Defence Science Journal, Vol 46, No. 4, October 1996, pp 269-27(


# Parallelisation & Application of Viscous Codes VASBI & RANS3D on PACE+

Biju Uthup and H.K. Narahari
*Aeronautical Development Agency, Bangalore-560 017*

**ABSTRACT**

The design and analysis codes VASBI and RANS3D have been made operational on the PACE+ parallel computer. The parallel version of VASBI was validated on BPPS computer and ported to PACE+ computer using the BARC message passing library available on PACE+. RANS3D code has been parallelised using PACE message passing library routines after validating the sequential code on IBM RS6000/560 at Aeronautical Development Agency (ADA), Bangalore. A speed up of 18-22 with respect to IBM RS6000/560 of ADA has been obtained for both codes with double precision calculations. The speed up remains fairly constant with increasing grid sizes on 32 nodes of PACE+ for the explicit code VASBI, but decreases for the implicit code RANS3D when communication between nodes is increased. The code VASBI was used to carry out several computations on two different ducts to compare the performance of the ducts for various conditions on 32 node PACE+.

## 1. INTRODUCTION

The development of three-dimensional compressible Reynold's averaged Navier-Stokes Solvers for complex geometries of practical importance has been one of the major difficult areas in CFD. The main reason for this difficulty is that the system of equations demand CPU-intensive computations. Usually numerical solution of these equations involve initialising the flow field by an intelligent initial guess and driving the solution to convergence by advancing the solution in time using Navier-Stokes Solver and applying suitable boundary conditions at each time step. The time taken to reach convergence depends on the initial condition, algorithm, size of the grid and the computing platform used.

One of the major difficulties faced by a Navier-Stokes code developer has been how to reduce CPU time required to obtain solutions for configurations of practical importance. At the algorithm level, various acceleration devices like grid sequencing, multigrid techniques, implicitisation, application of non-reflecting boundary conditions and the use of GMRES can be used to accelerate the convergence. The easiest way to reduce the CPU time is to use faster sequential computers to obtain the solution. However, not all the CFD code developers have access to high speed super computers which have been restricted only to the developed countries. With the advent of parallel computers in India, CFD developers in India can now benefit from supercomputing speeds thus making the development and application of Navier-Stokes codes possible.

This paper describes the parallel implementation of a three-dimensional explicit Reynold's averaged Navier-Stokes code VASBI and an implicit code RANS3D for obtaining turbulent viscous flow through a symmetric bifurcated intake duct and over a combat aircraft on the PACE+ parallel machine. The parallelisation effort was carried out since the sequential machine IBM RS6000/560 at Aeronautical Development Agency was not fast enough for validation and application of these codes.

## 2. PART A: PARALLEL IMPLEMENTATION OF VASBI

In this part parallel implementation and application of VASBI on PACE+ system is described.

Received 04 March 1996

### 2.1 Description of Code VASBI

VASBI is a three-dimensional explicit finite volume code for obtaining compressible turbulent viscous flows through symmetric bifurcated air-intake ducts, which solves three-dimensional unsteady Reynold's averaged Navier-Stokes equations. The code is based on MacCormack's predictor corrector scheme with fourth order smoothing to damp out spurious numerical oscillations. The details of the code and formulation are given in the literature[1,2].

A multiblock approach has been adopted to discretise the flow field with the grid having an inner block and an outer block. Details of the grid generation are given in the literature[3]. Derivatives at cell centroids of the finite volumes are calculated by method of least squares and these are used to obtain viscous fluxes at the finite volume face. The turbulence model used is the Baldwin-Lomax model. To accelerate convergence and to prevent reflection of waves at the outflow boundary, a non reflecting boundary condition has been used.

Even though the code has been developed for air-intake ducts of aircraft, it can also be used to compute compressible viscous flows through pipes or ducts of various shapes.

### 2.2 Structure of Sequential Code

The computational domain of the three-dimensional duct is schematically given in Fig. 1. It consists of two blocks, an inner block having the dimension IMAX1 * JMAX1 * KMAX and an outer block having the dimension IMAX2 * JMAX2 * KMAX.

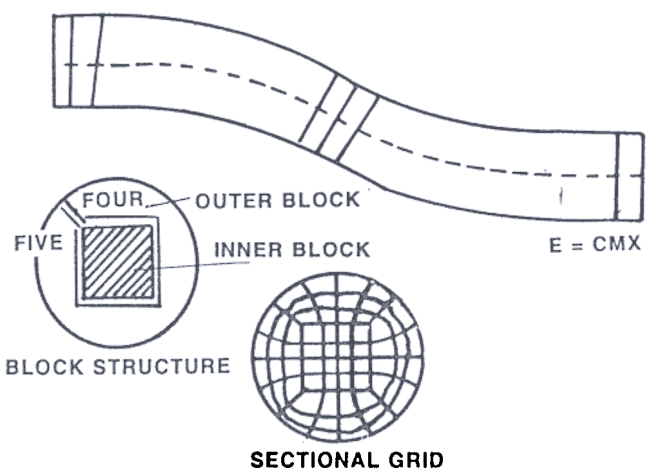


Figure 1.

As a first step, the grid file that has been generated by the grid generation programme is read and normalised. Then the flow field is initialised by an intelligent guess. This initial guess is advanced in time by updating the solution by predictor and corrector steps of MacCormack's Scheme, with the application of wall, inflow and outflow boundary conditions. This process of updating the solution continues till steady state is obtained.

### 2.3 Parallel Algorithm for VASBI

In the above sequential procedure most of the CPU time is spent in updating the solution. In general, it may take about 5000-15000 iterations for an explicit scheme to converge on a fine grid. In addition, the memory requirements for a fine grid calculation become so large that they cannot be handled by the single processor of a sequential machine. Parallelisation helps in overcoming the above CPU and memory constraints.

The parallelisation of VASBI has been carried out by domain decomposition. The computational domain is decomposed into $N$ subdomains in stream-wise direction (i.e. in the $K$ direction Fig. 2). As seen in the figure, there is an overlap region which is common to both neighbouring processors. This region is required since update of the solution at any section $K$ depends on the solution at sections $K$+2, $K$+1, $K$, $K$-1, $K$-2 of

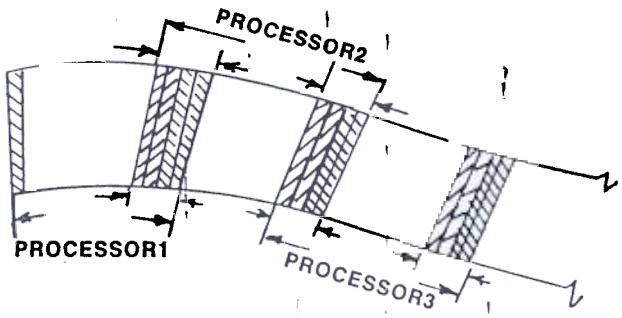


Figure 2.

previous time level. The above domain decomposition is chosen to minimise surface area between cuts which will in turn minimise communication between slaves.

The parallel algorithm for VASBI consists of two programmes, a master programme and a slave programme. The master programme is responsible for all inputs and outputs, sub-dividing the computational domain, downloading the data to all the slave processors and also to synchronise all activities of the slave processors. The slave programme consists of all

the CPU intensive instructions and once compiled resides in all the slaves and operates on the different data that reside in the slaves.

After updating the solution in each of the slaves, the data at the boundary of each slave are sent to the corresponding locations in the neighbouring slave, before the next updating is performed. In the case of VASBI, data from two planes, in the $K$ direction of the left boundary and two planes of the right boundary are passed on to the corresponding locations in the neighbouring slaves.

At the end of each iteration the maximum residue of each node and the location is passed on to the master to obtain overall maximum residue and their corresponding location. After a prescribed number of iterations the complete data are passed on to the master and stored into the disc so that solution process can be restarted from this point.

### 2.4 Porting Parallel Code to PACE+ System

The parallel implementation of VASBI on PACE+ was carried out with a minimum effort from the user, because of the availability of BARC message passing library routines on PACE+. The PACE+ system provides the user with the complete set of BARC message passing routines, which internally call PACE+ message passing routines. With this facility on the PACE+, a code that has been parallelised on the BPPS system can be run on PACE+ without any changes in the parallel code. This has eliminated the need to rewrite the parallel code to make it operational on PACE+.

## 3. PART B: EXPERIENCES OF DEVELOPING A PARALLEL VERSION OF RANS3D

RANS3D is a computer code developed in-house at ADA for obtaining solutions to viscous flow problems encountered by a combat aircraft. The set of partial differential equations describe compressible, turbulent and complex three-dimensional flow field. The choice of any specific solution methodology is a difficult task and is to be governed by the following factors:

(a) Critical project schedules where data has to be made available to the designers within very short times,

(b) Non-availability of high speed sequential machines capable of delivering 300 MFLOPs in the country due to western export restrictions, and

(c) Easy availability of moderate speed microcomputer CPU and memory chips.

The first factor forces the developer to opt for well-tested schemes used by many others, even though they may not be the most optimum, as there is very little time and scope for research. The second factor narrows down the choice further to those schemes that are amenable to be ported on parallel machines. The third factor implies that India has no choice but to go in for the development of parallel machines with a large number of nodes of moderate speeds but with large available memory. This aspect can be taken advantage of by the developer in choosing schemes which while being memory-intensive are amenable to parallel decomposition. The present work describes one such effort at ADA.

### 3.1 Physics of the Problem

Compressible, turbulent fluid flow in this study can be described by what are known as Navier-Stokes equations involving five unknowns, viz., three velocities in three directions, density and energy content of the flow. The ensuing set of partial differential equations has a strong hyperbolic character meaning that the direction of flow is an important factor. While converting the differential equations to appropriate difference equations, directional bias has to be correctly built-in.

### 3.2 Mathematics of the Problem

The 3D N-S equations can be cast into non-dimensional, vector form in generalised body fitted coordinates[4,5] as follows:

$$\frac{\delta Q}{\delta t}+\frac{\delta E_i}{\delta \xi}+\frac{\delta F_i}{\delta \eta}+\frac{\delta G_i}{\delta \zeta}=\mathrm{Re}^{-1}\left\{\frac{\delta E_v}{\delta \xi}+\frac{\delta F_v}{\delta \eta}+\frac{\delta G_v}{\delta \zeta}\right\} \quad (1)$$

Here, $Q$ is the vector of unknowns $(\rho, e, u, v, w)$

$E_i$, $F_i$, $G_i$ are the inviscid flux vectors

$E_v$, $F_v$, $G_v$ are the viscous flux vectors along with the pressure term

$(\xi, \eta, \zeta)$ are the coordinate directions.

$Re$ is the Reynold's number.

To give an 'upwind' bias to the difference equations the flux vectors are expressed in terms of their Jacobians. Linearising the terms over time level

'$n$' and collecting terms at different time levels we have

$$\left[\frac{I}{\Delta t} + \delta_\xi (\bar{A} + \delta_\eta \bar{B} + \delta_\zeta \bar{C})\right] \Delta Q^n = RHS \qquad (2)$$

Where, $\Delta Q^n = (Q^{n+1} - Q^n)$

Here $A$, $B$, and $C$ are the flux Jacobians $\frac{\delta E}{\delta \xi} = \frac{\delta E}{\delta Q}\frac{\delta Q}{\delta \xi} = A\frac{\delta Q}{\delta \xi}$ and $RHS$ is the collection of all viscous terms and terms at the time level '$n$'. In the above implicit equation $A$, $B$, $C$ are block matrices of size (5×5) and $\Delta Q$ is a block matrix of size (5×1). The implicit formulation is chosen because in general they converge faster and to lower residual values, than explicit methods, the drawback being that they are more memory-intensive, however, as noted earlier, the availability of main memory is not such a constraint anymore and hence this approach was chosen.

There are a variety of methodologies available in published literature for solution of Eqn 2. A choice has to be made keeping in view the ultimate target machine, viz., a large parallel computer. The simplest and most robust parallel computation strategy is 'Domain Decomposition'. While parallelisation at algorithmic level may yield better efficiencies, this could not be attempted due to lack of time and non-availability of parallel computer at the initial stage. The strategy therefore was to develop the code on the fastest sequential machine available and port it to the parallel computer as and when available with minimum loss of time and hence the choice of 'Domain Decomposition'. The choice then narrows down to what are referred to as two factor schemes, viz., appropriate factorisation in two directions coupled with relaxation in the third. This leads to the problem of solving block tridiagonal equations in ($\xi$ & $\eta$) directions (cross flow) and relaxation in the $\zeta$ direction (along the flow). This is referred to as successive planar Gauss Siedal method (SPGS) in literature. Here the computational domain is decomposed into several planes ( $\xi$ & $\eta$ ) along $K$-direction. In each ( $\xi$ & $\eta$ ) plane block tridiagonal matrix solution is obtained, once each in ($\xi$ & $\eta$) direction. The solution is then marched forward and backward along $J$ direction till the residuals fall by about three decades.

### 3.3 Discretisation of the Problem

The entire ($\xi$, $\eta$, $\zeta$) computational domain is discretised into IMAX, JMAX, KMAX) points, respectively. Here $I$ index varies radially outward starting from surface and ending at outer boundary . Index $J$ varies from top to bottom azimuthal direction and $K$ varies from inflow to outflow (KMAX). The grid is clustered at the surface to resolve the flow features. The number of grid points and the type of clustering depends to a great extent on the flow domain of interest. The standard SPGS algorithm starts the computation from $K$=2 plane and proceeds till the last but one plane, viz, KMAX-1. The first or the inflow boundary values are assumed to be known at the start of computation. The finite difference equations turn out such that when any $K$th plane is being computed, values for ($K$-1)th and $K$th plane are needed. As the algorithm proceeds along $K$ direction the latest available values for ( $K$-1)th plane are used. Values for down stream plane are of course at the old time level. When the direction of solution is reversed, this too is reversed. The important point to note here is that the latest available values for the upstream plane are used at each time level. This point poses a few restrictions on the domain decomposition strategy.

### 3.4 Domain Decomposition

Assuming that our parallel computer has '$n$' number of processors or nodes, the domain can be divided into ( KMAX/$n$) number of blocks. Each block of planes can then be downloaded into the nodes. However, because of the coupling nature of the implicit algorithm, two buffer planes are required, one for upstream edge and the other for downstream edge of each block of planes. Therefore, each node now will be solving for (KMAX/$n$+2) number of planes. After every iteration, each node has to communicate with its neighbouring node to send and receive data for the buffer planes. The major problem here is that each block starts with upstream values at the previous time level instead of the latest available value as in standard SPGS algorithm. This deviation can destabilise the overall algorithm if the domain is split into too many blocks, even otherwise the residuals will be marginally different, and slightly higher number of iterations are required as compared to the sequential algorithm.

### 3.5 Computational Details

Four grid sizes were chosen for the bench marking of RANS3D code. Table 1 gives the sizes of these grids. GRID2 and GRID3 are obtained by doubling the size of GRID1 in $K$ direction. This will ensure that the size of the data required for communication between nodes will be the same for all these grids, since domain decomposition has been carried out in the $K$ direction. In GRID4, IMAX and JMAX are larger than the previous grids. However, the total number of grid points are almost same as GRID3. All computations for above grids are done with double precision arithmetic.

Table 1

| | GRID 1 | GRID 2 | GRID 3 | GRID 4 |
|---|---|---|---|---|
| IMAX | 26 | 26 | 26 | 51 |
| JMAX | 60 | 60 | 60 | 119 |
| KMAX | 96 | 192 | 384 | 96 |

## 4. RESULTS AND DISCUSSIONS

The first check that was carried out after porting of the parallel VASBI code on PACE+ was to compare the results obtained on PACE+ with that on BPPS system for the same grid. Figure 3 shows residue plot for a duct on crude grid obtained on PACE+ and BPPS systems which are seen to be identical at graphical level. Figure 4 shows that $C_p$ distribution along the duct obtained from both the systems is also identical.

Having established accuracy of the results on PACE+, bench marking of PACE+ system was carried out at the Advanced Numerical Research & Analysis

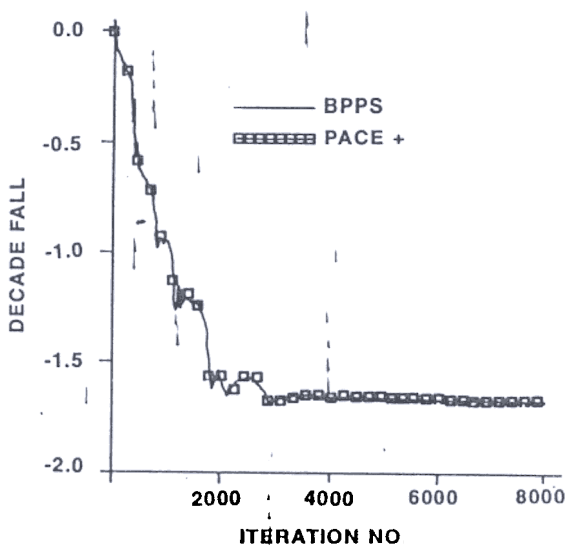


Figure 3.


Figure 4.

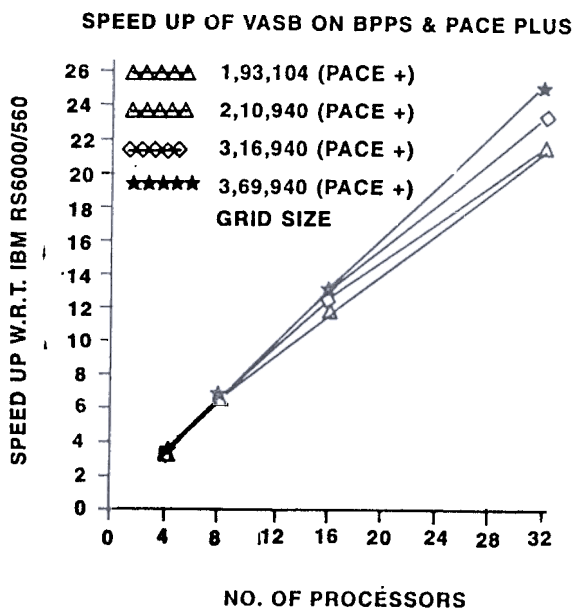


Figure 5.

Group (ANURAG), Hyderabad. Figure 5 shows the speed up of VASBI on PACE+ with respect to IBM RS6000/560 for increasing number of nodes. A maximum speed up of 25.5 times has been obtained with respect to IBM RS6000/560 system, for a grid having a total of 3.6 lakh grid points on 32 nodes of PACE+. Figure 6 shows the comparison of speed up on BPPS and PACE+ systems for the same grid. It is seen that PACE+ is about 3.2 times faster than BPPS. Figure 7 shows the parallel efficiency of PACE+ and BPPS using the parallel code VASBI for the same grid.

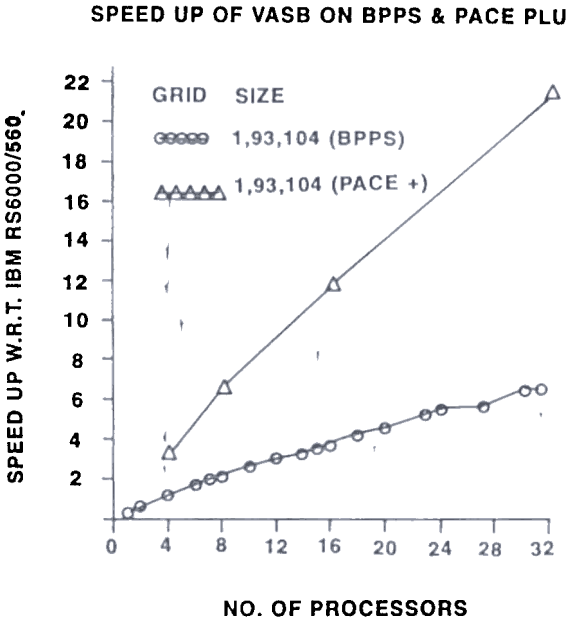


Figure 6

As seen from this figure, BPPS is more efficient than PACE+ even though the speed up on PACE+ is more than that of BPPS. The above computations on PACE+ was a single precision calculation while the calculation on IBM RS6000/560 and BPPS systems were double precision calculations. Moreover the size of the grid for the PACE+ calculations was increased without increasing the surface area of cuts between domains allocated for each node.

Subsequent to this, the PACE+ computer was installed at ADA and bench marking of VASBI was carried out again. This time double precision calculation was carried out on PACE+ and the grid size was increased by having no restrictions on the surface area between cuts. Figure 8 shows speed up obtained for varying grid sizes on 32 nodes of the PACE+ system. Also shown in the figure is the speed up obtained on the BPPS system based on the i860-XP chips (This chip is

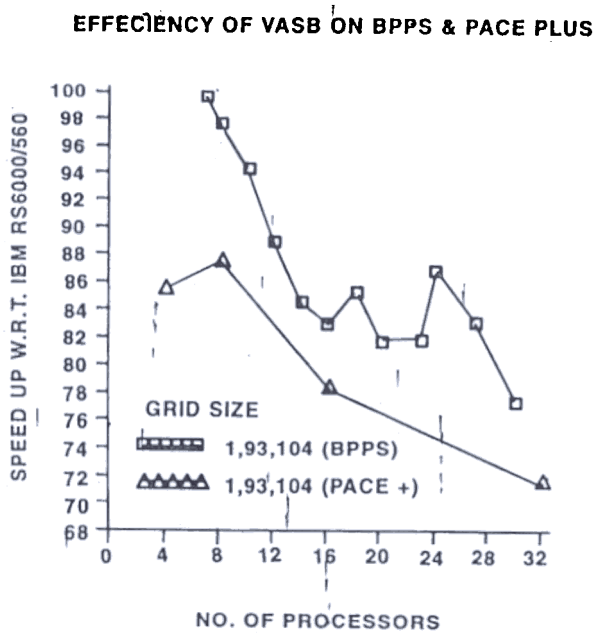


Figure 7.

faster than the earlier 32 node system based on i860 chip). As can be seen from this figure, there is a marginal decrease in the speed up on PACE+ as the grid size increases, while there is an increase of speed up with increasing grid size on the BPPS and saturates after about 1 lakh grid points. The maximum grid size of about 1.9 million grid points can be run on the 32-node PACE+ as compared to a maximum of 4.5 lakh grid points on the BPPS.

The results of the bench marking of PACE+ using RANS3D code are shown in Figs 9 & 10. Figure 9

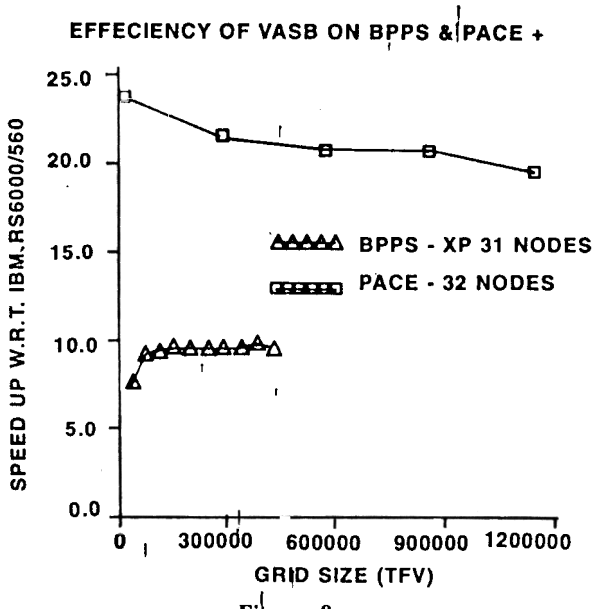


Figure 8.

shows the CPU time taken for two iterations (averaged over 12 iterations) as a function of number of nodes. Figure10 shows the speed up obtained as compared to the sequential machine IBM RS6000/560. The results show that the speed up increases with increasing grid size and number of processors, if communication is kept constant. A maximum speed up of 21 has been obtained with respect to IBM RS6000/560. It is also seen from this figure that the speed up drops drastically if communication is increased by increasing the grid size in *I* and *J* directions as in GRID4. The reason for this drop can either be due to insufficient memory in the reflective memory, responsible for inter-cluster communication or due to slow VME bus which is responsible for the communication between the nodes in a cluster. Since the speed up is linear for GRID4 between clusters it is unlikely that the drop in speed up for GRID4 is due to bottleneck in inter-cluster communication. More interaction with the designers of PACE+ is required to confirm this.

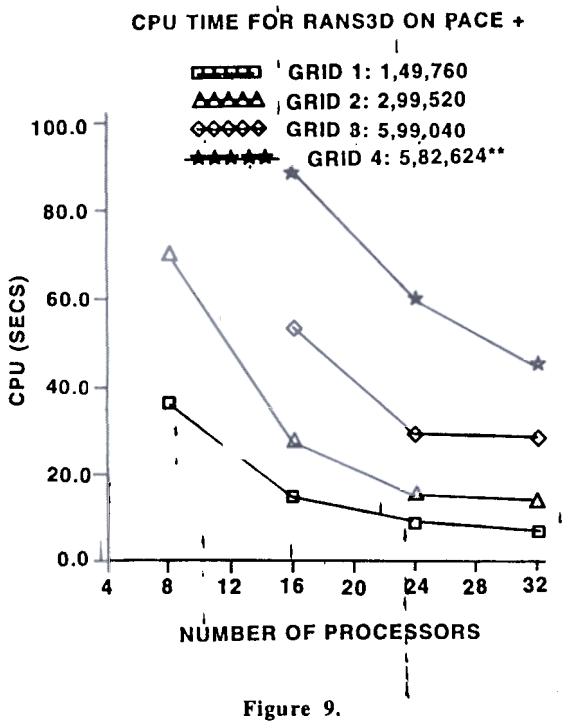


Figure 9.

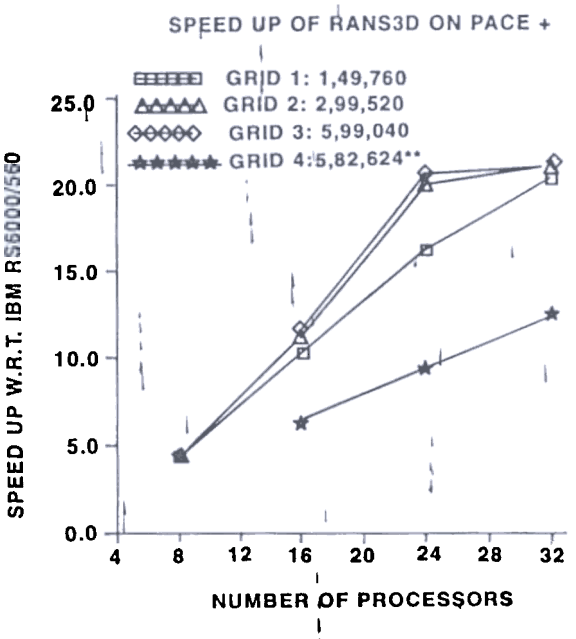


Figure 10.

## 5. CONCLUSIONS

The design and analysis codes VASBI and RANS3D have been made operational on PACE+ parallel computer. The parallel implementation of VASBI has been made easy with the availability of the BARC message passing library routines available on PACE+. Speed up of 18-22 were obtained on both the codes for double precision calculations. It was found that the BARC parallel computer was more efficient than PACE+ for a given grid size even though the speed up on PACE+ was about 3.2 times that on the BPPS system. It indicates the possibility of design modifications of PACE+ for increasing the speed up on PACE+ even further by improving the communication between nodes. From the results obtained from the implicit code RANS3D, it is seen that a maximum speed up of 21 has been obtained with respect to IBM RS6000/560. It has also been observed that communication bottleneck occurs for the implicit code if the grid size is increased in *I* and *J* directions. More interaction with the designers of PACE+ is required to identify the cause of the drastic drop in speed up for RANS3D

## ACKNOWLEDGMENTS

The authors would like to thank Shri N.C. Sampath of Ashok Leyland Information Technology, Bangalore, for the assistance rendered in parallelisation of the code. The assistance given by K. Rajesh of BARC to parallelise VASBI and K. Mahesh of ANURAG for porting the parallel code to PACE+ is acknowledged.

## REFERENCES


1. Biju Uthup. ADA Interim Report, ADA/'PD/CFD/ TR/ 025, 1993.
2. Biju, Uthup.; Deshpande, S.M.; Marathe, A.G. & Rajesh, K. Proceedings of the second European CFD conference, ECCOMAS 94, Stutgart Germany September 1994, Wiley International.
3. Biju Uthup. Grid generation for viscous flow calculation for LCA air-intake duct—Part II, ADA/TD/CFD/TR/019, October, 1992.
4. Narahari, H.K. Validation of compressible thin laser NS code, ADA/TD/CFD/TR/033, December 1993.
5. Narahari, H.K. RANS3D theoretical manual. ADA/TD/CFD/TR/051.
6. Sampath, N.C. Parallelisation of thin layer NS Equations solver. MTech Report. Dept of computer science, Pondicherry University, December 1994.


## Contributor

**Mr Biju Uthup** obtained his BTech in Aeronautical Engineering from Indian Institute of Technology, Kharagpur, in 1981 and ME in Aerospace Engineering from Indian Institute of Science (IISe), Bangalore, in 1993. He then worked on several DRDO sponsored projects at IISc before joining the Computational Fluid Dynamics (CFD) Group at the Aeronautical Development Agency, Bangalore in 1987. He has published several papers at national and international levels in the areas of applied CFD and parallel computing.