FLOSOLVER: A PARALLEL COMPUTER FOR FLUID DYNAMICS

U. N. SINHA, M. D. DESHPANDE and V. R. SARASAMMA
National Aeronautical Laboratory, Bangalore 560 017, India.

ABSTRACT

This paper describes Flosolver, a parallel processing computer designed and built at the National Aeronautical Laboratory, (NAL). The computer has two nodes each having four processors based on Intel 8086–8087 chips. In each node one of the processors acts as host and has access to a section of the private memory of the remaining three processors through the Multibus. Inter-node communication is done using parallel ports. Synchronization and inter-processor communication are done by passing the message through the global memory. Prior to execution the host processor loads the absolute codes from a disk to the respective processors. Several fluid dynamical problems of practical interest have been solved on Flosolver using concurrent algorithms, and these show that it is comparable in speed with mainframes available in the country.

INTRODUCTION

COMPUTERS have come to play an important role in the development of fluid mechanics during the last two decades[1]. The differential equations governing fluid flow have been known for about two centuries, but till recently it had not been possible to solve them except in very simple cases. Applications of these equations to problems of practical interest had to wait till adequate computing resources were developed. The complexity of the problems that can be solved depends on the power of the machine available. For example, computing speeds of the order of hundreds of MFLOPS (Million FLoating Point Operations per second) are now feasible; a decade ago this speed was only a few MFLOPS. The growth rate of the speed has been so enormous that machines performing GFLOPS (1000 MFLOPS) are now a practical reality. As the power of the machines increases more difficult problems have been tackled, but there is a growing demand for still more powerful computers among fluid dynamicists.

However, the scene in India is different. Computer speeds generally available are less than one MFLOPS. This clearly puts the practice of computational fluid dynamics (CFD) workers at a great disadvantage. Acquisition of powerful main frames poses problems. On the other hand there is the possibility of using several smaller machines to work together on a bigger problem. This has become attractive because of the revolutionary growth in microprocessor technology. But this is possible only if the hardware and the associated system software support concurrent/parallel processing, and the nature of the problem solved is such that the algorithm is amenable to such a computation. It may be pointed out that this requires the synchronization of all the processors to work together, and the handling of inter-processor communication.

Implementation of such ideas has been termed concurrent or parallel processing[2,3]. In parallel processing various processors act on the same task whereas in concurrent processing various processors act on independent sub-tasks which get integrated at a later stage. The two ideas are conceptually similar, the difference being one of level. Implementation at a coarser level is termed as concurrent processing and that at a finer level as parallel processing. However, we use the two terms interchangeably, even though the term concurrent processing is more appropriate in the present context.

Many fluid mechanics problems can be divided into several parts such that each one of them can be assigned to a separate processor. Although these ideas look simple and obvious they are not easy to implement. It may be mentioned here that even though the theoretical designs of a highly parallel computer were discussed in the early[4] 60's it was only in the 80's that the first machine was made operational[3]. It turns out that for the same number-crunching capability, such an approach is far more cost effective as compared to the calculations made on a main frame machine, provided appropriate software is available.

With this in view we embarked in 1986 on the Flosolver project, whose objective was "to design, develop, fabricate and use a suitable parallel processing computer for the application to fluid dynamical

and aerodynamical problems". This paper describes our experience to-date.

PRESENT APPROACH

The computational approach to solve complex flow problems demands large computing requirements[1]. Since these problems are amenable to concurrent processing, the objective was to build a machine which is tailor-made for this purpose. However, the need for custom-made fabrication and for the system software, to integrate these processing elements and to handle inter-processor communication, are serious constraints. With this in mind various contemporary approaches were examined to evolve the best suited one for our purpose.

These contemporary approaches can be classified into two broad groups, depending on whether the memory is shared by all the processors or each processor has its own local memory and inter-processor communication is achieved through dedicated hardware. Sequent Balance, Flex and Butterfly are some typical machines belonging to the former class; in the latter class, we have the Cosmic Cube from Caltech or iPSC from Intel and Columbia's concurrent machine. It may be noted that a clear-cut distinction between these two classes is not always possible as there are many examples where both features are combined. Cedar is an example of this class.

The Cosmic Cube from Caltech has been most widely discussed in the literature for concurrent processing. This is a network of 64 Intel 8086 processors. These processors are connected by a network of point-to-point communication channels in the topology of a binary 6-cube. The network has node-to-node bidirectional 2 Mb/s serial links. Each node has its own operating system. Thus the Cosmic Cube is a multiple instruction and multiple data (MIMD) type machine which uses messages passing for communication between concurrent processors.

Each one of the parallel machines mentioned above had custom-built hardware, at least in part and associated software that was very specific to the machine concerned. In the NAL context the custom-building of the hardware or even procurement at component level was out of the question because of the time scale involved, as well as the uncertainty about their availability in the Indian market.

This narrowed down the choice to basing the design on available hardware with minimal modification. Also, an attempt was made to use the available sequential software to its fullest extent, so that the development of software for building the machine was reduced to a minimum. The choice of a private memory for each processor to work upon its assigned task, and of a global memory to achieve inter-processor communication and synchronization, seemed logical.

Commercially, computers have emerged integrating on the same card the CPU (Central Processor Unit), memory and other support chips needed for handling input/output (I/O), etc. The growth in multiprocessing has given rise to the idea of a bus through which a common memory could be accessed by various processors residing on different boards. The bus is a series of control, address and data lines, supported by the peripheral chips meant for controlling the priority, timing, etc. Synchronization of the tightly coupled processors and inter-processor communication involving large volume of data are difficult tasks using commercially available buses even when the number of processors is small, since they are not designed with the objective of concurrent processing. It is for this reason, that the hardware used in the machines mentioned earlier had to be custom-built.

Intel has marketed single board computers with a standard bus interface known by the trade name Multibus-I. These seemed to provide convenient building blocks for the proposed machine at NAL, and were adopted for the Flosolver.

On the software side, the sequential software available on the single-board computer was planned to be supplemented by writing an executive program which will synchronize and handle inter-processor communication. The Concurrent Executive of the Flosolver was developed for this purpose.

ARCHITECTURE AND SYSTEM SOFTWARE

The Flosolver machine built at NAL has two nodes—node 1 and node 1A, each having four processors based on Intel 8086–8087 chips at 8 MHz clock speed. Node 1 has limited onboard memory of 128 KB per processor and a global memory shared by each processor of 512 KB. To overcome the difficulty of limited onboard memory, Intel cards having onboard memory of 512 KB were acquired. Four of these cards constitute node 1A.

In each node one of the processors acts as the host which runs Intel's real time operating system called iRMX. The host processor controls a 36 MB Winchester disk, floppy drive, dot matrix printer and console. It also has access to a section of the

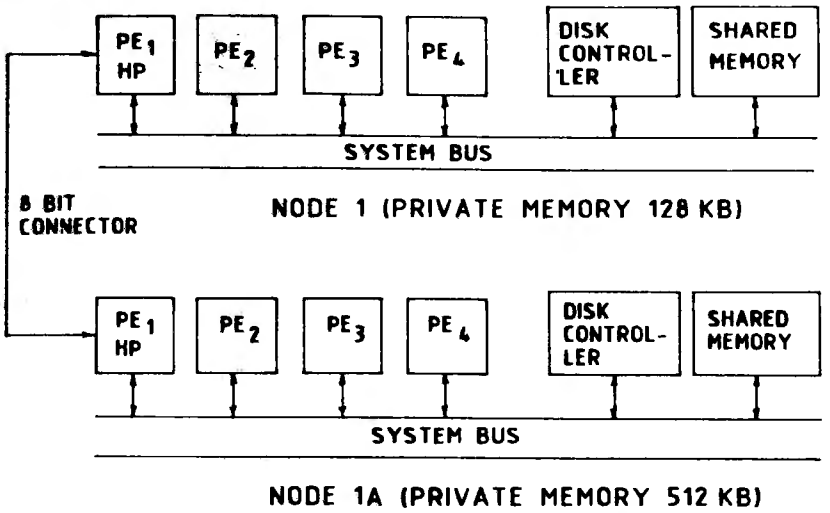

Figure 1. Architecture of Flosolver Mk1B. PE_n: nth processing element (Intel 8086/8087 Microprocessor). HP: Host Processor.

private memory of the remaining three processors through the Multibus, as shown in figure 1. Nodes 1 and 1A are complete systems having four processors per node. Processors in each of these are coupled by the shared memory, in sharp contrast to machines like the Cosmic Cube. Internode communication is done using parallel ports on these cards, and this results in having two time scales of communication:

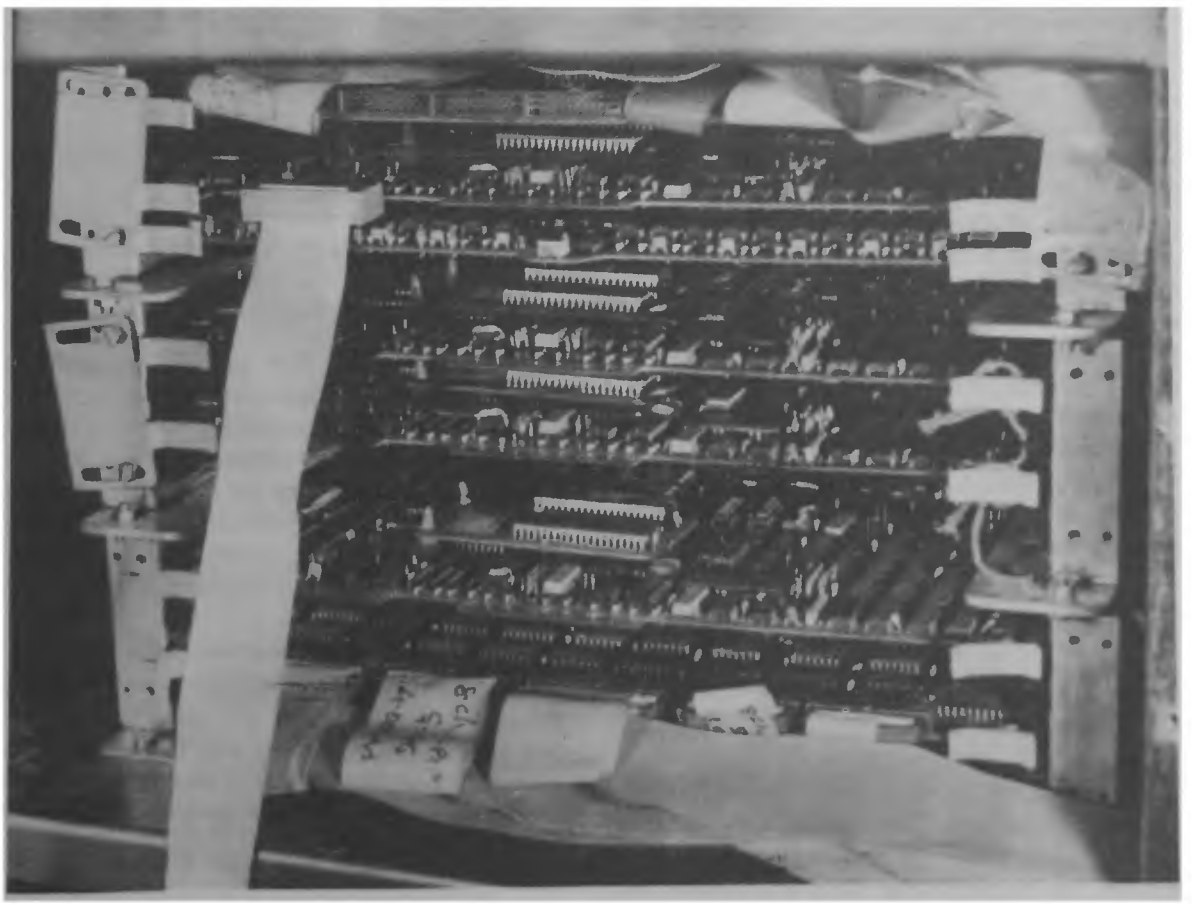

Figure 2. Photograph of node 1 of Flosolver.

(i) Short range communication: inside a node, very fast.
(ii) Long range communication: between nodes, an order of magnitude slower than short range communication.

Figure 2 is a photograph of node 1 of the Flosolver. Seven boards can be seen in the rack. They are, from the top, the disk controller card, CPU-1, global memory card, CPUs −2 −3 and −4 and the communication controller card for serial I/O of the first CPU.

The application codes meant for solving the flow problems are written in Fortran. These are compiled and linked to get the absolute code, which is then stored on the disk. Prior to execution, the host processor loads the absolute codes from the disk to the respective processors under the control of the Concurrent Executive which is an important part of the system software. Primitives of synchronization and inter-processor communication are Fortran callable subroutines and they are a part of the library of the Concurrent Executive. Inter-processor communication is handled by the Concurrent Executive by passing the message into global memory.

APPLICATION SOFTWARE

As mentioned earlier, the Flosolver project is aimed at building a machine to solve fluid-dynamical problems which are computation-intensive. The governing partial differential equations are solved numerically using concurrent algorithms. The following two-dimensional problems have been solved using concurrent algorithms: (a) Laplace equation, (b) Transonic small perturbation equation (TSP), (c) Navier-Stokes equations, and (d) Compressible Euler equations.

These equations cover a wide range of CFD problems and are of practical importance. Moreover, the structure of the application programs meant to solve these equations is the same and hence only one problem, namely the Navier-Stokes equations, will be discussed in detail. A brief discussion of the TSP equation is presented subsequently.

Navier-Stokes equations

The problem of laminar axisymmetric jet impingement on a plate[5] is solved here using a concurrent algorithm[6]. The governing differential equations for the dependent variables, namely the

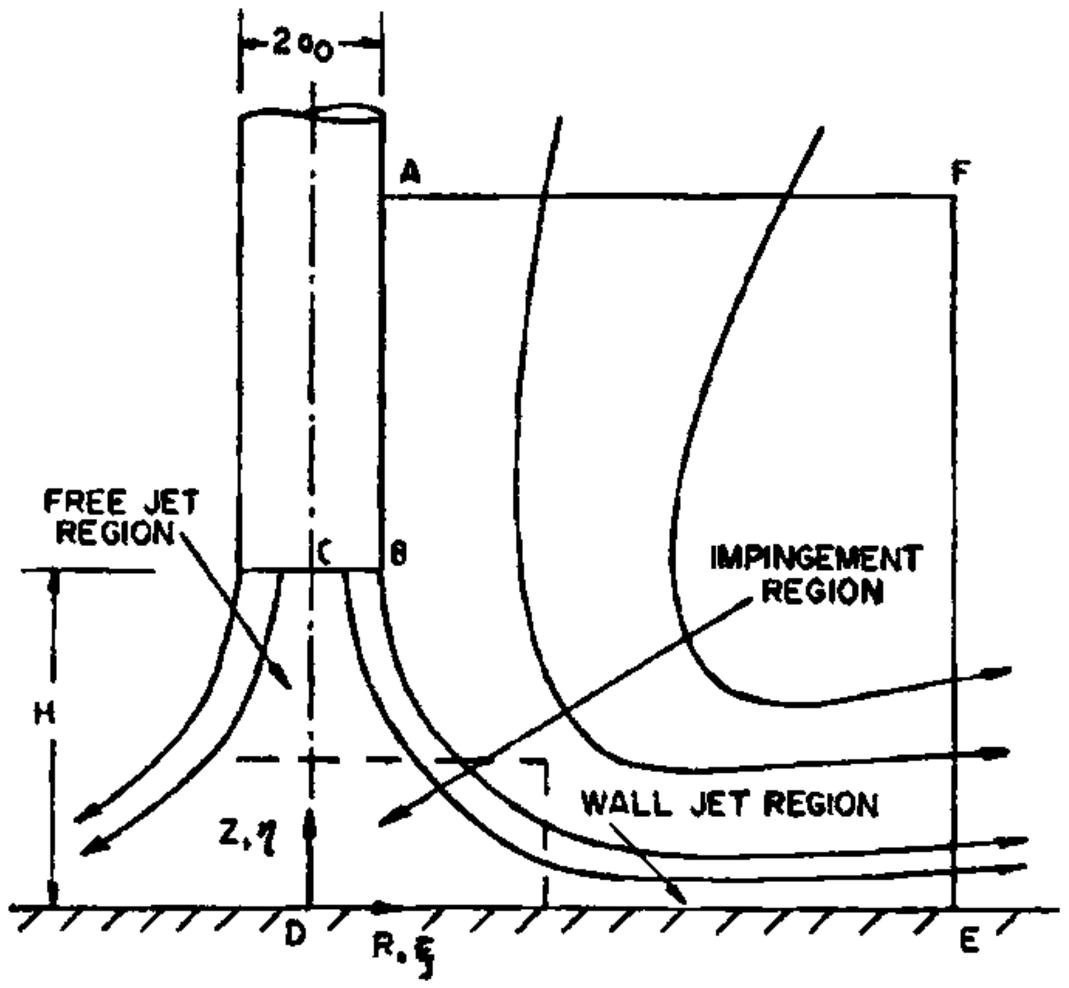

Figure 3a. Flow configuration for jet impingement problem.

stream function ψ and the vorticity ω are

$$\omega = -\frac{1}{R}\left(\frac{\partial^2 \Psi}{\partial R^2} + \frac{\partial^2 \Psi}{\partial Z^2}\right) + \frac{1}{R^2}\frac{\partial \Psi}{\partial R} \qquad (1)$$

$$R^2\left[\frac{\partial}{\partial Z}\left(\frac{\omega}{R}\frac{\partial \psi}{\partial R}\right) - \frac{\partial}{\partial R}\left(\frac{\omega}{R}\frac{\partial \psi}{\partial Z}\right)\right] -$$

$$-\frac{1}{\mathrm{R_e}}\frac{\partial}{\partial Z}\left[R^3\frac{\partial}{\partial Z}\left(\frac{\omega}{R}\right)\right] -$$

$$-\frac{1}{\mathrm{R_e}}\frac{\partial}{\partial R}\left[R^3\frac{\partial}{\partial R}\left(\frac{\omega}{R}\right)\right] = 0, \qquad (2)$$

where $R = r/a_0$ and $Z = z/a_0$ are the non-dimensional cylindrical coordinates with a_0 as the tube radius (figure 3a). The computation is performed over the computational domain shown as ABCDEF. The boundary conditions are specified for Ψ and ω all around this computational domain[5]. The problem is solved by the point relaxation iterative technique, an algorithm ideally suited for concurrent computation.

A coordinate transformation $(R, Z) \rightarrow (\xi, \eta)$ is used so that the grid points in the transformed plane are distributed uniformly providing more points in the physical plane where the flow variables are expected to change rapidly. Finite difference equations, derived from the transformed equations, are:

$$\phi_P \sum_j (A_j + B_j) = \sum_j [\phi_j (A_j + B_j)] - V_P \bar{d}_P, \qquad (3)$$

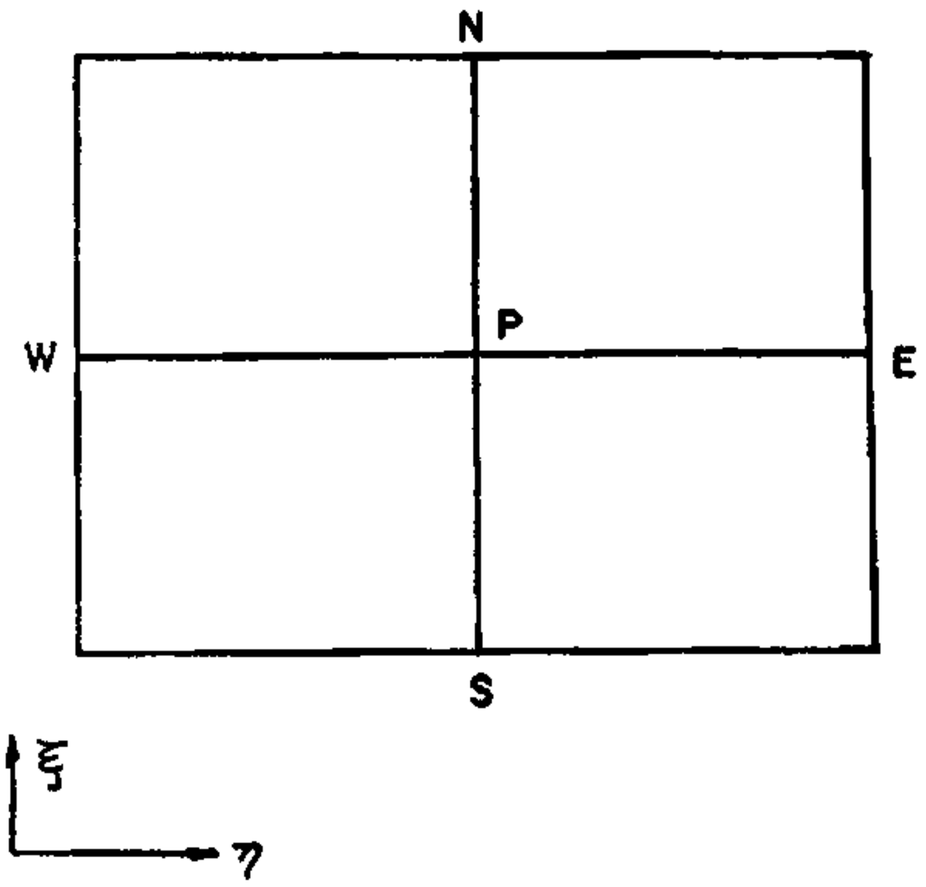

Figure 3b. Grid for numerical scheme.

where ϕ represents the dependent variables. Summation is carried out over the four neighbouring points E, W, N and S (see figure 3b) and the suffix P refers to the nodal point P. Also,

$$V_P = \tfrac{1}{4} R_P(\eta_N - \eta_S)(\xi_E - \xi_W). \qquad (4)$$

$$A_E = \bar{a}_P[(\psi_N + \psi_{NE} - \psi_S - \psi_{SE}) + |(\psi_N + \psi_{NE} - \psi_S - \psi_{SE})|]/8 \qquad (5)$$

with similar expressions for A_W, A_N and A_S, and

$$B_E = \frac{b_{1E} + b_{1P}}{8} \cdot \frac{R_E + R_P}{\xi_E - \xi_P}(\eta_N - \eta_S) \qquad (6)$$

with similar expressions for B_W, B_N and B_S. Further details can be found in reference 5.

When one solves the two coupled systems of difference equations (3) on a single CPU, a sequential method is followed. But when these are solved on N CPUs ($N = 4$ in this example) the computational domain is subdivided into N parts and each CPU solves these systems on its assigned subdomain. To calculate the values of ψ or ω at point P (figure 3b) we need their values at the neighbouring nodal points N, E, S, etc. and the latest values available are used. Once the entire computational field (or the subdomain assigned to the particular CPU in the

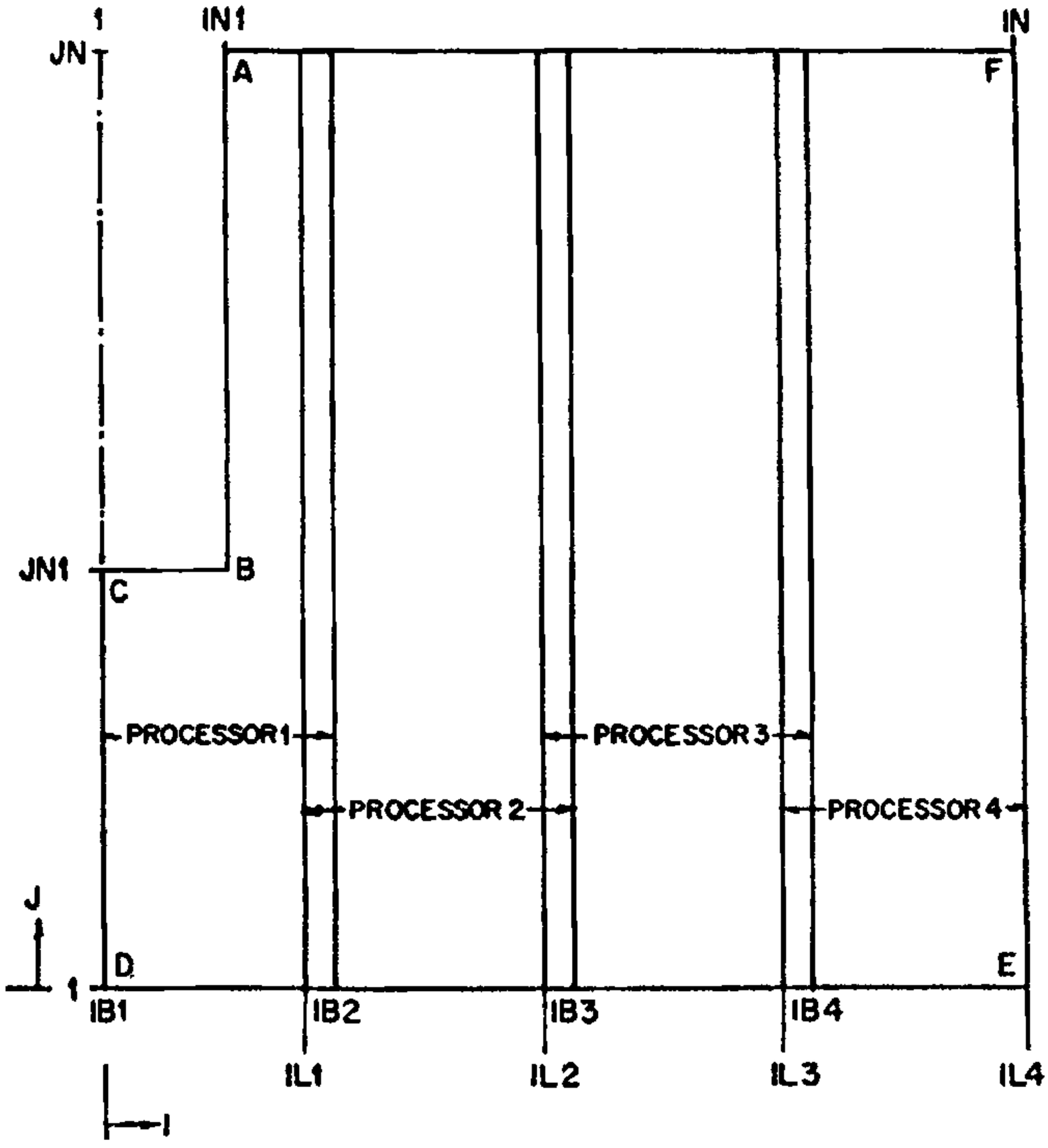

Figure 3c. Computational subdomains for jet impingement problem.

case of a concurrent algorithm; this will be discussed in detail below) is swept during one cycle of an iteration, the boundary conditions are recomputed from the latest values of ψ and ω and the process is continued until convergence is achieved.

Now the details of the concurrent algorithm will be given. The computational domain is divided into four parts with overlap as shown in figure 3c. Processor 1 has its computational subdomain defined by $R(IB1) \leq R \leq R(IL1)$. The values on the boundary line CD($I = IB1 = 1$ and $1 \leq J \leq JN1$) and AB ($I = IN1$ and $JN1 \leq J \leq JN$) are obtained from the boundary condition and the values on the boundary of the subdomain with $I = IL1$ are obtained from processor 2. The computational subdomain for processor 2 is from $I = IB2$ to $I = IL2$ and for processor 3 it is from $I = IB3$ to $I = IL3$. We note that $IB2+1 = IL1$ and $IB3+1 = IL2$. The computations on processor 2 are done only for $R(IB2+1) \leq R(I) \leq R(IL2-1)$ and the values of ψ and ω on the boundaries of this subdomain, viz. for $I = IB2$ and $I = IL2$, are obtained from the neighbouring processors 1 and 3, respectively. Similarly processors 3 and 4 have their own subdomains of computations.

The computational steps are as follows. The starting values for ψ and ω are assigned at each nodal point. During an iteration each processor updates the values of ψ and ω at each nodal point in its subdomain by a systematic sweep. At the end of each iteration the values of ψ and ω at the boundaries of the subdomains are updated using the values from the neighbouring processors. Moreover the boundary conditions of the problem applied on ABCDEF (figure 3a) are also updated. At this step synchronization of the CPUs is required. Once it is ascertained that all the CPUs have completed the current iteration they are allowed to proceed to the subsequent iteration.

The results obtained by these calculations are indicated in figure 3d. This streamline pattern corresponds to Reynolds number $R_e = 1000$ and height of the jet exit plane $H = 4$ tube radii. These results were obtained on a (61×36) grid which is the same as that used earlier[5]. With this size the program could not be fitted on a single machine for sequen-

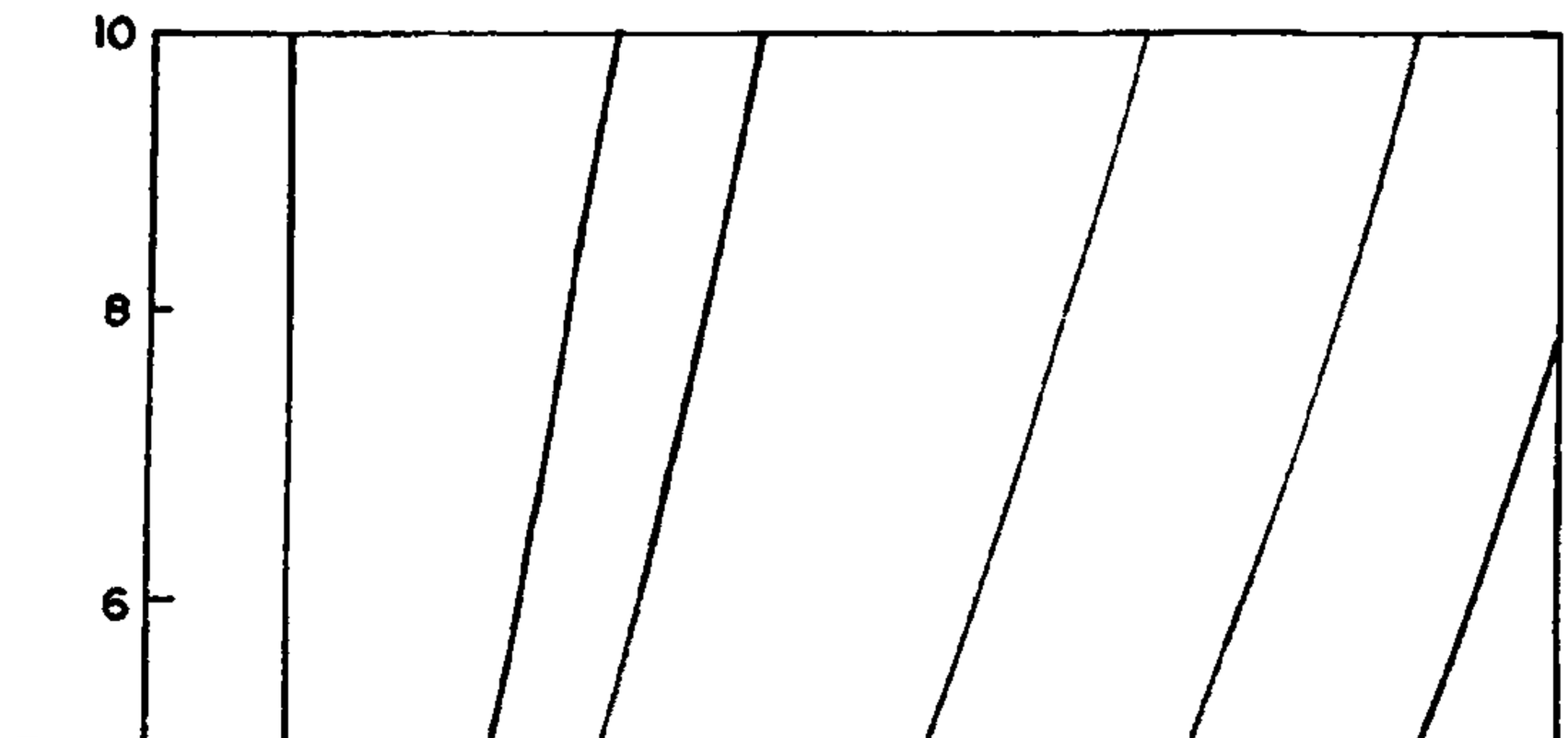

Figure 3d. Streamlines. $R_e = 1000$, $H = 4$.

tial operation. Thus the problem that was too big for a single processor was solved using four processors.

Transonic small perturbation equation

The TSP equation is known for its nonlinearity and mixed character— it is elliptic in the subsonic region and hyperbolic in the supersonic region. The TSP equation for flow past an NACA 0012 airfoil at zero incidence has been solved using a concurrent algorithm. The governing equation for perturbation potential ϕ is

$$\{1-M_\infty^2-(\gamma+1)M_\infty^2\phi_x\}\phi_{xx}+\phi_{yy}=0. \qquad (7)$$

where M_∞ is the free stream Mach number and γ is the ratio of specific heats.

The boundary conditions for flow tangency at the airfoil surface, defined by $y=f(x)$ say, are

$$\frac{df}{dx}=\frac{\partial\phi}{\partial y}(x,0). \qquad (8)$$

Moreover the perturbation potential ϕ decays to zero at infinity.

At a grid point the form of the difference equation depends upon whether the flow is subsonic (case 1) or supersonic (case 2).

Case 1. Subsonic grid point

$$V_{i,j}^n[\phi_{i+1,j}^n-2\phi_{i,j}^{n+1}+\phi_{i-1,j}^n]\frac{1}{\Delta x^2}$$

$$+[\phi_{i,j+1}^{n+1}-2\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}]\frac{1}{\Delta y^2}=0, \qquad (9)$$

where $V_{i,j}^n$ is the discrete approximation of $1-M_\infty^2-(\gamma+1)\phi_xM_\infty^2$ based on the values from the previous iteration; the superscript denotes the number of iterations and subscripts (i, j) denote the computational grid point.

Case 2. Supersonic grid point

$$V_{i,j}^n[2\phi_{i,j}^{n+1}-\phi_{i,j}^n-2\phi_{i-1,j}^{n+1}+\phi_{i-2,j}^n]\frac{1}{\Delta x^2}$$

$$+[\phi_{i,j+1}^{n+1}-2\phi_{i,j}^{n+1}+\phi_{i,j-1}^{n+1}]\frac{1}{\Delta y^2}=0. \qquad (10)$$

In both cases ϕ_{xx} near the body boundary point is approximated as

$$\phi_{xx}(X,\Delta)\approx\frac{2}{\Delta x(\Delta y+2\Delta)}\phi''(x,\Delta+\Delta y)$$

$$-\frac{2}{\Delta x(\Delta y+2\Delta)}\phi''(x,\Delta)-\frac{2}{(\Delta y+2\Delta)}\phi_x''(x,0). \qquad (11)$$

The computational domain is divided into four parts as shown in figure 4. There are two overlapping lines in this case which is therefore slightly different from that in the previous example. This is basically to accommodate the algorithm for line relaxation at supersonic points as may be seen from equation (10).

The difference equations (9) to (11) have been solved iteratively and the C_p distribution for a typical Mach number $M_\infty=0.95$ is shown in figure 4. The calculation was carried out on a (121×18) grid.

EFFICIENCY OF FLOSOLVER Mk1

The efficiency of concurrent computing is defined as

$$\eta=\frac{T_s/N}{T_c}$$

where T_s is the time taken by one of the N processors to solve the problem sequentially, T_c the time taken by all the N processors working concurrently to solve the same problem and N the total number of processors.

The efficiency is less than unity because of the overhead due to synchronization, idling and interprocessor data transfer. One can expect that the efficiency will go down as the number of processors N increases. In the present context an efficiency better than 95% has been realized when only one node was used in which case, the communication overhead is very small. When both the nodes were used the efficiency dropped to 90%. It may be remarked that the efficiency is not expected to deteriorate further even if one uses up to 16 nodes with 64 processors because the communication overhead can be overlapped.

To get the maximum efficiency, the optimal choice of subdivision of work is such that all the processing elements complete their share of the computing load simultaneously. It is not always straightforward to equalize the load due to the fact that I/O operation is being handled by only the host CPU; the problem is more severe when the geo-

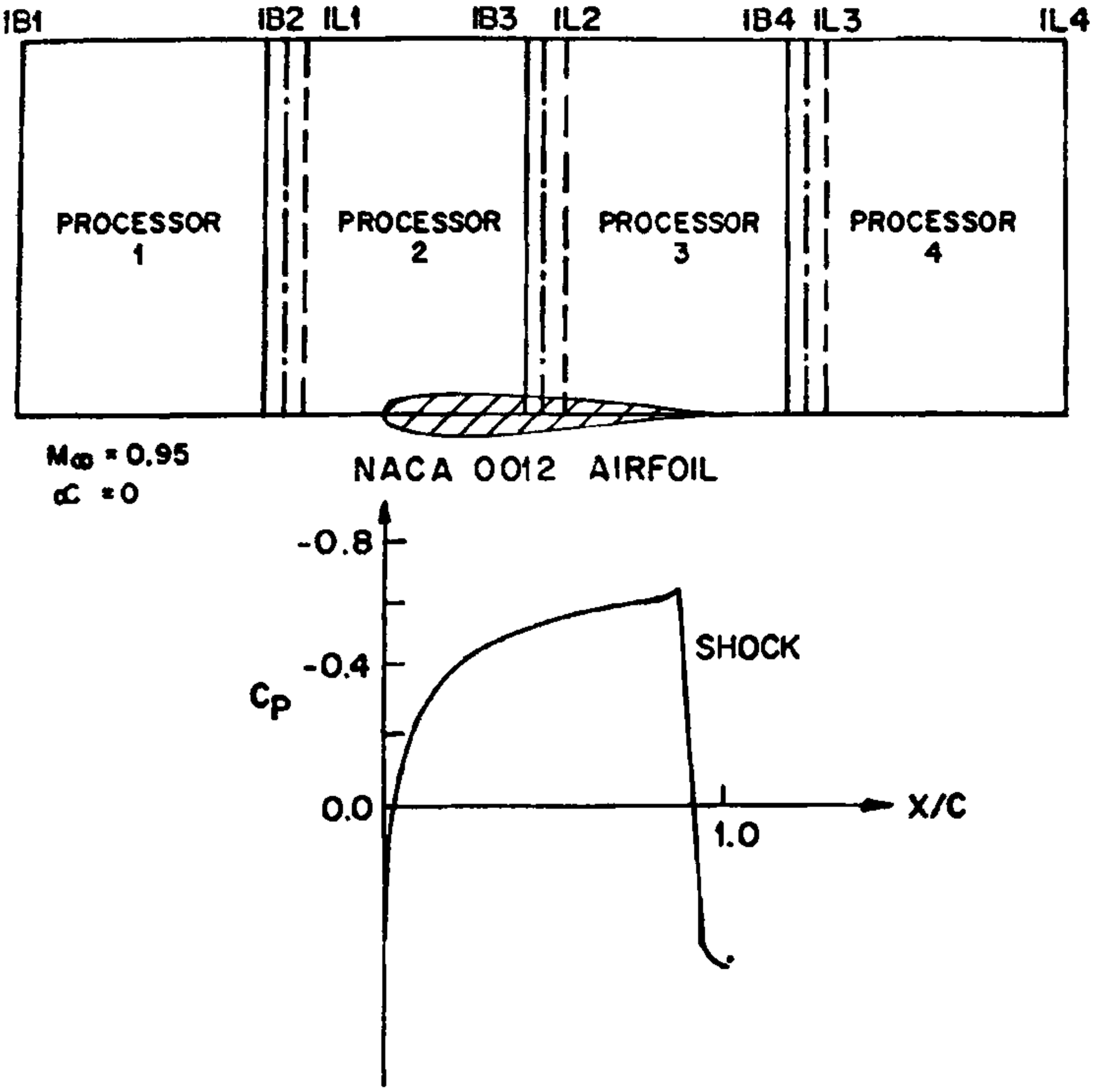

Figure 4. Computational subdomains and C_p distribution for TSP equation.

metry involved is complex. As an example, in the case of Navier-Stokes equations when (IB1, IB2, IB3, IB4) were taken to be (1, 13, 29, 45) respectively the computational time needed was 12 min (see figure 3c). However, by fine tuning of the load distribution this time was reduced to 9 min 34 sec and the corresponding IB's were (1, 18, 32, 46). If the geometry is complex and if one chooses a crude grid the load distribution may not be equalized easily[6].

Experiments have shown that Flosolver Mk1 is comparable in speed with mainframes available in the country today.

DISCUSSION

For many fluid mechanics problems concurrent algorithms are well suited. These ideas were discussed in reference 7 and have been realized in Flosolver Mk1. It may be recalled that the Flosolver Mk1 is a concurrent machine in which synchronization and inter-processor communication are done through shared (global) memory and the unit task is done in local memory. For interfacing each processor with the global memory 'Multibus I' with its standard bus arbitration logic is used.

Multibus I has to be adapted by suitable programming to meet the requirements of the Concurrent Executive. The library of system software includes the procedures for synchronization, inter-processor communication and load estimation on the processors.

Flosolver Mk1 is already a good match for the main frames whose overall cost is an order of magnitude higher. The implication is not that the Flosolver Mk1 is a substitute for a main frame; but if the objective is to solve a specific problem, concurrent computing is a very cost effective alternative. A single CPU board is almost a nonentity for CFD calculations but when many of these are combined they become a force to reckon with.

At the time of submitting of this paper, Flosolver Mk2 having four processing elements based on Intel 80386–80387 is also operational. It is about 3.5 times as powerful as Flosolver Mk1.

ACKNOWLEDGEMENTS

It is our pleasure to thank Prof. R. Narasimha for his constant encouragement and continued support.

Thanks are also due to our colleagues at NAL, Mr B. R. K. Iyengar, Dr A. Krishnan, Mr V. Mahalingam, Mr S. Panchapakesan, Mrs. Rajalakshmy Sivaramakrishnan, Mr A. Ramakrishna, Mr C. R. Ramakrishnan, Dr P. N. Shankar, Dr R. Srinivasan, Mrs. Vijayalakshmi Shankar, and to Dr (Mrs.) Preethi Shankar of IISc, for their valuable contributions to the project.

The help received from WIPRO in making the hardware for the project is also gratefully acknowledged.

11 July 1988

1. Chapman, D. R., *AIAA J.*, 1979, **17,** 1293.
2. Ramani, S. and Chandrasekar, R., *Sadhana*, 1986, **9,** 121.
3. Seitz, C. L., *Commun. ACM*, 1985, **28,** 22.
4. Squire, J. S. and Palais, S. M., *Proc. Spring Joint Computer Conf.*, 1963, p. 395.
5. Deshpande, M. D. and Vaishnav, R. N., *J. Fluid Mech.*, 1982, **114,** 213.
6. Deshpande, M. D., Sinha, U. N., Sarasamma, V. R., Rajalakshmy Sivaramakrishnan and Ramakrishnan, C. R., *Solution of Navier-Stokes equation on Flowsolver using concurrent parallel algorithm*, NAL PD FM 8720, 1987.
7. Sinha, U. N., Mahalingam, V. and Deshpande, M. D., *Some thoughts on parallel and concurrent computers at NAL*, NAL PD FM 8603, 1986.
8. Sinha, U. N., Sarasamma, V. R., Deshpande, M. D. and Rajalakshmy Sivaramakrishnan, *Solution of transonic small perturbation equation on Flowsolver Mk1 using concurrent/parallel algorithm*, NAL PD FM 8719, 1987.

NEWS

NATIONAL S & T BODY PROPOSED

The Prime Minister's Science Advisory Council has called for integrating science and technology with economic planning and setting up of special organizations for various sectors to enable the country to occupy a 'position of pride and respect' in the world by the turn of the century.

The Council has recommended creation of a National Science and Technology Commission (NSTC) to integrate S & T with planning, national mission for population control, exclusive export zones for increasing exports through S & T efforts, civil aeronautics board for balanced development of aviation industry and an advanced materials research and development board.

In its 26-page document entitled 'An Approach to a Perspective Plan for 2001 AD: Role of Science and Technology' the Council says that without close integration of S & T into planning, 'it is most unlikely that we will achieve the ambitious social and economic goals.'

The document says that the country should develop a sense of urgency and dedication to the concept of 'getting things done' in a given time frame by preparing effective implementation mechanisms. (*Standard India*, Vol. 2, October 1988, p. 288. Published by Bureau of Indian Standards, Manak Bhavan, New Delhi 110 020.)