COMPUTATIONAL STRATEGIES FOR THREE-DIMENSIONAL FLOW SIMULATIONS ON DISTRIBUTED COMPUTER SYSTEMS

Progress Report for the Period

August 15, 1993 to February 15, 1994

Attn: Dr. Terry L. Holst
NASA Ames Research Center
Moffet Field, CA 94035-1000

Prepared by

R. A. Weed
Graduate Research Assistant
and
L.N. Sankar
Professor, School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332

March 1994

SUMMARY

The following tasks were accomplished during the reporting period of August, 15, 1993 to February 15, 1994:

1.) A copy of the TEAM code was obtained from the Air Force and modified to function as a distributed parallel flow solver using the PVM software interface.

2.) Validation cases using the MBB body of revolution geometry and the ONERA M6 wing geometry were successfully executed on a homogeneous system of Hewlett Packard workstations

3.) Performance test were conducted using the MBB, ONERA M6, and Lockheed WING C geometry on both Hewlett-Packard and Digital Equipment ALPHA workstations.

4.) Initial modifications were made to replace the TEAM explicit flow solver with an implicit scheme

The results from these tasks are presented in the enclosed draft thesis proposal and an abstract of a paper submitted to the AIAA Fluid and Plasma Conference to be held in June. The paper was accepted for the conference.

Work to be Performed during the next Reporting Period

During the next reporting period Mr. Weed will be working on improving the performance of the PVM-TEAM code. Implementation of a suitable load balancing algorithm and the application of the PVM-TEAM code for unsteady flow analyses will be the primary areas of research during this period.

COMPUTATIONAL STRATEGIES FOR THREE-DIMENSIONAL UNSTEADY FLOW SIMULATIONS ON DISTRIBUTED COMPUTER SYSTEMS

A Thesis Proposal
Presented to

The Faculty of the School of Aerospace Engineering

by

Richard Allen Weed

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in the School of Aerospace Engineering

Georgia Institute of Technology

March 1994

ABSTRACT

An increasing amount of research activity in Computational Fluid Dynamics has been devoted to the development of efficient algorithms for parallel computing systems. The increasing performance performance to price ratio of engineering workstations has led to research to development procedures for implementing a parallel computing system composed of distributed workstations. This thesis proposal outlines an ongoing research program to develop efficient strategies for performing three-dimensional flow analysis on distributed computing systems.

The PVM parallel programming interface was used to modify an existing three-dimensional flow solver, the TEAM code developed by Lockheed for the Air Force, to function as a parallel flow solver on clusters of workstations. Steady flow solutions were generated for three diffent wing and body geometries to validate the code and evaluate code performance. The proposed research will extend the parallel code development to determine the most efficient strategies for unsteady flow simulations.

TABLE OF CONTENTS

1. INTRODUCTION

During the past decade, an increasing amount of research activity in Computational Fluid Dynamics has been devoted to harnessing the power of parallel computing architectures to improve the total throughput of CFD codes. This effort was prompted by the fact that existing vector supercomputers such as the CRAY are approaching the theoretical limit in processing speed obtainable by a single processor. Therefore, research has centered on developing new algorithms that take advantage of the task level parallelism inherent in the numerical solution of the Euler or Navier-Stokes equations and to the problems associated with porting existing algorithms to different parallel architectures. Most of the research effort has been targeted at massively parallel architectures composed of large numbers of simple processors such as the Connection Machine CM2 and distributed multi-processor systems such as the CRAY Y/MP and INTEL iPSC/860 which are composed of a relatively small number of powerful vector class processors. However, the difficulties in porting existing codes to these systems along with their huge costs and limited availability have prevented their wide spread use by the aerospace industry.

Recently, the increasing performance to price ratio and availability of engineering workstations have led some researchers to explore the feasibility of linking clusters of workstations together to form distributed parallel computing systems. This interest was also prompted by the fact that most engineering workstations are idle during off hours and represent an underutilized computing resource of enormous potential. In addition, the recent development of standardized application programming interfaces such as the Parallel Virtual Machine[1,2,3] (PVM) system have greatly reduced the effort required to link

together workstations as a distributed parallel system. The PVM system provides libraries of user callable procedures in either the FORTRAN or C languages that provide efficient functions for passing data and messages between clusters of heterogeneous workstations. The details of data conversion and communication protocols between systems are hidden from the user. This makes it possible to use a variety of different types of computers in a parallel system. The code modification requirements for implementing a CFD algorithm on such a parallel system are also greatly reduced. Therefore, parallel solution strategies such as domain decomposition which map blocks of grids to individual processors can be implemented on a distributed workstation system with only a modest amount of code modification.

The recent work of Smith et. al.[4,5] has demonstrated the effectiveness of a PVM based parallel CFD algorithm using domain decomposition. Their work also points out some of problems associated with a workstation based distributed system. An effective algorithm for balancing the computational load among processors and minimizing processor idle time is crucial to obtaining speedup of the parallel code over a serial code running on a single processor. The amount of data communications between processors must be kept to a minimum in order to reduce the idle time incurred by slow communications hardware. Therefore, an efficient procedure for updating the boundary data common to grid systems on separate processors that does not impact convergence is also essential. These problems must be resolved before a workstation based distributed system can be competitive in a real world engineering environment with a dedicated supercomputer.

The research outlined in this proposal will address the issues involved in porting, fine-tuning and improving the algorithms of existing flow solvers for optimal performance on a distributed parallel computer system. The research will emphasis

the development of parallel solution strategies for unsteady flow simulation.

2. APPROACH

The problems associated with implementing a distributed parallel CFD code will be investigated by using the PVM software to modify an existing serial flow solver to function as a distributed parallel flow solver. The flow solver selected for this research is the Transonic Euler/Navier-Stokes Analysis Method (TEAM) code[6] developed by the Lockheed Aeronautical Systems Company (LASC) for the U. S. Air Force. The following sections present detailed descriptions of the TEAM code, the PVM system and the modifications made to TEAM to implement a distributed parallel system.

2.1 THE TEAM CODE

The TEAM code can solve either the Euler or Navier-Stokes equations using a finite volume formulation of the governing equations and a multi-stage Runge-Kutta time-stepping algorithm. The code can accommodate both single and multiple zone grid topologies. Therefore, complex geometries such as a complete aircraft can be modeled. TEAM has been applied to a wide variety of configurations and flow regimes.

2.1.1 Finite Volume Discretization

The finite volume formulation of the Euler or Navier-Stokes equations starts with writing the integral form of the governing equations for an arbitrary control volume Ω, enclosed by a surface of area A in Cartesian coordinates as:

$$\frac{\partial}{\partial t}\int_{\Omega} Q \, d\Omega + \int_{A} \vec{F} \bullet \hat{n} \, dA = 0 \qquad (2.1)$$

where Q is the vector of dependent variables $[\rho, \rho u, \rho v, \rho w, \rho E]$, F is the flux vector and $\hat{n}$ is the unit normal vector. The flux vector is defined as $F = F_{inv}$ for Euler flows and $F = F_{inv} - F_{vis}$ for Navier-Stokes flows. F_{inv} contains the inviscid components of the flux vector and F_{vis} contains the viscous contributions to the momentum and energy equations. In the finite volume formulation, the control volume is taken to be an individual cell of the computational grid. If Q is taken to be a mean value defined at the cell center as the cell volume approaches zero, Equation 2.1 can be replaced by the semi-discrete differential form

$$\frac{dQ\Omega}{dt} + \sum_{i=1}^{N} \vec{F}_i \bullet \vec{S}_i = 0 \qquad (2.2)$$

where N is the number of cell faces surrounding the cell center and $\vec{S}_i$ is the outward facing area vector at the centroid of each cell face. The flux vector, $\vec{F}_i$, is evaluated at cell faces using averages of quantities at the adjoining cell centers. The finite volume formulation is applicable to arbitrary grid systems. All metric quantities are defined geometrically from the Cartesian coordinates of the grid nodes. Therefore, a transformation of the governing equations to a local curvilinear coordinate system is not formally required as it is with finite difference discretizations. In addition, the problems associated with defining metric quantities at grid singularities are avoided. The finite volume formulation simplifies the treatment of zonal boundaries for multi-zone grid systems since values on the grid boundaries are not required. A layer of ghost cells surrounding the boundaries can be used to hold the cell center values required from adjoining blocks. Therefore, finite volume schemes

are well suited for multi-zone solutions about complex geometries.

2.1.2 Multi-Stage Solution Algorithm

TEAM uses an m-stage explicit time stepping scheme based on the Runge-Kutta method to integrate Equation 2.2 in time. Numerical dissipation terms, D(Q), are added to enhance stability and ensure correct shock capturing. The default dissipation models in TEAM are based on a blended combination of second and fourth order differences. The second order dissipation is required for flows with shocks to prevent non-physical overshoots in the shock that violate the entropy condition. The fourth order dissipation is required to damp short wavelength error components that arise from odd-even decoupling and aliasing phenomenom where the short wavelengths interact to form destabilizing long waves. These models are described in Appendix A. Equation 2.2 can be rewritten as

$$\frac{dQ}{dt}+R(Q)=0 \tag{2.3}$$

where R is the residual defined as

$$R(Q) = \frac{1}{\Omega} \left(F(Q) - D(Q)\right)$$

The multistage scheme is then used to advance the solution from time level n to n+1. For example, a four stage scheme can be written as:

$$\begin{aligned}
Q^{(0)} &= Q^{n} \\
Q^{(1)} &= Q^{(0)} - \alpha_1 \Delta t \, R^{(0)} \\
Q^{(2)} &= Q^{(0)} - \alpha_2 \Delta t \, R^{(1)} \\
Q^{(3)} &= Q^{(0)} - \alpha_3 \Delta t \, R^{(2)} \\
Q^{(4)} &= Q^{(0)} - \alpha_4 \Delta t \, R^{(3)} \\
Q^{n+1} &= Q^{(4)}
\end{aligned} \tag{2.4}$$

where the α coefficients have values of 1/4, 1/3, 1/2, and 1. The time step, Δt, can be either fixed or spatially varying for steady state calculations. The preceding four stage explicit scheme is stable for CFL numbers of up to $2\sqrt{2}$ for a 1-D model problem. For steady state calculations, TEAM uses two acceleration techniques, Enthalpy Damping and Residual Smoothing, to increase the effective CFL limit of the scheme to 6. Enthalpy damping introduces forcing terms into Equation 2.3 that help maintain constant total enthalpy during the solution process. Residual Smoothing applies a Laplacian like smoothing operator to the residual term in Equation 2.3.

2.2 THE PVM SYSTEM

The Parallel Virtual Machine (PVM) programming interface was developed by Oak Ridge National Laboratories to provide an integrated software framework for the development of distributed parallel computing systems using existing networks of computers that can be either homogenous systems of the same type or heterogeneous systems composed of computers of different types. The PVM system is based on the message passing model[7] of parallel processing which has become the standard model on large multiprocessor distributed memory systems. In the message passing model, communication between processors is performed by passing message packets containing data common to all processors over interprocessor communication channels. In a dedicated multiprocessor system, these channels are usually special purpose high-speed internal hardware systems. In a network environment, these channels are normally slower external hardware systems such as Ethernet. The PVM system emulates a generalized distributed memory multiprocessor in a heterogeneous network environment.

Therefore, a virtual parallel machine can be constructed from any set of machines available on the network.

The PVM system has two primary components. The first is a control process or daemon in UNIX terminology that runs on all systems in the virtual machine. The daemons act as an interface that routes messages to and from individual user applications running on different machines. The daemons provide buffers to hold incoming and outgoing messages and ensure that these message are processed in the correct order. The PVM system starts the daemons from a list of hosts supplied by the user.

The second component of the PVM system is a library of user callable routines for message passing, spawning processes, sequencing tasks, and dynamically reconfiguring the virtual machine. These routines must be linked to the users application. The message passing routines form the core of the programming interface. They allow the user to open message buffers that can be packed by data of varying data type and send the data to individual processes in the virtual machine with unique message tags. The PVM software performs the data conversions required when binary data is passed between machines of different architectures. This simple but complete library allows users to develop parallel applications with minimal modifications to their existing code.

2.3 IMPLEMENTATION OF PVM INTO TEAM (PVMTEAM)

The PVM software interface has been used to modify a version of TEAM obtained from the U. S. Air Force to function as a distributed parallel flow solver. The multi-zone solution capabilities of TEAM allows the code to be modified to use domain decomposition to perform the solution for different grid blocks in parallel. Following Smith and Palas, a Manager/Worker[4,5,7] was

adopted to control the sequencing of tasks and the data communications required between processors.

2.3.1 The Manager/Worker Strategy

The overall performance of a message passing parallel system is dependent on the speed of the communications channels connecting the processors and the number of messages that must be passed between processors. In a multi-zone parallel flow solver, boundary data from adjoining zones must be passed to each processor at the end of an iteration. This can be done by either letting individual processors communicate with each other or by passing the data to a single processor that accumulates the data and passes it to the separate processors as needed. Allowing individual processes to communicate with each other greatly adds to the complexity of data sequencing because of the number of possible data paths when a large number of grids or processors are used. The alternative approach of having individual processors communicate with only a single master process is called the Manager/Worker strategy. This is illustrated in Figure 1. The Manager process controls the sequencing of tasks such as metric calculation, time stepping, etc. and the accumulation and distribution of boundary data required by individual processors. The Worker processes are independent programs that perform the bulk of the computational work. Typically one or more computational grids are mapped to separate processors in the virtual machine. The Manager/Worker strategy greatly simplifies the logic required to implement a parallel flow solver by reducing the number of message paths required for task sequencing and boundary data transfer.

The Manager/Worker strategy was implemented in TEAM by first breaking the baseline code into two separate programs. The

Manager code retains TEAM's memory allocation, data input, and initialization functions. PVM logic was added to control worker task sequencing and boundary data updates. The Worker program consists of the core solution routines from the original code. These include time stepping, residual calculation, etc. PVM logic was included to receive initialization and task sequencing data, and to receive and return boundary data. The logic flow for the baseline TEAM code, the Manager process and the Worker process are presented as pseudo-code in Figure 2. Each send or receive in the Manager and Worker represents a sequence of calls to PVM's message passing routines.

2.3.2 Load Balancing

Load balancing is the process of dividing the computational work among individual processors in a manor that keeps all processors busy and reduces idle time spent waiting for data. Proper load balancing is crucial if the expected speedup of the parallel version of TEAM over the baseline version is to be obtained. No formal load balancing algorithm has been implemented in the current version of PVMTEAM. Load balancing will constitute one of the major areas of investigation in this research.

3.0 PVMTEAM TEST RESULTS

A series of tests were conducted to validate the modifications made to TEAM to implement a distributed parallel flow solver. Solutions were generated using standard CFD test cases for a body of revolution and two wing alone geometries. Initial tests were performed to validate code modifications and identify logic errors. Additional tests were performed to study performance related issues such as the effect on convergence of lagging the update of boundary data for grids on different processors and the effect of load imbalance on total turnaround time. The accuracy of the parallel code was established by comparing computed aerodynamic loads with results from the baseline TEAM code. These tests were performed on homogenous parallel systems composed of Hewlett-Packard PA-RISC workstations and Digital Equipment Alpha Processor workstations.

3.1 CODE VALIDATION TESTS

The initial code validation test consisted of an Euler solution about the MBB body of revolution[8] shown in Figure 3 for a freestream Mach number of 0.8 and an angle of attack of zero degrees. The grid system consisted of two blocks containing 55x21x40 (46200) points and 56x21x40 (47040) points. This system was chosen to provide a close but not exact load balance. Both the baseline TEAM code and the PVMTEAM were run with the same grid system for 500 time steps using spatially varying time stepping and a CFL number of 1. PVMTEAM was run on a virtual machine consisting of three Hewlett-Packard workstations, a model 730 with 48 Mbytes of memory and two model 720 workstations with 36 and 18 Mbytes of memory. For these tests, the Manager and Worker processes ran on different machines. Each grid block was

assigned to separate Worker processes. Figure 4. correlates the average change in density with time computed by TEAM and PVMTEAM. This is a measure of convergence. For this case, PVMTEAM and TEAM had virtually identical convergence histories. The leeward pressure distributions computed by TEAM and PVMTEAM are compared in Figure 5 with experimental data. Both codes produced identical pressure distributions that are in close agreement with the experimental data. These tests validated that the PVM modifications did not effect the accuracy or convergence rate of the code for this case. Turnaround times for the PVMTEAM varied from three to eight hours. A typical turnaround time for the baseline code running on a single machine, the model 730, was three to four hours. This wide range in turnaround performance illustrates one of the problems encountered when a PVM system is run on machines during periods of heavy usage by other processes. They also prompted a recoding of PVMTEAM to eliminate the transfer of the complete solution arrays back to the Manager at end of each time step that was present in the initial code. This led to a substantial reduction in the amount of data being passed back and forth during each iteration.

The next validation case run was for the ONERA M6 wing[9] geometry shown in Figure 6. A standard computational grid consisting of five relatively small grid blocks containing 15028, 3680,7820, 1792, and 1216 points was provided by LASC for these tests. The virtual parallel system of Hewlett Packard workstations used for the MBB body tests was used for the ONERA M6 tests. Euler solutions were generated for a Mach number of 0.84 and an angle of attack of 3.06 degrees.

To maintain a roughly even load balance, grid block 1 was assigned to one Worker process and the remaining four grids were assigned to a second Worker process. Initial solutions were run for 1000 time steps at a CFL number of 6. As shown in Figure 7.,

PVMTEAM and the baseline TEAM code yielded drastically different convergence rates. Initially, this was felt to be due to the lag in updating the ghost boundary data shared by grids on the different processors. In the baseline TEAM code, zonal boundary arrays are updated with the most recent data before the next grid block is processed. In PVMTEAM, only the boundary data for the grids on a single processor are updated with the most recently available data. Grids on other processors must wait for the data to be available from the Manager before the solution can proceed. This results in a lag in some of the boundary data as grid blocks are processed in parallel. The baseline TEAM code was modified to use previous time step data for the ghost boundary conditions instead of the most recently available data. As shown in Figure 8., this produced a convergence rate identical to the PVMTEAM results.

The slow convergence rate of both TEAM and PVMTEAM indicated that the amount of artificial dissipation used in the solution was to low. The initial tests were performed using the standard adaptive dissipation model (SAD) with the second order parameter,VIS2, set to 0.1 and the fouth order parameter, VIS4, set to 1.0. The ONERA M6 runs were repeated using the modified adaptive dissipation model (MAD) with VIS2 set to .5 and VIS4 set to 2.0. The convergence rates for baseline TEAM and PVMTEAM with the increased dissipation are shown in Figure 9. Both codes have virtually identical convergence rates up until 600 time steps. At that point, the PVMTEAM job reached the automatic cutoff condition of a six order of magnitude drop in DR/DT. The baseline code continues for another 150 steps. A third run was made using the SAD dissipation model with VIS2=.5 and VIS4=1.5. The force and moment coefficients computed by both TEAM and PVMTEAM are compared in Table 1. along with the maximum and average values of DR/DT at convergence and the number of supersonic points for the

different dissipation models. The codes are seen to produce virtually identical results at the same levels of convergence when appropriate levels of dissipation are used. These results also illustrate the effect that the numerical dissipation can have on the computed loads. The chordwise pressure distributions at two span stations computed by both versions of TEAM are compared with experimental data in Figure 10. The poor correlation with experimental data in the shock region is a result of the coarse grid used in these tests. However, the computed results are identical. It can be concluded from these results that increasing the level of artificial dissipation appears to alleviate the problems associated with lagging boundary data for steady state solutions.

3.2 PERFORMANCE TESTS

A second series of tests were performed to determine the effect of load balance, processor speed, and system utilization on the turnaround performance of PVMTEAM. The ONERA M6 case was rerun on the HP workstations using different grid and processor combinations during periods of light and heavy utilization by other users. The total turnaround time and percentage of time spent computing by the worker processes are shown in Table 2. for three different cases. In case one, three workers were used with the largest grid on the fastest processor. For case two, the largest grid was placed on the slowest processor. In case three, wo worker processes running on the slowest machines were used during a period of heavy utilization by other usage. The fastest turnaround time (1387 seconds) occured for case 1 and the slowest turnaround time (3750 seconds) occured for case 3. The wide range of turnaround time and percentage of time spent computing

illustrate the dramatic affect that processor speed, system utilization and load balance have on performance.

The MBB and ONERA M6 cases were rerun on a system of identical DEC ALPHA workstations. The turnaround times were compared with the time required by the baseline TEAM code running on one processor. Both cases were run using two worker processes with the Manager running on a third machine. The workers were run on different machines during periods of light and heavy usage. The performance for the fastest and slowest turnaround times are compared in Table 3. The baseline code was run on a single machine that was idle when the run was started. For the MBB case, the fastest turnaround time was 2560 seconds and the slowest time was 4224 seconds. The turnaround times for the ONERA M6 case ranged from 1152 seconds to 1280 seconds. The differences in the time spent computing are due to the overhead required for initializing and passing messages and the idle time by the operating system when the machine is shared with other users.

A third set of performance tests have been conducted using a large 7 block grid system generated by LASC for the Lockheed Wing C^{10} geometry shown in Figure 11. This case consists of 179309 total grid points. Euler solutions were generated on both the HP and DEC systems for a Mach number of 0.89 and an angle of attack of 5 degrees. The HP systems consisted of the three machines used previously along with a faster model 735 system. The Manager and one Worker process ran on the 735 system. The load balance in terms of total grid points per processor was as follows: 29,522 points and 38,012 points on the Model 720 systems, 54,080 points on the Model 730, and 58,695 on the model 735. The DEC system consisted of four machines with three worker processes. The load balance on the workers was 66,534, 54,080, and 58,695 points. The elapsed time for the baseline and PVM codes, the final value of DR/DT, lift coefficient (CL) and drag coefficient (CD) are

compared in Table 4. The codes produced almost identical results at the same levels of convergence. The poor turnaround performance of the PVM code on both the HP and the DEC systems is being investigated. The drastic reduction in performance on the HP system is most likely due to running both a Manager and a Worker process on the same system. The large memory of both the Manager and Worker processes for this case can lead to excessive system overhead due to swapping. Heavy system usage has prevented a more complete analysis of the performance reduction on the DEC system.

The performance tests revealed the need for an effective load balancing procedure. They also point out some of the problems encountered when the machines that make up the virtual machine shared with other users.

4.0 THE PROPOSED RESEARCH

The proposed research will extend the effort to date to examine the problems associated with porting and running large CFD codes on a distributed parallel system. Emphasis shall be given to evaluating and improving load balancing procedures and the problems associated with using a PVM based parallel system for unsteady flow analysis.

4.1 LOAD BALANCING PROCEDURES

Load balancing algorithms can be static in which the balance is fixed at the start of the job or dynamic where the balance is varied throughout the course of the run. Two static procedures will be evaluated. The first procedure is the heuristic algorithm given in Reference 4. This procedure assigns grids to processors based on the excess capacity of each machine. The excess capacity is a function of the processor speed and the total number of grid points on each processor. Grids are first sorted by size and then assigned to processors based on which processor has the most excess capacity. The load balance is then optimized to improve the predicted run time.

The second procedure to be evaluated is the Pool of Tasks procedure proposed by Johnson[11]. This procedure is implemented by first creating a queue of idle processors that are ordered by relative processor speed with the most powerful processor at the top. A work queue is then formed from the available grids by ordering the grids by the relative computing time with the longest running entry at the top of the queue. The top work queue entries are then assigned to the top idle queue entries until all the processors are busy or there are no more work queue entries. The manager waits until a worker signals it is idle and then

assigns the next work queue entry to the fastest available processor. This process is repeated during the first iteration until all the tasks are complete. The same distribution is used for subsequent iterations.

Both of these algorithms will be evaluated to determine the most effective approach for use with PVMTEAM. Modifications will be made to allow more than one grid block per processor.

4.2 UNSTEADY FLOW CALCULATIONS

Unsteady analysis will be performed using the F5 wing[12] geometry. This geometry was chosen because of the availabilty of both experimental and computational test results. The unsteady flow analysis will focus on two objectives. The first objective is to evaluate different solution algorithms to determine the most appropriate procedure for unsteady flow calculations using PVM. Modifications will be made to both the baseline TEAM and PVMTEAM to replace the explicit flow solver with an implicit solver to allow larger CFL numbers to be used for the unsteady flow simulation. The implicit solver scheme chosen for this effort is the LU-SGS algorithm described in Appendix B. This algorithm has been used for a wide range of flow simulations.

The second objective is to determine the most appropriate procedure for updating zonal boundary data. Unlike steady flow simulations, the time levels of boundary data must be consistent in order to maintain time accuracy. Therefore, the zonal update procedure in the current version of PVMTEAM which allows some boundary data to be lagged in time must be reevaluated. The effect of the boundary update procedure on the convergence and accuracy of both the explicit and implicit codes will be determined. Finally, the effect of the zonal boundary update procedure on total code performance will be investigated.

APPENDIX A.

TEAM ARTIFICIAL DISSIPATION MODELS

This section describes the two standard dissipation models used in TEAM[6]. The first model is the standard adaptive dissipation model (SAD). The modified adaptive dissipation model is a modified form of the SAD model that is less dissipative than the SAD formulation and normally is used for viscous simulations. In the following, the subscripts I,J and K refer to cell center values and subscripts I+1/2 etc refer to values on cell faces.

Standard Adaptive Dissipation (SAD): Both the SAD and MAD dissipation schemes use blending of first and third differences in each coordinate direction. The dissipative flux for an interior cell face whose surface normal is directed in the along I can be written as:

$$d_{I+1/2,J,K} = \varepsilon_2 e_{I+1/2,J,K} - \varepsilon_4 (e_{I+3/2,J,K} - 2e_{I+1/2,J,K} + e_{I-1/2,J,K}) \qquad (A.1)$$

where

$$e_{I+1/2,J,K} = \alpha (Q_{I+1,J,K} - Q_{I,J,K})$$

$$\alpha = \tfrac{1}{2}(\Lambda_{I,J,K} + \Lambda_{I+1,J,K})$$

$$\Lambda_{I,J,K} = \lambda^I_{I,J,K} + \lambda^J_{I,J,K} + \lambda^K_{I,J,K}$$

The spectral radii of the flux-Jacobian matrices ($\lambda^I, \lambda^J, \lambda^K$) at the cell centers are evaluated for each coordinate direction using metric data averaged from the surrounding cell faces. The coefficients of the first and third difference terms in Equation A.1 are defined to ensure that proper shock capturing. A switch

function is defined by taking the normalized second difference of pressure, P:

$$\nu_{I,J,K} = \left| \frac{P_{I+1,J,K} - 2P_{I,J,K} + P_{I-1,J,K}}{P_{I+1,J,K} + 2P_{I,J,K} + P_{I-1,J,K}} \right|$$

and then defining

$$\bar{\nu}_{I+1/2,J,K} = \max(\nu_{I+2,J,K}, \nu_{I+1,J,K}, \nu_{I,J,K}, \nu_{I-1,J,K})$$

The coefficients ,ε_2 and ε_4, are then defined as:

$$\varepsilon_2 = \kappa_0 \bar{\nu}_{I+1/2,J,K}$$

$$\varepsilon_4 = \max(0, \kappa_1 - \varepsilon_2)$$

The user specified coefficients, VIS2 and VIS4, determine the values of κ_0 and κ_1. The input value of VIS4 is divided by 64 inside TEAM. The pressure switch that scales VIS2 produces a large amount of dissipation near shock waves and stagnation points. The coefficient of the third difference terms is set to zero in regions where the value of the third difference coefficient is less than the first difference coefficient. The dissipation contribution at I,J, or K is the difference in the dissipation d at the surrounding cell faces.

<u>Modified Adaptive Dissipation:</u> The primary difference in the MAD and SAD schemes is the definition of the spectal radius factor used to scale the first difference terms. For the MAD scheme, the scaling factor α is redefined as:

$$\alpha = \tfrac{1}{2}\left[\lambda^I_{I,J,K} + \lambda^I_{I+1,J,K}\right]$$

In addition, ε_2 is redefined to restrict its maximum value:

$$\varepsilon_2 = \min\ (1/2, \kappa_0 \bar{\nu}_{I+1/2,J,K})$$

These modifications act to limit the amount of artificial dissipation added in each coordinate direction.

APPENDIX B.

THE LU-SGS IMPLICIT SCHEME

Implicit formulations of equation 2.2 are obtained when the flux vector at time level N+1 is replace by the linear approximation $F^{N+1} = F^N + \frac{\partial F}{\partial Q}^N \Delta Q$. This leads to a linear system of equations of the form $M\Delta Q = -\Delta t R(Q^N)$. The matrix operator, M, is a full matrix that is expensive to solve by direct inversion. The LU-SGS implicit algorithm[13,14,15] uses the symmetric Gauss-Seidel relaxation procedure and first order upwind differencing of the flux Jacobians, $\frac{\partial F}{\partial Q}$, to factor matrix M into three matrix factors composed of the lower, upper and diagonal block components of M. The solver can be written as:

$$(LD^{-1}U)\Delta Q = -\Delta t R \tag{B.1}$$

where the operators L, D, and U are defined at a node i,j,k as

$$L = I + \Delta t(\nabla_\xi A^+ + \nabla_\eta B^+ + \nabla_\varphi C^+ - A^- - B^- - C^-)_{i,j,k} \tag{B.2}$$

$$D = I + \Delta t(A^+ - A^- + B^+ - B^- + C^+ - C^-)_{i,j,k}$$

$$U = I + \Delta t(\Delta_\xi A^- + \Delta_\eta B^- + \Delta_\varphi C^- + A^+ + B^+ + C^+)_{i,j,k}$$

The positive and negative matrices, A, B, and C are the upwind approximations to the Jacobians of the flux vector F, ΔQ is the update to the solution vector Q, R is the residual and Δt is a time step. The Jacobian matrices can be approximated as:

$$A^+ = \tfrac{1}{2}(A + \beta|\lambda|S) \quad , \quad A^- = \tfrac{1}{2}(A - \beta|\lambda|S) \tag{B.3}$$

where $\beta > 1$. and $|\lambda|$ is an approximation of the spectral radius of A, B, or C. $|\lambda|$ can be taken to be $|\vec{U} \bullet \vec{n}| + c$ where U is the velocity vector, n is a unit vector normal to a cell face, S is the area of the cell face, and c is the local speed of sound. The Jacobian matrices are evaluated using cell center flow variables and metric quantities at cell faces. For instance, $A_{i,j,k} = A(Q_{i+1,j,k}, S_{i+1/2,j,k})$. With the approximate Jacobians defined by A.3, the operators L,D, and U become:

$$L = D_{i,j,k} - \Delta t(A^{+}_{i-1,j,k} + B^{+}_{i,j-1,k} + C^{+}_{i,j,k-1}) \qquad (B.4)$$

$$D = I + \beta\Delta t(|\lambda|^{i}S^{i} + |\lambda|^{j}S^{j} + |\lambda|^{k}S^{k})_{i,j,k}$$

$$U = D_{i,j,k} + \Delta t(A^{-}_{i+1,j,k} + B^{-}_{i,j+1,k} + C^{-}_{i,j,k+1})$$

The solution for ΔQ can then be written as two sweeps:

$$L\Delta Q^{*} = -\Delta t R \qquad (B.5)$$

$$U\Delta Q = D\Delta Q^{*}$$

$$Q^{N+1} = Q^{N} + \Delta Q$$

The solution is obtained in a set of forward and backward sweeps on planes where i+j+k are constant The D operator can be written as a scalar diagonal matrix. Therefore, only scalar diagonal inversions are required during each sweep since values at the preceeding plane are known. As the time step approaches infinity, the algorithm becomes a Newton-like iteration scheme. This eliminates the need to define a time step for steady state calculations.

For time accurate calculations, subiterations can be used to reduce the factorization error due the approximate Jacobians. The LU scheme can then be written as:

$$(LD^{-1}U)\Delta Q^P = -\Delta t R^{*P} \qquad (B.6)$$

Where

$$R^{*P} = R^P + Q^N \Delta V + V(Q^P - Q^N)$$

P is the current subiteration, N is the most recent time step and V is the cell volume.

REFERENCES

1. Geist, G. A., Beguelin, A., Dongarra, J., Jiang, W., Manchek,R., Sunderam, V.," PVM3 Users Guide and Reference Manual ," ORNL/TM-12187, May, 1993.

2. Dongarra, J., Geist, G. A., Manchek, R., and Sunderam, V. S., "Integrated PVM Framework Supports Heterogeneous Network Computing", Computers in Physics,Vol 5, No. 2, 1993

3. Geist, G. A. and Sunderam, V. S., "Network Based Concurrent Computing on the PVM System," ORNL/TM-11760, June 1991

4. Smith, M. H. and Pallis, J.M., "MEDUSA - An Overset Grid Flow Solver for Network-based Parallel Computer Systems,", AIAA paper no. 93-3312-CP, Proceedings of the AIAA 11th CFD Conference, Orlando, FL July 6-9, 1993.

5. Smith, M. H. "Distributed Parallel Flow Solutions Using Overset Grids," Proceedings of NASA Workshop on Distributed Computing in Aerosciences Applications, Oct. 1993

6. Raj, P., Brennan, J.E, Keen, J.M., Mani, K.K., Olling, C.R., Sikora, J.S. , Singer S.W.," Three-Dimensional Euler/Navier-Stokes Aerodynamic Method (TEAM)," Vol. 1-3., AFWAL-TR-87-3074, June, 1989.

7. Ragsdale, S. ed, *Parallel Programming*, McGraw-Hill, 1991.

8. Lorenz-Meyer, W. and Aulehla, F., " MBB Body of Revolution No. 3," AGARD AR-138, pp. C1-C31, 1979.

9. Schmitt, V. and Charpin, F. "Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers," AGARD AR-138, pp. B1-B43, 1979.

10. Hinson, B.L. and Burdges, K.P., "Acquisition and Application of Transonic Wing and Far-Field Test Data for Three-Dimensional Computational Method Evaluation," AFOSR-TR-80-0421, March 1980.

11. Johnson,J."Distributed Parallel Processing in Computational Fluid Dynamics," Proceedings of NASA Workshop on Distributed Computing in Aerosciences Applications, Oct. 1993.

12. Tijdeman, J., et al., "Transonic Wind Tunnel Tests on an Oscillating Wing with External Stores. Part II-Clean Wing," AFFDL-TR-78-194, March 1979

13. Yoon, S. and Jameson, A., "Lower-Upper Symmetric Gauss-Seidel Method for the Euler and Navier Stokes Equations," *AIAA Journal*, Vol. 26, No. 9, 1988.

14., Yoon, S. and Kwak, Dochan, "Implicit Navier-Stokes Solver for Three-Dimensional Compressible Flows," *AIAA Journal*, Vol. 30.,No. 11, 1992.

15. Chen, C.L., McCrosky, W.J., and Obayashi, S., "Numerical Solutions of Forward Flight Rotor Flow Using An Upwind Method,", AIAA-89-1846, June 1989.

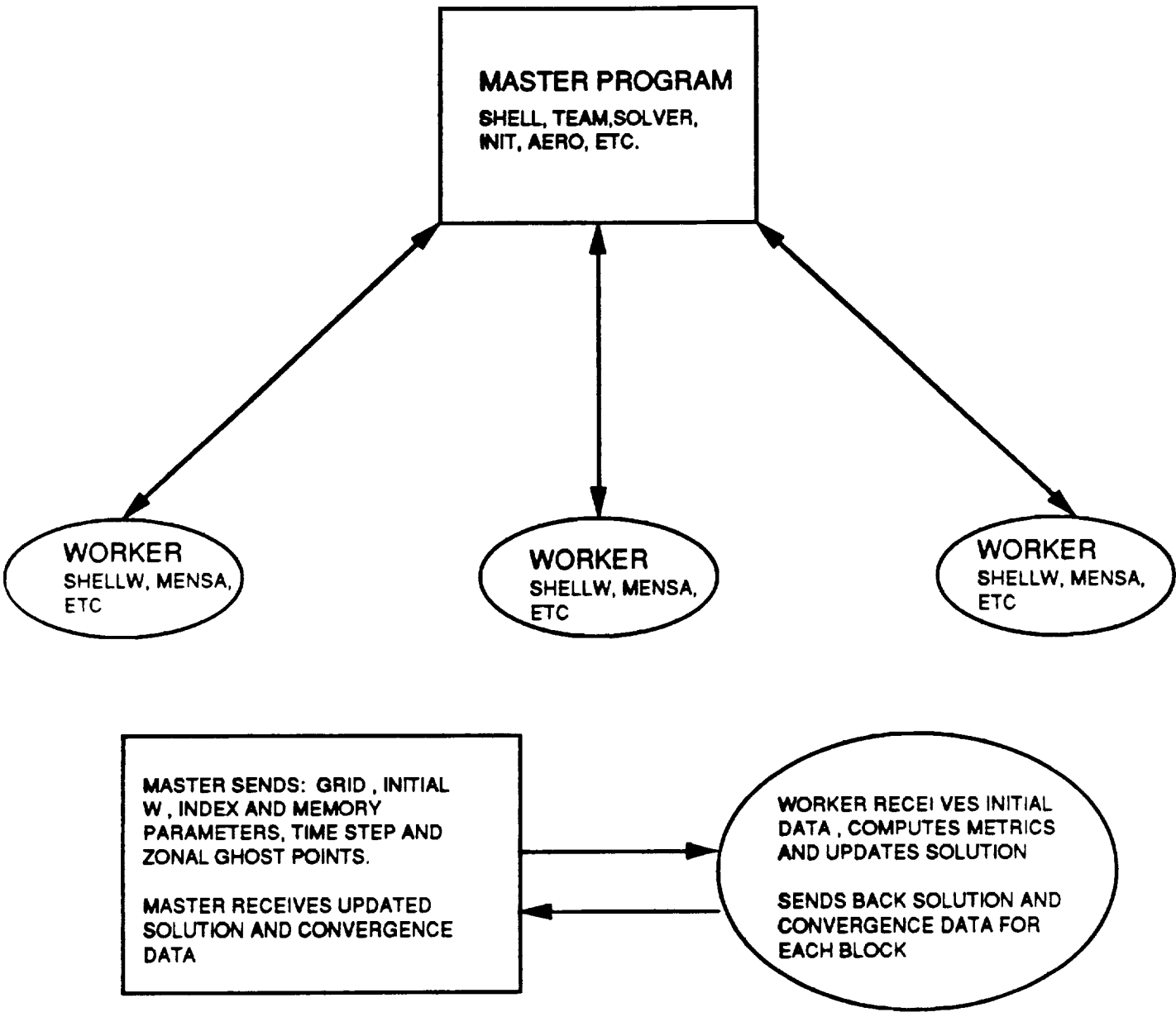


Figure 1. Manager / Worker Scheme

Baseline TEAM Psuedo code

```
SHELL
  .
  .
Input memory size parameters

Allocate memory

Call TEAM.....TEAM
                    .
                    .
                  Input grid ,flow and Bc data

                    For all grids
                       Compute volumes and metrics
                       Initialize Flow  variables
                    End

                  Call SOLVER .....SOLVER

                  STOP                    .
                                          .
                                        Do until max steps or convergence

                                          If time to update DT then
                                            For all grids
                                               Compute time step and spectral radius
                                            End
                                          End if

                                          Call MENSA.....MENSA

                                        Enddo                          .
                                                                       For L=1,NGRIDS
                                        Compute Aero data                 Update BC's
                                                                          Integrate Equations for current step
                                        RETURN                            Compute Convergence data
                                                                          Update Ghost Boundary array ZONGST
                                                                       End

                                                                       RETURN
```

Figure 2a. Baseline TEAM Pseudo-Code

PVM-TEAM Manager Pseudo code

```
TEAMM
  .
  .
Input memory size parameters

Allocate memory

Call TEAM.....TEAM
                    .
                    .
                  Input grid ,flow and BC data
                  Input virtual machine configuration

                   Start worker processes
                  Send workers memory sizing data, number of grids assigned to processor, etc.
                  Send workers grid sizes BC arrays, etc.
                  Send  required grids to each processor

                    For all grids
                       Compute volumes and metrics
                        Initialize Flow  variables
                    End

                  Call SOLVER .....SOLVER

                  STOP                    .
                                          .
                                        For all  Grids
                                           Compute initial values of time step and spectral radii
                                           Initialize ZONGST array
                                        End

                                        Send required initial W, DT's and spectral Radii to each worker

                                        Do until max steps or convergence
                                           Send complete ZONGST array , NCYC etc to workers
                                           Signal workers to call  MENSA
                                           If all workers have exited MENSA then
                                             If time to update DT then
                                               Signal workers to update time step
                                               Receive DTMIN from workers
                                             End if
                                             Signal workers to return  new ZONGST and convergence
                                             data
                                             For each worker
                                                Receive ZONGST for each grid assigned to worker
                                                Receive convergence data  for each worker
                                             End
                                             Update global convergence data
                                           End if
                                        Enddo
                                        Signal workers to return updated solutions (W) etc
                                        Receive updated solutions from workers
                                        Compute aero data

                                        RETURN
```

Figure 2.b PVMTEAM Manager Pseudo-Code

PVM-TEAM Worker Pseudo code

```
TEAMW
  .
  .
Do until signaled to exit
   Receive task id from manager
   If task id = 10 then
    Receive memory size parameters from manager
    Allocate memory
   End if
   If task id >10  and < 1000 then

     Call  SOLVEW.....SOLVEW
                                          .
                                          .
                                 If task id =20 then
                                   Receive grid sizes , global grid indexes, common data, etc
                                 End if

                                 If task id=30 then
                                   Receive grid points for each grid assigned to processor
                                   Compute volumes and initial metrics for each grid
                                   Set local memory maps for points and volumes
                                 End if

                                 If task id=40 then
                                   Receive initial W, DT, and Radii
                                   Signal manager data received
                                 End if

                                 If task id=50 then
                                    Receive ZONGST , NCYC, etc
                                    Signal manager ready to continue
                                 End if

                                 If task id = 60 then
                                   Call MENSA.....MENSA
                                    Signal manager MENSA exited
                                 End if                                  .
                                                                         .
                                                                 For LP=1, # grids on processor
                                 If task id =61 then                        Set L= global index of each grid
                                  For each grid on processor                Update local BC's
                                      Update DT and Radii                   Integrate solution
                                  End                                       Compute convergence data for processor
                                 End if                                     Update ZONGST for local grids
                                                                         End
                                 If task id =62 then                     RETURN
                                   Return ZONGST for each grid
                                   Return Convergence data for processor
                                 End if

                                 If task id = 70 then
                                  Return solution array W, DT for each grid on processor
                                 End if
   If task id=1000 then STOP     RETURN
Enddo
```

Figure 2.c PVMTEAM Worker Pseudo-Code

GEOMETRY

1102140 GRID

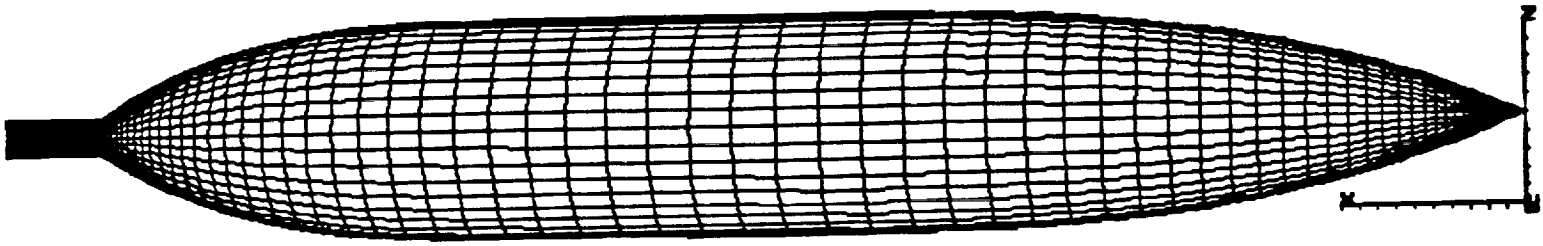


Figure 3. MBB Body No. 3 Geometry

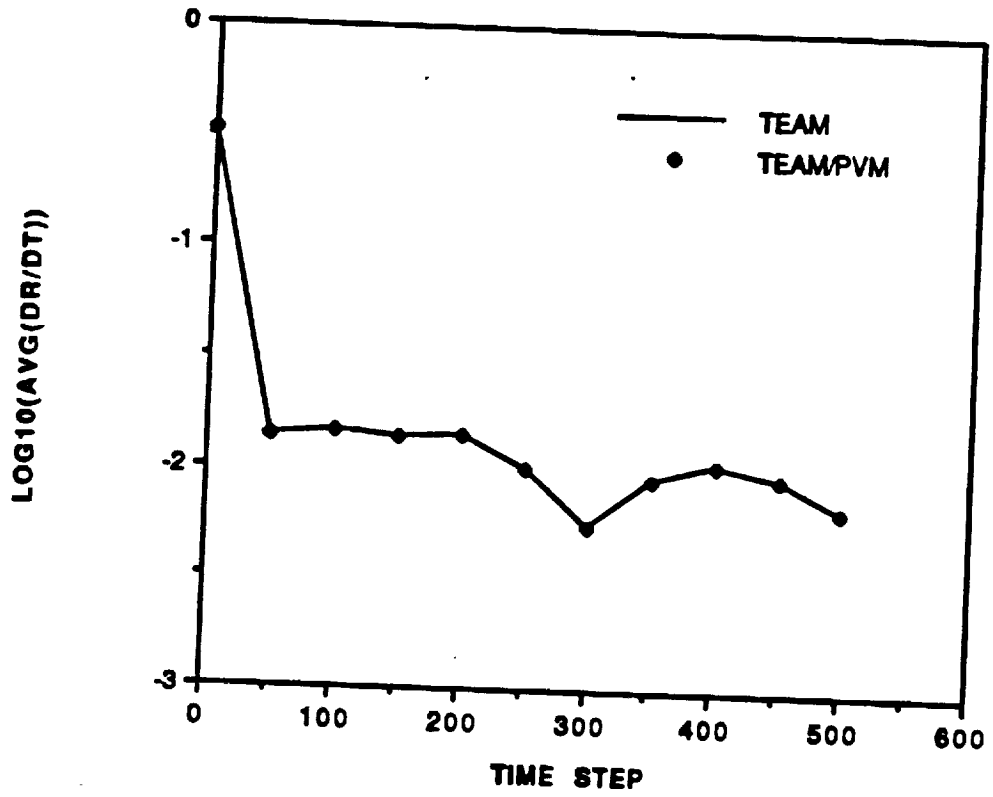


Figure 4. Correlation of the Average Change in Density per Time Step for MBB body

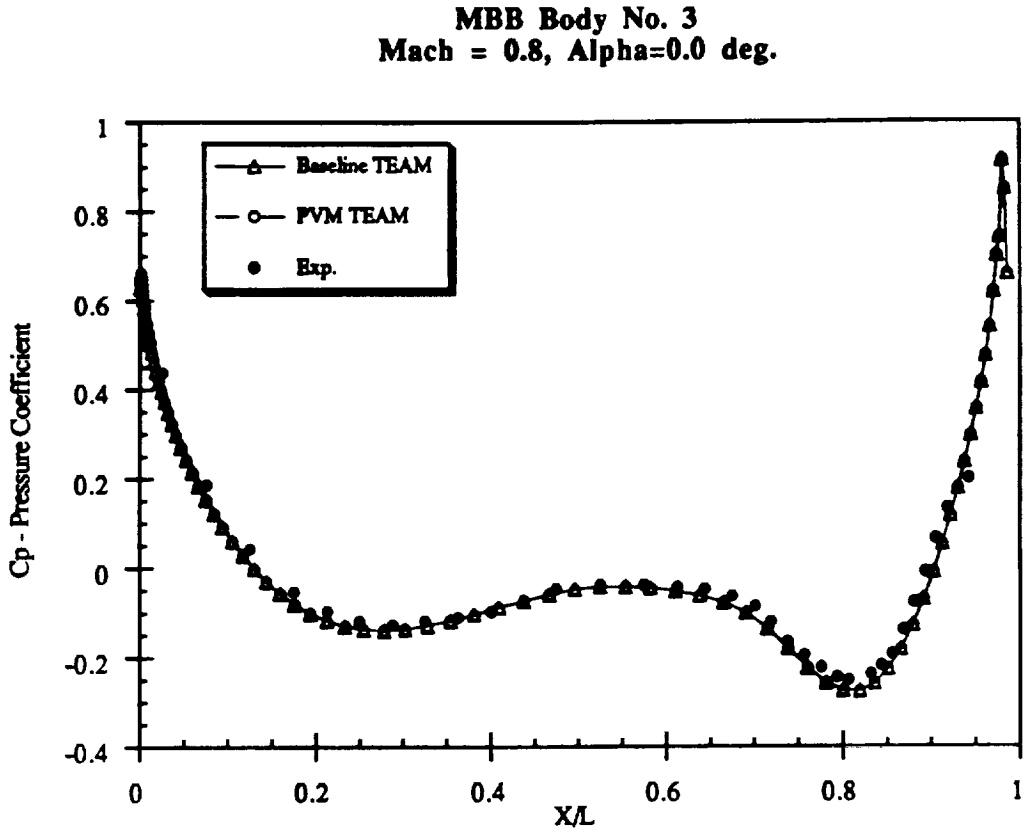


Figure 5. Correlation of MBB Body No. 3 Computed and Experimental Pressure Distributions

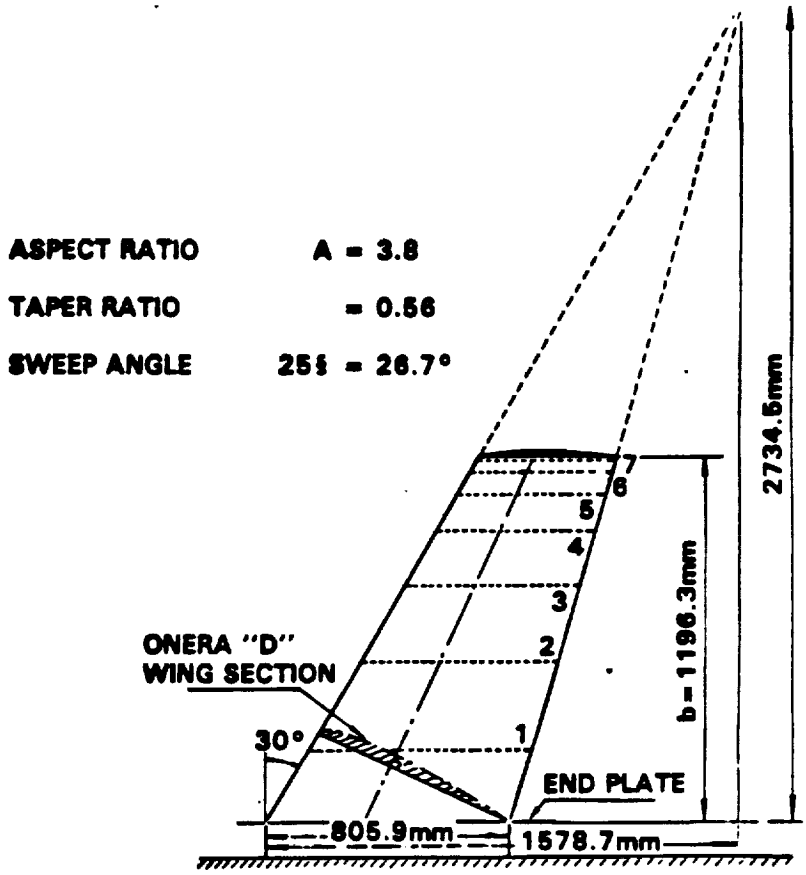


Figure 6. ONERA M6 Wing Geometry

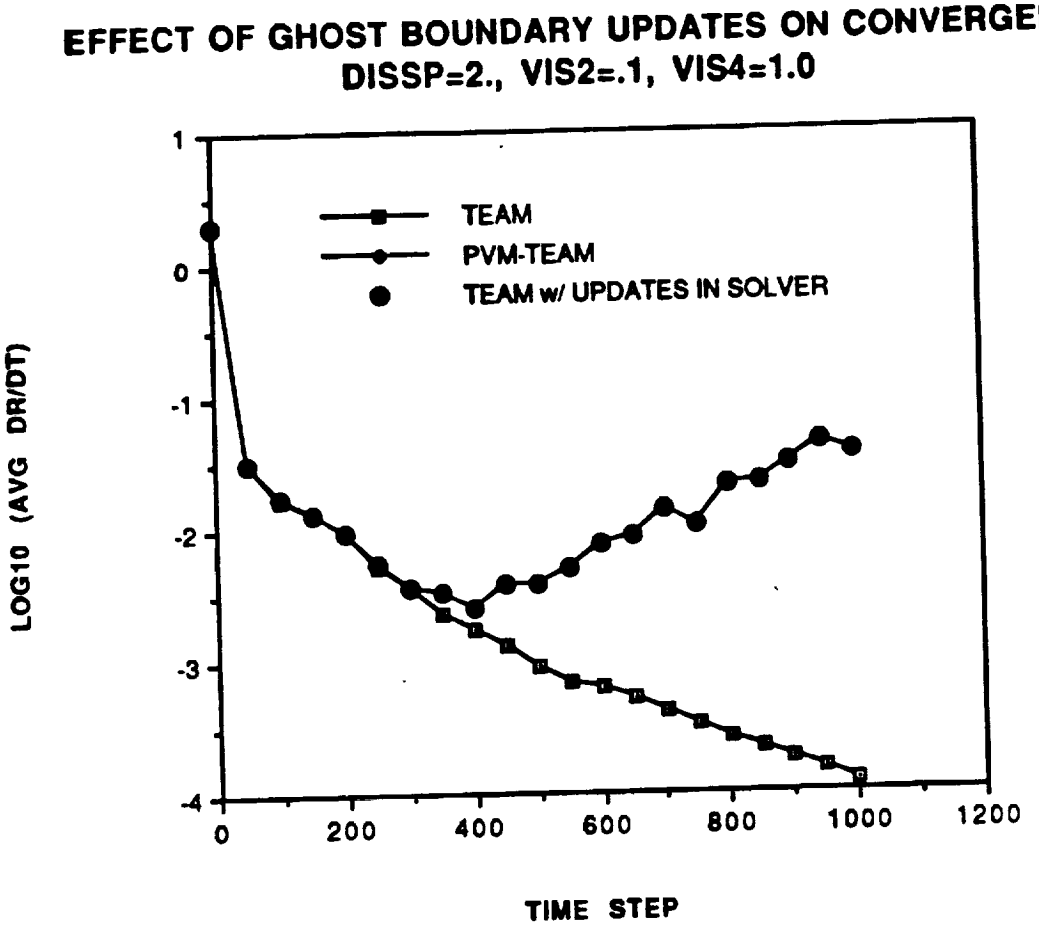


Figure 7. Effect of Lagging Ghost Boundary Updates on Convergence for ONERA M6 Wing

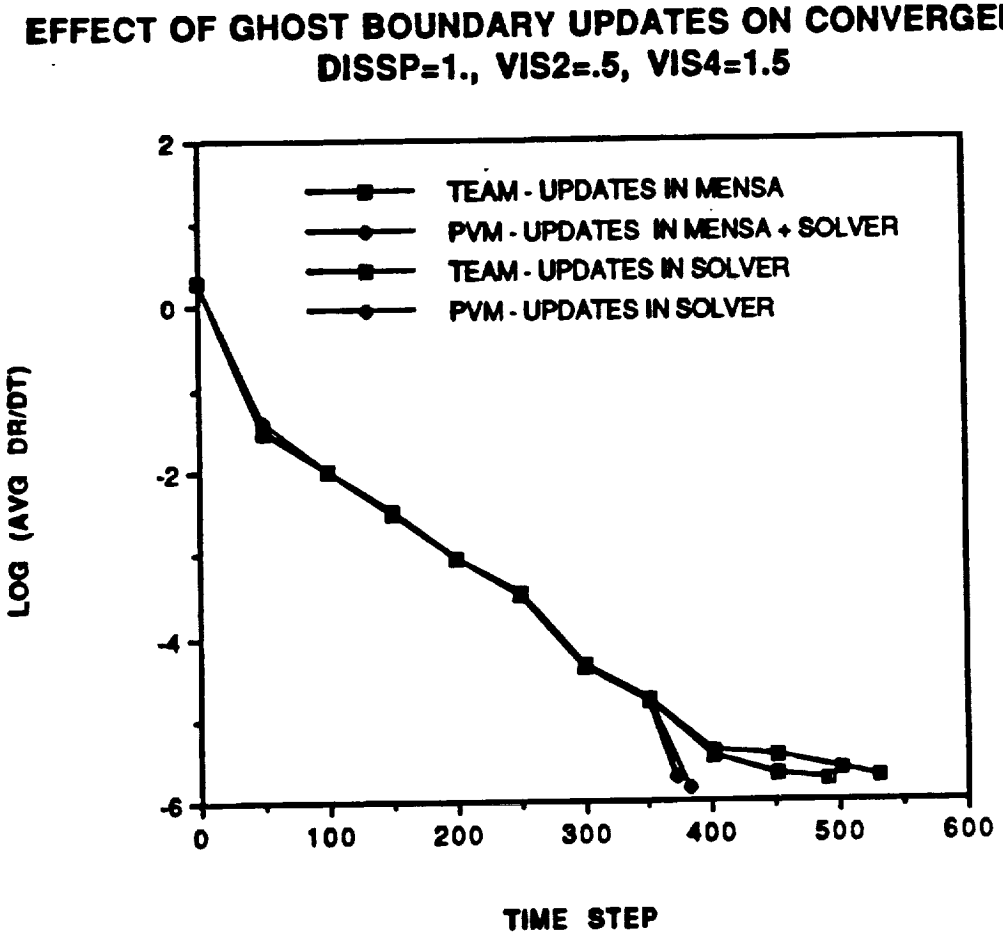


Figure 8. Comparison of ONERA M6 Convergence for Different Update Procedures

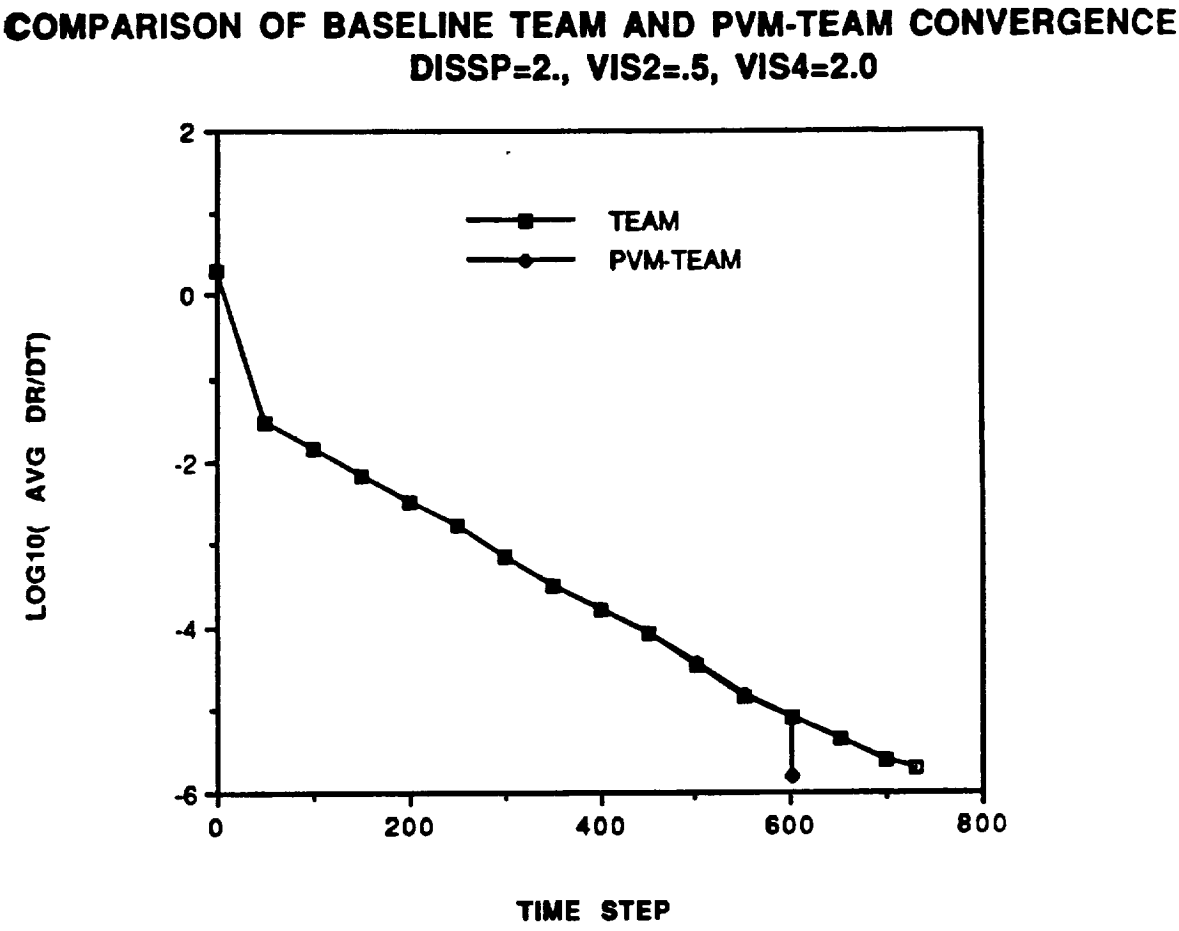
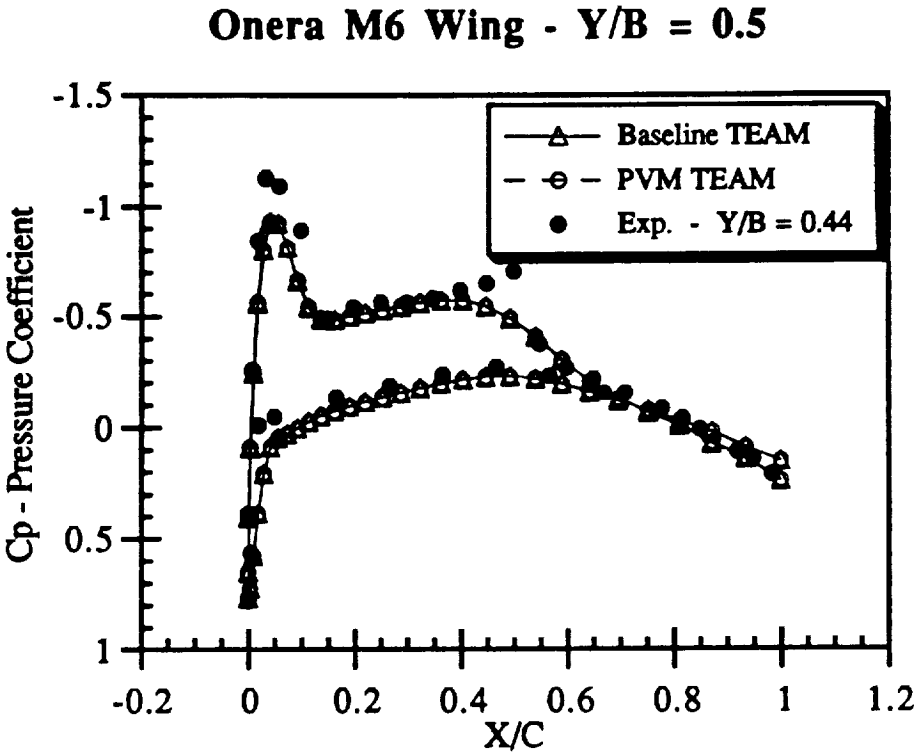


Figure 9. Comparison of ONERA M6 Convergence with Increased Dissipation



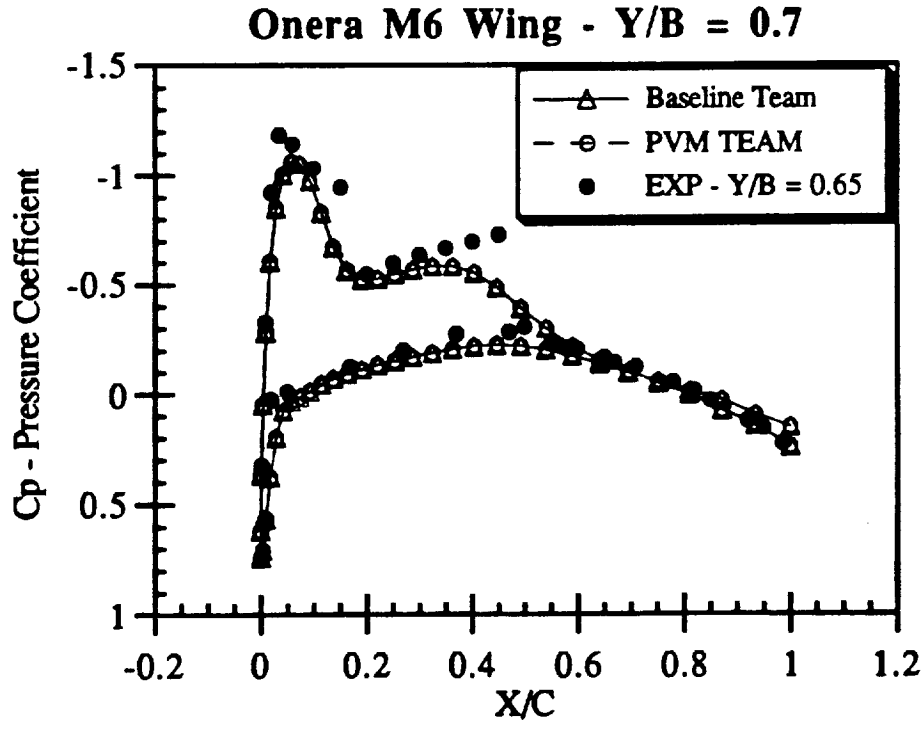
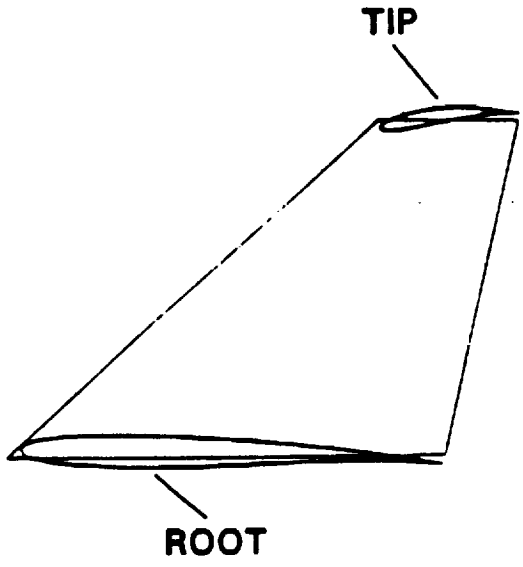


Figure 10. Correlation of ONERA M6 Computed and Experimental Pressure Distributions



AR	= 2.6
$\Lambda_{L.E.}$	= 45°
S/2	= 0.625 m^2
t_{ROOT}/C_R	= 0.06
C_{ROOT}	= 1.066 m
M.A.C.	= 0.760 m
$\alpha_{O,R}$	= 2.38°
λ	= 0.3
$\Lambda_{T.E.}$	= 9.7°
b/2	= 0.902 m
t_{TIP}/C_T	= 0.10
C_{TIP}	= 0.318 m
$\alpha_{O,T}$	= −5.79°

Figure 11. Wing C geometry

Table 1. Comparison of PVM and Baseline Team Onera M6 results

TEAM parameters are as follows : CFL= 6., SMOOTH=1., DISSP=1., VIS2=.1, VIS4=1., NSTEPS=1000., MACH=.84, ALPHA=3.06.

	TEAM	PVM TEAM	TEAM(LAGGED BC's)
Max DR/DT	4.602e-3	-1.063	-1.062
AVG DR/DT	1.212e-4	3.685e-2	3.682e-2
NSUP	834	835	835
CX	-.003922	-.003951	-.003952
CZ	.299346	.299170	.299166
CL	.299129	.298954	.29850
CD	.012063	.012025	.012024
CM	-.23072	-.23056	-.23055

DISSP=2., VIS2=.5, VIS4=2.0

	TEAM	PVM TEAM
MAX DR/DT	4.659e-5	3.205e-3
AVG DR/DT	1.88e-6	1.548e-6
NSUP	798	798
CX	-.002871	-.002870
CZ	.294527	.294528
CL	.294260	.294262
CD	.012855	.012856
CM	-.228115	-.228118

Table 1. (continued)

DISSP=1. , VIS2=.5, VIS4=1.5

	TEAM	PVM TEAM
MAX DR/DT	4.798e-5	3.134e-4
AVG DR/DT	1.787e-6	1.873e-6
NSUP	663	663
CX	-.000561	-.000560
CZ	.285250	.285234
CL	.284873	.284857
CD	.014667	.014667
CM	-.227160	-.227152

Table 2. Turnaround Performance For Onera M6 Wing Tests on HP System

Case	Workers/Points	Total Time (seconds)	% of Time Computing
1. 3 workers	730/15028	1387.	62.18
	720/11500		63.32
	720/3008		16.58
2. 3 workers	730/3008	2501.	18.29
	720/15028		84.23
	720/11500		41.62
3. 2 workers	720/15028	3750.	45.15
	720/14508		86.42

Table 3. Turnaround Performance For MBB and Onera M6 Wing Tests on DEC System

Case	Workers/Points	Total Time (seconds) PVM max	PVM min	TEAM	% of Time Computing max	% of Time Computing min
1. ONERA M6	DEC1 /15028 DEC2/14508	1280	1152	1280	35 30	56 28
2. MBB Body	DEC1/46200 DEC2/47040	4224	2560	4080	79 47	75 80

Table 4. Comparison of Baseline and PVM TEAM Results for Wing C.

	HP		ALPHA	
	Baseline	PVM	Baseline	PVM
Elapsed Time	19620 secs	29942 secs	21060 secs	24300 secs
Final DR/DT	.00138	.00169	.00138	.00167
CL	.563200	.563205	.563199	.563202
CD	.040204	.040204	.04024	.042040

COMPUTATIONAL STRATEGIES FOR THREE-DIMENSIONAL FLOW SIMULATIONS ON DISTRIBUTED COMPUTER SYSTEMS
(Abstract)

R.A. Weed and L.N. Sankar
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Phone: 404-894-6814/3014 Email: ae219rw@prism.gatech.edu

Submitted to the AIAA 25th Fluid Dynamics Conference
June 20-23, 1994

COMPUTATIONAL STRATEGIES FOR THREE-DIMENSIONAL FLOW SIMULATIONS ON DISTRIBUTED COMPUTER SYSTEMS

R. A. Weed and L. N. Sankar
Georgia Institute of Technology

INTRODUCTION

During the past decade, there has been considerable interest within the computational fluid dynamics community in harnessing the power of parallel computer architectures to improve the total throughput of CFD codes. Work in this area has centered on the development of new algorithms to take advantage of the parallelism inherent in many of the tasks that comprise the numerical solution of the Euler or Navier-Stokes equations and the porting of existing algorithms to different parallel architectures. These architectures come in three general forms: (a) massively parallel systems such as the Connection Machine CM-2, where thousands of relatively inexpensive processors are used to solve the flow equations, (b) Cray Y/MP, Intel iPSC/860, and KSR class machines where a relatively few processors are used and (c) distributed systems, where a collection of workstations (each one possibly made by a different vendor) is employed to solve the flow equations using domain decomposition to spread the solution process across the varied processors. Of these three general forms, the distributed memory systems such as the CM-2 and the Multiple Instruction Multiple Data (MIMD) systems such as the Intel and Cray systems have been the focus of much of the research in this area. Several flow solvers have been ported to these systems with varying degrees of success.

This paper focuses on the issues involved in porting an existing flow solver such as the TEAM code (Reference 1.) to the third form of parallel architectures, the distributed system. Until recently, the lack of standardized software and program interfaces to automate the passing of data and messages between different workstation architectures and the use of hardware interconnections with relatively slow data transfer rates such as Ethernet have limited distributed computing applications to tightly coupled homogeneous systems such as Digital Equipment VAX clusters. Therefore, the tens (or even hundreds) of workstations that lie idle at night and off-peak hours in the typical government, university , and industry research organization represent an underutilized computing resource of enormous potential. The relatively low cost of ownership of today's engineering work stations and the increasing ratio of performance to price make them an viable alternative to large specialized architectures if the performance bottlenecks imposed by data communication between machines and load balancing can be overcome.

The potential of distributed computing systems has led to considerable research activity in the past five years that has yielded the following breakthroughs:

(a) Standardized application program interfaces such as the Parallel Virtual Machine (PVM) interface described in Reference 2. have become available. These are user callable

libraries that provide generic functions to high level languages such as FORTRAN or C that hide the details of passing data and messages between clusters of heterogeneous workstations that are defined by the user. In addition, support is provided for dynamically re configuring the system to add or delete processors as required. This helps to make the system fault-tolerant and eliminates some of the problems imposed by load balancing requirements.

(b) New high speed data transfer technologies such as the FDDI supported by many vendors, and the Gigaswitch hardware developed by Digital Equipment Corp. are becoming widely available. These new technologies will provide significant reductions in the overhead imposed by data communication.

(c) New CPU designs such as the DEC ALPHA processor can perform floating point operations at speeds as high as 200 MFLOPS. This increase in speed allows workstations to be utilized for the large scale flow solutions formerly reserved for CRAY class supercomputers.

Because of these breakthroughs, many researchers are now actively performing innovative research to port existing large scale flow solvers to distributed architectures. One such pioneering effort was recently described in Reference 3. In addition, work is underway to develop algorithms that are fine tuned for distributed systems. This paper describes a joint effort between Lockheed Aeronautical Systems Company, Digital Equipment Corp. and Georgia Tech funded by NASA Ames to investigate the issues involved in porting, fine-tuning and improving the algorithms in existing flow solvers for optimum performance on fast distributed systems such as the DEC AXP workstations..

APPROACH

The PVM system described in Reference 2. has been used to modify the TEAM (Three-Dimensional Euler/Navier-Stokes Aerodynamic Method) code (Reference 1) developed at Lockheed Corporation to function as a distributed parallel flow solver. The baseline TEAM code is a multi-block solver that solves the Euler or Navier-Stokes equations for complex configurations using explicit Runge-Kutta time stepping to integrate a finite volume discretization of the governing equations. The solver can be used for both steady and unsteady applications. Acceleration techniques such as local time stepping, residual smoothing, and enthalpy damping can be used to speed up steady state solutions. The version of TEAM used in this research also supports dynamic memory allocation. This eliminates the need to recompile the code for when grid dimensions change.

Of the available strategies for implementing distributed parallel systems, the Manager/Worker approach described in References 3 and 4 was adopted for this research. A simple diagram of this approach as applied to the TEAM code is shown in Eigure 1. In the Manager/Worker scheme, a control program, the Manager, controls the allocation of individual tasks to the separate processors and functions as the central point of

communication of data required by the individual tasks. The Manager inputs the information required by the solution process (flow conditions, boundary conditions, grids etc.) and sends this information to the individual Worker processes running on individual workstations. The worker processes integrate the flow solution for a single grid or group of grids assigned to each workstation. At the end of each time step or iteration, the workers return the updated solution and convergence information for each grid system. The manager updates the boundary conditions at block interfaces and sends this information to the workers and then signals each worker to perform a new time step.

The Manager/Worker approach simplifies the load balancing for a distributed system with processors of different speeds and memory sizes. In addition, having one process control the flow of data reduces the number of messages that are passed to and from each worker process and eliminates the need for workers to communicate with each other. The PVM system allows messages from the workers to the manager to be processed on a first in - first out basis. Therefore, the manager does not have to wait for all the worker processes to finish before it processes the updated solution received from the individual workers. This helps reduce idle time spent waiting for messages to arrive.

TEAM CODE MODIFICATIONS

The modifications to the TEAM code for the initial phases of this research have been kept to minimum. Since initial testing is being performed on a homogenous cluster of workstations, an elaborate load balancing algorithm has not been implemented. Initial testing is being performed with grid blocks of similar size. The worker process was constructed from the core flow solver routines in TEAM (MENSA, etc.). The manager program retains the dynamic memory allocation, input and output routines, and aerodynamic loads calculations of the original code in addition to the task sequencing and data communication code required by PVM. The initial version of the code processes all data to and from the workers in sequence. This imposes a performance penalty on the code but simplifies the verification of the data communications and the sequencing of tasks. In addition, the boundary conditions at grid interfaces will be lagged by one iteration because all the worker processes must finish a step before the Manager can perform the update. This differs from the baseline TEAM code that updates the boundary conditions using the most recently available results as it sequences through a set of grids. Results for improved versions of the code that remove these restrictions will be presented in the final paper.

RESULTS AND CONCLUSIONS

An initial tests of the distributed parallel TEAM code has been performed using a two block grid system to solve the steady Euler equations for the MBB body of revolution (Reference 5.) shown in Figure 2. The initial flow conditions are a Mach no. of 0.8 and zero angle of attack. The grid system is composed of two blocks of 55x21x40 and 56 x21x40 grid points. The code was run on a distributed system composed of three Hewlett-Packard PA-RISC workstations, a model 730 with 48Mbytes of memory and two

model 720 workstations with 32 Mbytes and 16 Mbytes of memory respectively. A series of tests were conducted with the manager and worker processes rotated among the three machines. The convergence rate, computed loads, and pressure distributions from these tests were compared with results from the baseline code running on a single processor (the model 730 system). A comparison of the average change in density per unit time for the two codes is shown in Figure 3. The slow convergence rate is due to the low CFL number (CFL=1) used for these tests. The convergence rates are virtually identical and indicate that the modified procedure for updating grid interface boundary conditions has little impact on convergence rate. The surface pressure distributions along the leeward symmetry plane are compared in Figure 4. The two codes are seen to produce identical results. The turnaround performance (measured by the total elapsed time, not CPU time) of the distributed code varied with the overall load on each of systems. Turn around times ranged from 3 to 8 hours for 500 time steps. This compares with an range of 3 to 4 hours for the baseline code. The best performance of the distributed code was obtained with the manager process running on the model 730 and the two worker processes running on the model 720 workstations. The wide range of performance of the initial code illustrates the problems that can be encountered when the distributed code is run during periods of heavy utilization by other users. However, these results indicate that an effective distributed system can be implemented with relatively minor modifications to the baseline code. An improved version of the solver is being developed with more elaborate task sequencing and load balancing that should result in much improved performance.

The final paper will present results for a steady viscous and inviscid flow about a standard CFD test geometry such as the Lockheed/AFOSR Wing C configuration (Reference 1). Results will be presented for various load balancing schemes and system configurations. In addition, modifications to the TEAM solution algorithm to improve performance on distributed systems will be explored. Computations will also performed on a system of DEC ALPHA chip workstations to be provided by Digital Equipment Corp using a version of PVM fine tuned for the DEC workstations. The final paper will discuss in detail the issues involved in porting an existing flow solver such as TEAM to a distributed parallel system and the cost effectiveness of using such systems in a production environment.

ACKNOWLEDGMENT

This research is being performed under a contract funded by NASA Ames Research Center. Dr. Terry Holst is the contract monitor. The authors would also like to acknowledge the support of Dr. Pradeep Raj of the Lockheed Aeronautical System Co. and Olin Holly and Brad Lowe of Digital Equipment Corp.

REFERENCES

1. Raj, P., Brennan, J.E, Keen, J.M., Mani, K.K., Olling, C.R., Sikora, J.S. , Singer S.W.," Three-Dimensional Euler/Navier-Stokes Aerodynamic Method (TEAM)," Vol. 1-3., AFWAL-TR-87-3074, June, 1989.

2. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V.," PVM3 Users Guide and Reference Manual ," ORNL/TM-12187, May, 1993.

3. Smith, M. H. and Pallis, J.M., "MEDUSA - An Overset Grid Flow Solver for Network-based Parallel Computer Systems,", AIAA paper no. 93-3312-CP, Proceedings of the AIAA 11th CFD Conference, Orlando, FL July 6-9, 1993.

4. Ragsdale, S. ed, **Parallel Programming**, McGraw-Hill, 1991.

5. Lorenz-Meyer, W. and Aulehla, F., " Experimental Data Base for Computer Code Assessment," AGARD AR-138, pp. C1-C31, 1979.

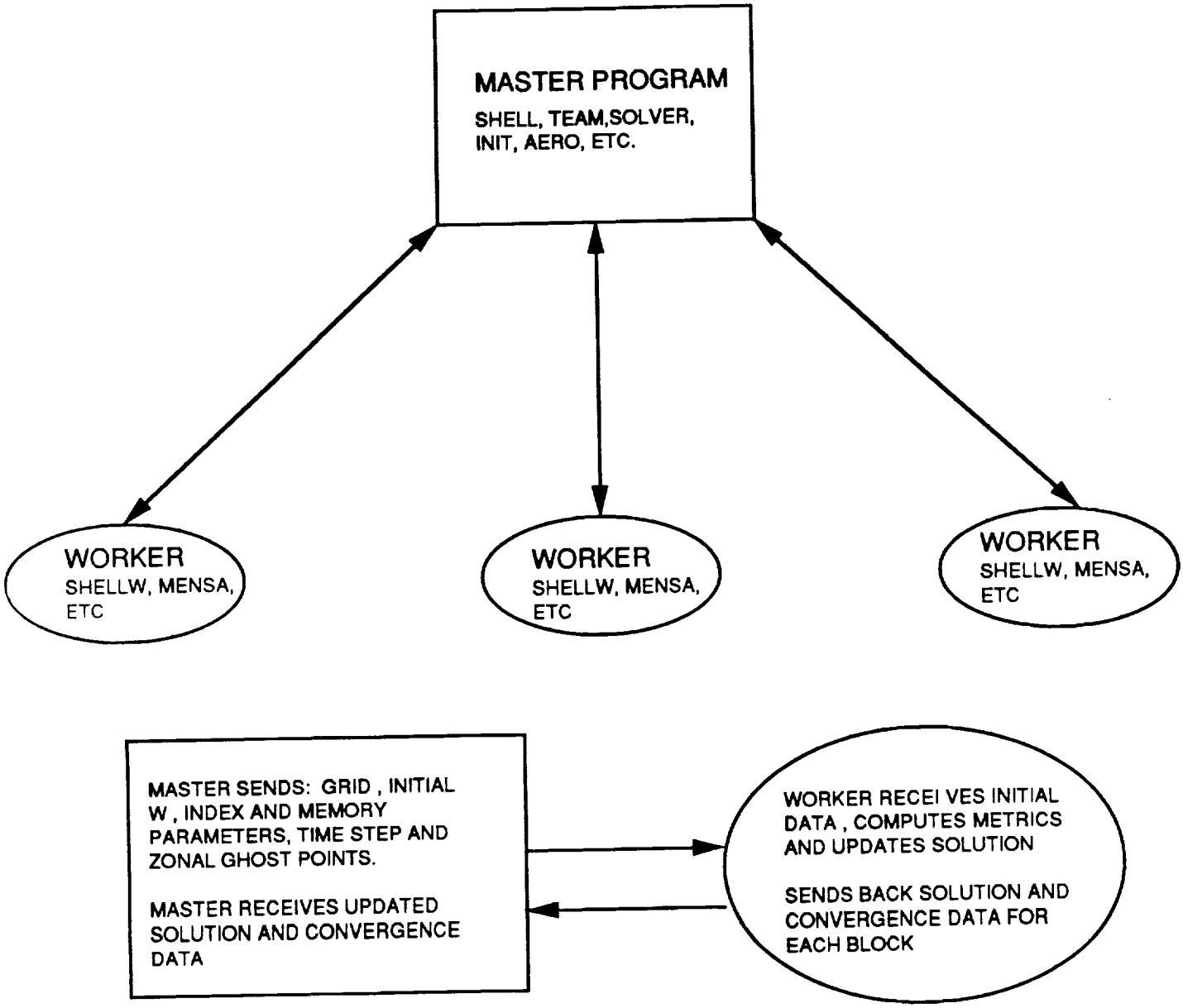


Figure 1. Manager / Worker Strategy For Team

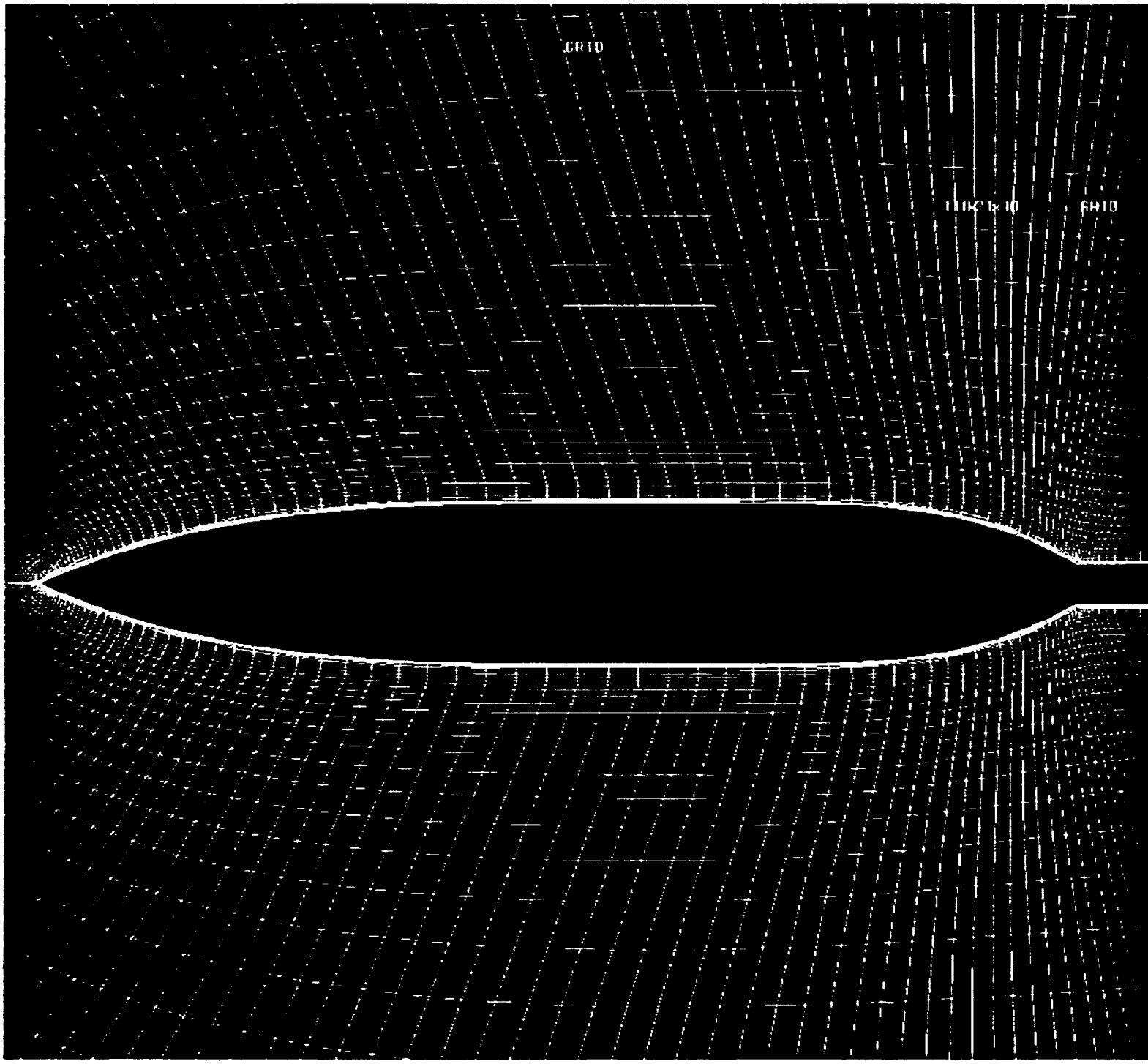


Figure 2. MBB Body Grid

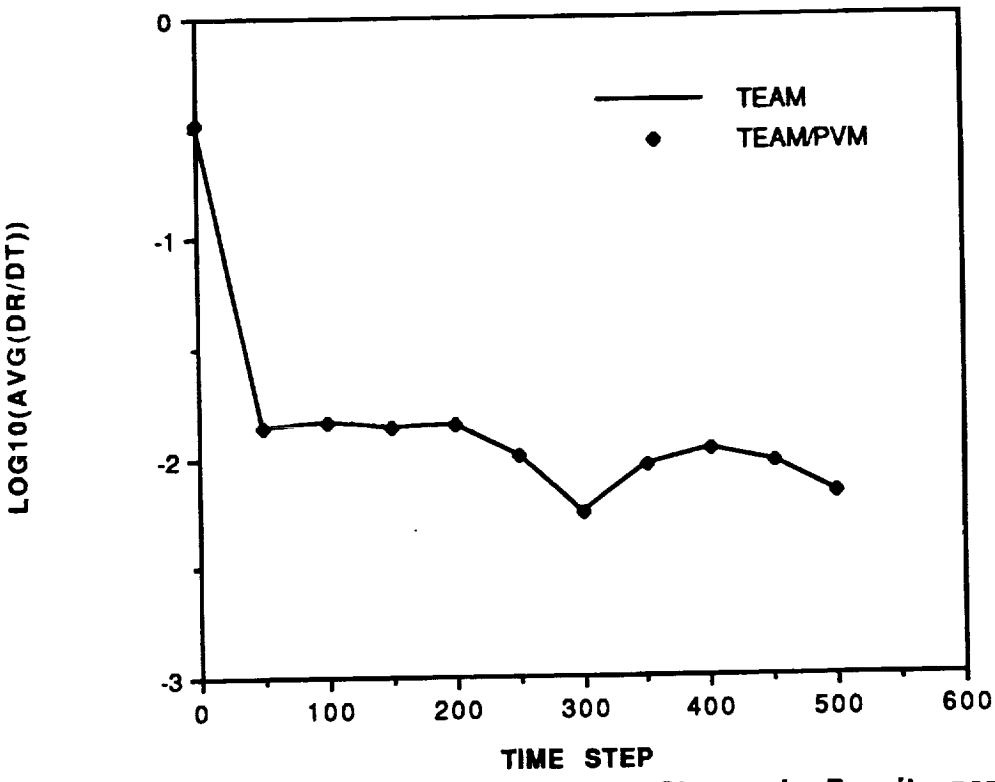


Figure 3. Correlation of the Average Change in Density per Time Step

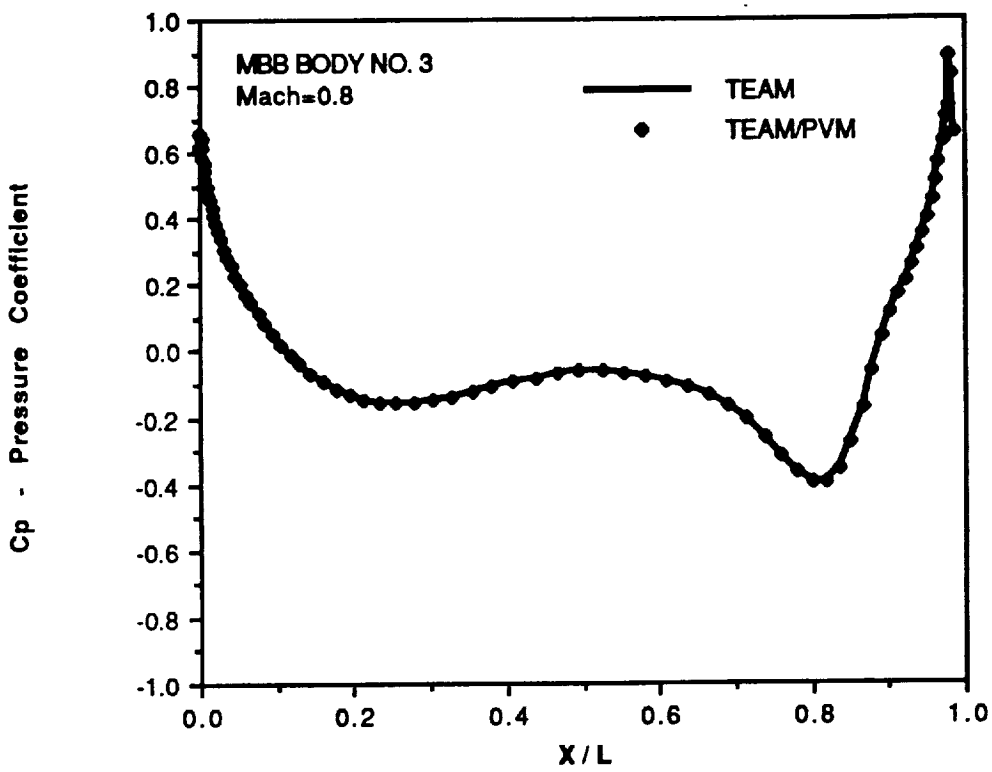


Figure 4. Correlation of Leeward Pressure Distributions Computed By Baseline and Distributed Team Codes