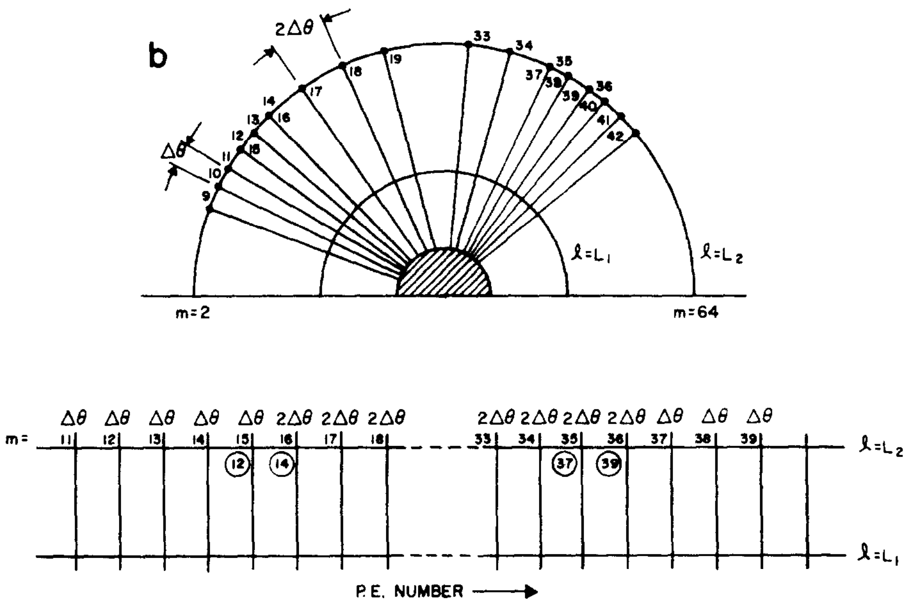


FIG. 5(b). Points superposed to change mesh size in tangential direction.

through 13 to PE numbers 1 through 13 as before. But now the variables at point 12 in the physical space, Figs. 5(a) and 5(b) are stored in PE number 12 and also in PE number 15, Fig. 5(b). So also the variables at point 14 are stored both in PE number 14 and PE number 16. Point 15 in Fig. 5(a) now becomes point 17. Two extra PE's, namely 15 and 16, are used to store and calculate the variables P, u, v, and ρ corresponding to points 12 and 14 in the physical space (Fig. 5(a)). Even under the new reordering of points the variables at mesh point 13 in the physical space can still be computed in the usual manner. This is possible because with the Lax–Wendroff scheme we need the values of P, u, v, and ρ at points 12 and 14, distant $-\Delta\theta$ and $+\Delta\theta$, respectively, from point 13, to compute $(n + 1)\,\Delta t$ values at point 13.

Remembering that we are computing in parallel, PE number 14 will attempt to calculate the $(n + 1)\,\Delta t$ values at point 14. In computing the finite differences representing $\partial P/\partial\theta$, $\partial u/\partial\theta$, and $\partial v/\partial\theta$ during the process of transferring values across PE's by shifts of the same amount as any other PE, PE number 14 will use the physical variables stored in PE 13 and PE 15, distant $-\Delta\theta$ and $-2\Delta\theta$, respectively, from PE 14 under the arrangement Fig. 5(b). The same instruction stream supplied to PE 14 as to PE 13 will compute erroneous values of P, u, v, and ρ in PE 14 for point 14. Similarly it can be shown that $(n + 1)\,\Delta t$ values computed by PE 15 are also wrong.

Coming now to PE 16 which actually contains the physical variables of point 14 in Fig. 5(b), we need the values of P, u, v, and ρ at $n\,\Delta t$ from PE 15 and PE 17 to find the values of $(n + 1)\,\Delta t$ at point 16. Since both PE 15 and PE 17 store variables which are $-2\Delta\theta$ and $+2\Delta\theta$, respectively, from PE 16, we are assured of obtaining the correct values for the finite difference approximations to the partial derivatives $\partial P/\partial\theta$, $\partial u/\partial\theta$, $\partial v/\partial\theta$, etc. Therefore PE 16 computes the correct values for P, u, v, and ρ at $(n + 1)\,\Delta t$ at physical point 16 under the new arrangement of points. Thus all the PE's up to PE 13 compute correct values of the physical variables at $(n + 1)\,\Delta t$, while PE 16 calculates the correct values for point 16. PE's 14 and 15 are needed for computing $(n + 1)\,\Delta t$ values at 13 and 16, respectively, but themselves do not compute correct $(n + 1)\,\Delta t$ values at points 16 and 12 which they represent. To ensure that PE's 15 and 14 contain correct $(n + 1)\,\Delta t$ values for later computation of $(n + 2)\,\Delta t$ values, we need to update the values of the physical variables in PE's 15 and 14 from correct $(n + 1)\,\Delta t$ values computed by PE's 12 and 16, respectively. This is done by shifts of the specified amounts across the PE's after calculating all the $(n + 1)\,\Delta t$ values at all the mesh points, since at this time there are other reassignments to be made, and a number of these may be done simultaneously.

To reduce the tangential mesh dimension we again overlap two points in the physical space by points 35, 37 and 36, 39. The correct values at $(n + 1)\,\Delta t$ are computed in PE 35 and PE 39. In order to have the correct values at $(n + 1)\,\Delta t$

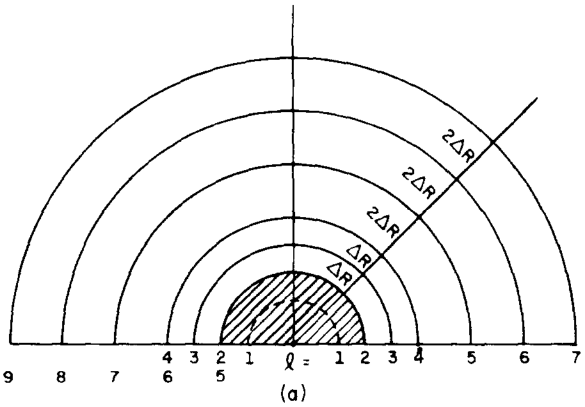


FIG. 6(a). Physical mesh layout.

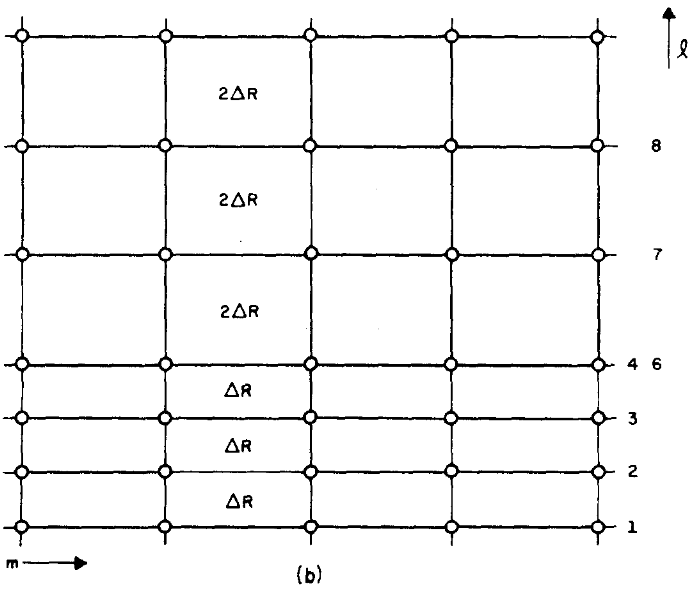


FIG. 6(b). P. E. Memory storage for mesh size change in radial direction.

in PE's 36 and 37 for computing $(n + 2)\,\Delta t$ values, the physical variables in PE's 36, and 37 must again be updated at the appropriate time before the commencement of the $(n + 2)\,\Delta t$ computation.

B. *Mesh Size Change in the Radial Direction*

While two extra PE's are involved for each mesh size change in the tangential direction, the mesh size change in the radial direction involves extra computer memory. The method of changing the mesh dimension from Δr to $2\Delta r$ in the radial direction is illustrated in Figs. 6(a) and 6(b). As the radius increases, the derivatives $\partial p/\partial r$, $\partial u/\partial r$, and $\partial v/\partial r$, decrease and it becomes possible to increase the mesh size in the radial direction without sacrificing accuracy. Each circumferential row of points is designated by the value of the index l. For the sake of illustration let it be required to change mesh sizes in the radial direction at the row corresponding to $l = 4$. To compute $(n + 1)\,\Delta t$ values for points on row 3 we need $n\,\Delta t$ values on row 2 and row 4 and these are readily available. To compute $(n + 1)\,\Delta t$ values on row 4 we need points on row 7 and row 2 at time $n\,\Delta t$, and also a change in the instruction stream to the PE's at this point in the computation. Instead of changing the instruction stream, we choose to label points on row 4 as also being the points on row 6 and we label the points on row 2 as also being the points on row 5. After calculating points on row 3 in the usual manner, we now increase the row index l in the calculation to 6 and compute points on row 6. Calculation of points on row 6 at $(n + 1)\,\Delta t$ requires points on row 5 and row 7. These latter rows being spaced at $-2\Delta r$ and $+2\Delta r$ from row 6 involve a change in the instruction stream to all the PE's which is accomplished simultaneously. Thus changing the mesh size in the radial direction involves storing the requisite information for two rows of points twice, suitably labeled so as to be available at the right location. Any number of changes in mesh size in the radial direction may be made as dictated by the physical problem.

8. Results and Discussion

Figure 7 shows the surface tangential velocity obtained from a simulated computation for a Mach number of 0.05. Since the flow can be considered essentially incompressible at this low Mach number, the result can be compared

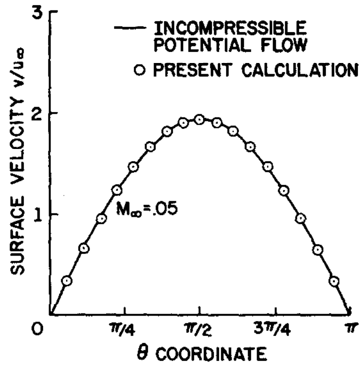


FIG. 7. Comparison of cylinder surface velocity with potential theory results.

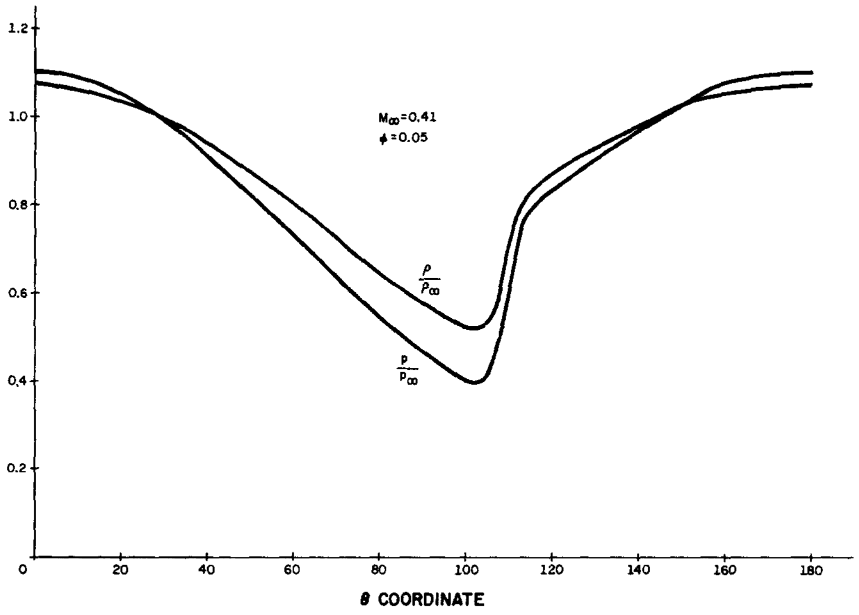


FIG. 8. Normalized pressure and density at cylinder wall for a free stream Mach number of 0.41.

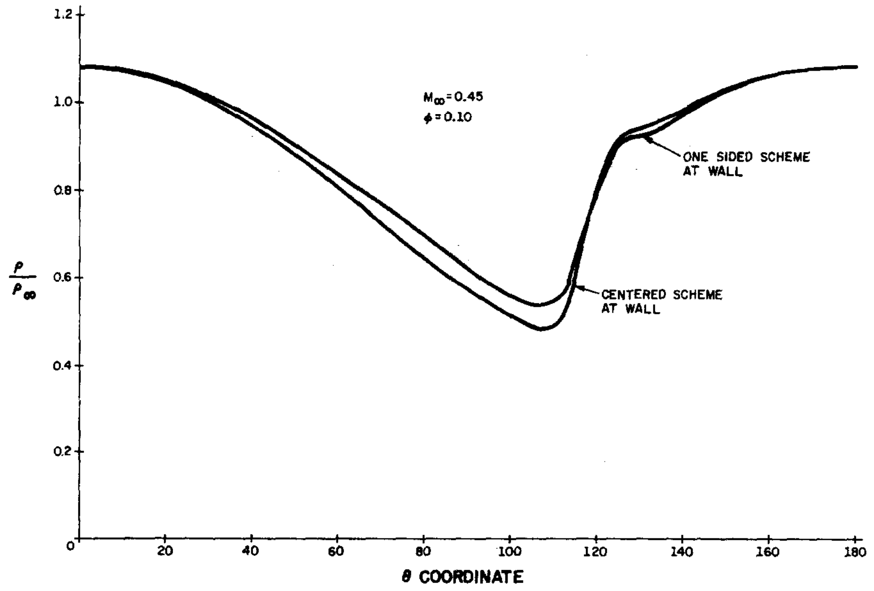


FIG. 9. Comparison of results for different formulations of wall boundary condition.

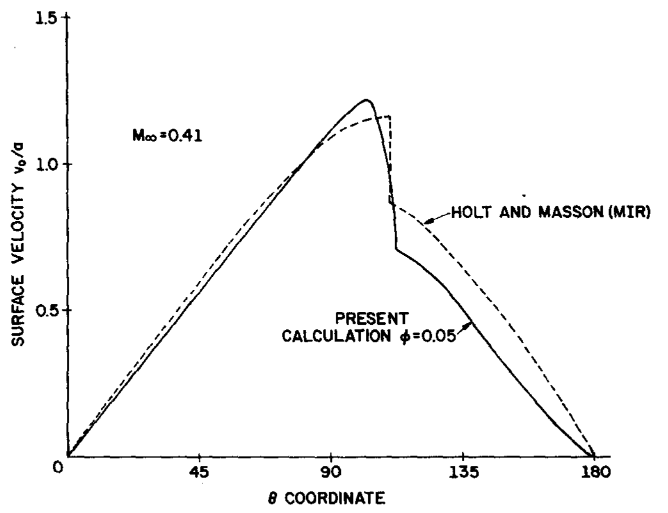


FIG. 10. Comparison of present results with those obtained by Holt and Masson by the method of integral relations.

with that obtained from incompressible potential flow theory. The curves show that the cylinder surface velocity from the present method shows excellent agreement with that obtained from potential flow theory. At this low Mach number, the value of the artificial viscosity parameter ϕ is small and its influence on the incompressible flow computation is negligible.

Values of p/p_∞ and ρ/ρ_∞ for a free stream Mach number of 0.41 are shown in Fig. 8. The value of ϕ for this computation is 0.05. From the curves we can clearly discover the recompression shock located at $\theta = 110$ degrees on the cylinder arc.

Figure 9 shows the effect of different finite difference formulations of the cylinder wall boundary condition. The radial velocity at the wall is forced to zero in both cases. For a free stream Mach number of 0.45 and a value of $\phi = 0.1$, a one-sided scheme and a centered scheme are used at the wall. For the one-sided scheme the additional row of points shown in Fig. 1 (at $l = 1$) is no longer required. The curves in Fig. 9 show that the shock location is not appreciably affected by the type of scheme at the wall. However the strength of the shock is changed, with the centered scheme yielding a stronger shock. Since the centered scheme is spatially more accurate than the one sided scheme, it would be reasonable to assume that the solution obtained with it would come closer to the true solution. Figure 10 compares the nondimensional surface velocity computed by the present method with the result obtained by Holt and Masson [8] by the method of integral relations. The velocity curve from the present calculation is smooth although there is some difference in the shock strength obtained by the two methods. The shock location is in good agreement, while the velocity profiles agree better on the forward side of the cylinder than on the rearward side.

9. Conclusions

For the new generation of computing machines, the importance of correlating the formulation of the physical initial-boundary value problem with the actual computational algorithm to obtain a high degree of machine utilization is demonstrated. Unless this is done at every stage of the computation, much of the advantage offered by the new machines, like the parallel processing Illiac IV, is lost. The time dependent method of realizing the steady state solution used in this work is shown to be eminently suited for parallel computation, and is easily modified to include more complex problems. Sample results presented are in good agreement with results obtained from potential theory and from other investigators. Other results will be presented elsewhere [9].

References

1. R. Magnus and H. Yoshihara, *AIAA J.* **8** (1970), 2157–2162.
2. E. M. Murman and J. D. Cole, *AIAA J.* **9** (1971), 114–121.
3. T. C. Tai, Applications of the method of integral relations to transonic airfoil problems, *AIAA* Paper No. 71–98, Jan. 1971.
4. J. L. Steger and H. Lomax, *AIAA J.* **10** (1972), 49–54.
5. C. P. Kentzer, Computations of time dependent flows on an infinite domain, AIAA Paper No. 70-45, Jan. 1970.
6. G. Moretti, "Transient and Asymptotically Steady Flow of an Inviscid Compressible Gas Past a Circular Cylinder," Polytechnic Institute of Brooklyn, Report No. 70-20, 1970.
7. S. Rajan, A Computational Technique for High Subsonic Compressible Flow past a circular cylinder on a Parallel Computer, Center for Advanced Computation Report, University of Illinois, Urbana, No. 50, 1972.
8. H. Holt and B. S. Masson, The Calculation of High Subsonic Flow past bodies by the Method of Integral Relations, "Proc. of the Second International Conference on Numerical Methods in Fluid Dynamics, Berkeley," Springer-Verlag, N.Y., 1971.
9. S. Rajan, Evaluation of Numerical Viscosity Effects in Transonic Flow Calculations, Paper No. 73-131 presented at the AIAA 11th Aerospace Sciences Meeting, Jan. 10–12, 1973, at Washington D.C.
10. G. Moretti, "The Choice of a Time-Dependent Technique in Gas Dynamics," Polytechnic Institute of Brooklyn, Report No. 69-26, 1969.