AIAA

AIAA 94-0760

A Massively Parallel Solution of the Three Dimensional Navier-Stokes Equations on Unstructured, Adaptive Grids

Z. Weinberg and L. N. Long
The Pennsylvania State University
University Park, PA

32nd Aerospace Sciences Meeting & Exhibit
January 10-13, 1994 / Reno, NV

AIAA-94-0760

A Massively Parallel Solution of the Three Dimensional Navier-Stokes Equations on Unstructured, Adaptive Grids

Zvi Weinberg*
Lyle N. Long†
The Pennsylvania State University
Department of Aerospace Engineering
233 Hammond Building
University Park, PA 16802
(814) 865-1172

Abstract

A massively parallel solver for the Navier-Stokes equations is presented. The solution is achieved explicitly by marching the time dependent Navier-Stokes equations using the four stage Runge-Kutta method. Roe's approximate Riemann solver is used to upwind the fluxes. The code uses unstructured grids with hexagonal, prismatic or tetrahedral cells. The algorithm includes a directional adaption capability according to momentum gradients. The computational results show good agreement with the exact solution of a laminar boundary layer even in cases where relatively few cells are allocated in the viscous region. Good agreement is also obtained with transonic flow over over a two dimensional airfoil and a three dimensional delta wing.

1 Introduction

The advances in computational fluid dynamics (CFD) combined with the increased availability of computers means CFD will play a major role in the future design of aircraft. Using CFD as part of the design cycle reduces the the risk and design time, and opens new design options.

The complexity of the configurations, as well the harsh flow regimes for modern aircraft increases the demand on new computers. Currently the performance of conventional vector machines is leveling off and introduction of new versions of these machines results in only modest improvement in performance.

*Doctoral candidate
†Assistant Professor of Aerospace Engineering. Senior member, AIAA

A breakthrough in computer performance is possible, as has been demonstrated in the last years [1], by using the massively parallel machines [2]. Massively parallel computers use very large number of processors operating simultaneously in either SIMD (Single Instruction Multiple Data), MIMD (Multiple Instruction Multiple Data) or a combination of the two.

The availability of massively parallel machines in the market created a need for software which is capable of taking advantage of the new technology. Several problems arise when an efficient implementation on a massively parallel machine is sought. The most time consuming part of a massively parallel computation is the interprocessor communication rather than floating point operations [3]. An effort must be directed, thus, to more efficient communication. An efficient treatment of boundary conditions is mandatory in massively parallel applications. Typically only a small portion of the processors are active during boundary conditions implementation while the rest are idle. This causes the process to be inefficient, and can significantly decrease the advantage of using massively parallel machines.

A large part of the total turnaround time, for a typical complex geometry computation, is dedicated to grid generation. Efficient conventional structured grid generation for a complex geometry can be very laborious and time consuming. Such grids have to be orthogonal and the domain discretization has to comply with the flow field characteristics (boundary layers, shock waves etc.). Consequently the generation time of a conventional grid for a complex geometry can be sometimes as much as months. Furthermore, it is very difficult to simulate the fine details of the configuration using conventional grids.

On the other hand unstructured grids are much more flexible in simulating complex geometries. The orthogonality requirement is relaxed in such grids and these grids do not have to be mapped into a rectangular computational space. In addition, unstructured grids can be adapted much easier then the conventional ones. These features make unstructured grids very attractive for CFD for complex geometries. The generation of unstructured grids can be done without much effort using commercially available software such as IDEAS or PATRAN, for mildly complex configurations, or dedicated software (i.e. [4], [5]) for more complicated geometries.

The current work presents a development of a massively parallel solver for the compressible, viscous flow over complex geometries. Section 2 describes the main features of the algorithm. The implementation on a massively parallel machine (CM-200) is discussed in section 3 and the grid adaption is described in section 4. Some results are given in section 5, and a performance summary is provided in section 6.

2 Algorithm

2.1 Discretization of the governing equations

The solution algorithm is based on marching the time dependent Navier-Stokes equations from a given initial flow to a steady state. The governing equations are written in a control volume representation:

$$\frac{\partial}{\partial t}\int\int\int_V \mathbf{U}dV = \int\int_S \vec{\mathbf{F}}\cdot\hat{n}dS \qquad (1)$$

The dependent variable vector, $\mathbf{U}$, consists of density, three components of momentum and total energy. The flux vector, $\vec{\mathbf{F}}$ includes the three components of both inviscid and viscous contributions.

Discretization of the governing equation is done by taking the average values of $\mathbf{U}$ in the space discretization cell and average fluxes through its faces. The averaging leads to the approximated equation for the cell c surrounded by N_{f_c} faces:

$$\frac{\partial \mathbf{U}_c}{\partial t} = \frac{1}{V}\sum_{k=1}^{N_{f_c}}\vec{\mathbf{F}}_k\cdot\hat{n}_k S_k \qquad (2)$$

where $\mathbf{U}_c$ is the cell averaged state vector, $\vec{\mathbf{F}}_k$ is the face averaged flux vector and $\hat{n}_k$, S_k are the unit normal and the area of the face k.

2.2 Computational grid

The flowfield is discretized into an unstructured assembly of three dimensional cells. The program supports hexagonal, tetrahedral and prismatic cells, or any combination of the three.

The discretization scheme, as presented in Equation 2, requires two different data sets: cell based (the state vector $\mathbf{U}$, for example) and face based (the fluxes $\vec{\mathbf{F}}$). The connectivity between the two sets consists of two face based address arrays pointing to the two cells on the "left" and on the "right" of each face. The definition of left and right is associated with the direction of the face normal such that it points from the "left" cell to the "right" (Figure 1).

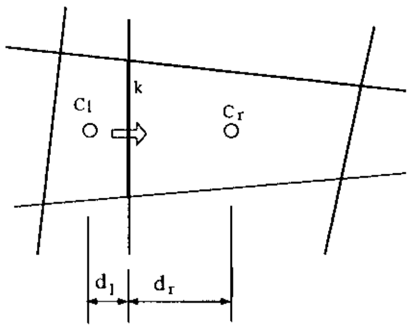

Figure 1: face to cell distances

2.3 Time integration

The temporal discretization of Equation 2 results in:

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \frac{\Delta t}{V}\sum_{k=1}^{N_{f_c}}\vec{\mathbf{F}}_k^n\cdot\hat{n}_k S_k = \Delta t\mathbf{R} \qquad (3)$$

The time stepping is performed using a four stage Runge-Kutta scheme. The scheme used here is a full fourth order scheme given by:

$$\begin{aligned}\mathbf{U}^{(0)} &= \mathbf{U}^n\\ \mathbf{U}^{(1)} &= \mathbf{U}^n + \alpha_1\Delta t\mathbf{R}^{(0)}\\ \mathbf{U}^{(2)} &= \mathbf{U}^n + \alpha_2\Delta t\mathbf{R}^{(1)}\\ \mathbf{U}^{(3)} &= \mathbf{U}^n + \alpha_3\Delta t\mathbf{R}^{(2)}\end{aligned} \qquad (4)$$

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\sum_{i=1}^{4}\beta_i\mathbf{R}^{(i-1)}$$

where

$$\alpha = \left[\frac{1}{2},\frac{1}{2},1\right] \ ; \ \beta = \left[\frac{1}{6},\frac{1}{3},\frac{1}{3},\frac{1}{6}\right]$$

The complete fourth order scheme has been chosen over the compact form suggested by Jameson and Baker [6] since it gives a higher convergence rate as shown in Figure 2.

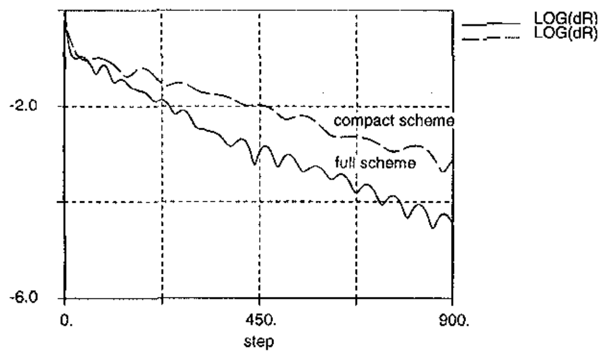

Figure 2: comparison between the full 4th order Runge-Kutta scheme and the compact one

The time step is calculated locally, based on the cell size and the maximal value of the disturbance propagation speed from its faces. The local time stepping can be switched off for time accurate computation. In such cases the whole flowfield is marched forward using the smallest time step found in the flow field.

2.4 Piecewise linear reconstruction

The flux calculation requires the evaluation of the state vector at the faces. This is done by reconstructing a cellwise linear spatial variation of state variables.

The linear reconstruction is done in two steps. First a linearly interpolated state vector, U, is calculated at the faces (see Figure 1):

$$\mathbf{U}_k^* = \frac{\mathbf{U}_l d_r + \mathbf{U}_r d_l}{d_l + d_r} \tag{5}$$

Next the cell centered gradients are calculated. The gradients calculation is based on Gauss' theorem:

$$\int\int\int_V \nabla \mathbf{U} dV = \int\int_S \mathbf{U}\hat{n} dS \tag{6}$$

When assuming a constant value of $\nabla\mathbf{U}$ inside a given cell, and using the face interpolated values of $\mathbf{U}^*$, the cell centered gradients are obtained by:

$$(\nabla \mathbf{U})_c = \frac{1}{V}\sum_{k=1}^{N_{fc}} \mathbf{U}_k^* \hat{n}_k S_k \tag{7}$$

The cell centered gradients are used to evaluate the face based state variables:

$$\mathbf{U}_k = \mathbf{U}_c + (\nabla \mathbf{U})_c \cdot \delta\vec{r} \tag{8}$$

where $\delta\vec{r}$ is the vector from the cell center to the face center.

Evaluation of the state vector at the faces using the linear reconstruction procedure would result, generally, in two different values of U for each face, corresponding to the "left" and "right" cells reconstructions. The situation, at this stage, resembles thus the Riemann problem in which a membrane, separating between two different states, bursts. The flux evaluation is modified next by using Roe's approximate Riemann solver [9], consisting of upwind biasing the fluxes.

2.5 Upwind Biasing

The upwinding scheme is used to bias the relative weight given to the two cells on both sides of a given face according to the characteristic direction of the signal propagation from the face:

$$\vec{\mathbf{F}}_k = \frac{1}{2}\left[\vec{\mathbf{F}}(\mathbf{U}_l) + \vec{\mathbf{F}}(\mathbf{U}_r) + \mathbf{b}_k\right] \tag{9}$$

The face related bias is calculated by [10]:

$$\mathbf{b}_k = |\mathbf{A}_k|(\mathbf{U}_l - \mathbf{U}_r) \tag{10}$$

where $\mathbf{A}_k$ is an approximated face based Jacobian matrix exhibiting Roe's property [9]. **A** is calculated based upon the density ratio weighting of the state vector in the "left" and "right" cells (Roe averaging).

The matrix $|\mathbf{A_f}|$ is obtained by decomposing the approximated Jacobian into the eigenvalues and right eigenvector matrices:

$$\mathbf{A_f} = \mathbf{R}^{-1}\mathbf{\Lambda}\mathbf{R} \tag{11}$$

and replacing the eigenvalue matrix , $\mathbf{\Lambda}$, by $|\mathbf{\Lambda}|$ defined as follows:

$$|\mathbf{\Lambda}_{ij}| = \begin{cases} \tilde{\lambda}_i, & \mathrm{i = j} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where $\tilde{\lambda}$ is given by:

$$\tilde{\lambda} = \begin{cases} |\lambda|, & |\lambda| \geq 2\beta \\ \lambda^2/4\beta + \beta, & \text{otherwise} \end{cases} \tag{13}$$

Here β serves as a an eigenvalue limiter and its value is taken in the range:

$$0.1 \leq \beta \leq 0.3 \tag{14}$$

2.6 Boundary conditions

Application of boundary conditions on a massively parallel computer has to be carried out with maximum efficiency. At the stage of boundary conditions implementation only a small portion of the cells are "active" (the boundary cells) while the majority of the cells (interior cells) are idle rendering the massively parallel code inefficient if not done properly.

The emphasis in boundary conditions application in the present method is, thus, on simplicity which translates into essentially a "zero cpu time" requirement for the stage at which only the boundary cells are active. The "zero cpu time" stage is followed by upwind biasing the boundary faces value. The upwinding is, however, applied simultaneously to all the faces in the flow field (boundary and interior) so it does not diminish the efficiency of the parallel process.

The implementation of the boundary condition is done by constructing hypothetical ghost cells on the outer sides of the boundaries. The values of the flow properties in these ghost cells are determined by a simple substitution of the known values of the corresponding boundary cell according to the type of boundary under consideration:

- **Solid boundary** - opposite direction of momentum, compared to the corresponding boundary cell, and the same values of density and energy. This implies zero momentum (velocity) on the solid boundary and zero density and energy (pressure) gradients normal to the wall.

- **Far field** - undisturbed values of density, momentum and energy are assigned to the ghost cells. It is important to note that the application of the upwind biasing to this boundary renders the far field boundary nonreflective.

- **Out flow** - the values of U assigned to the ghost cells are equal the ones of the boundary cells zeroing the gradients normal to the outflow boundary.

- **Symmetry plane** - can be chosen along constant x, y or z. The ghost cell values consist of the same scalar properties (density and energy) and mirrored values of the momentum vector, as compared to the boundary faces.

3 Massively parallel implementation

The code has been written in FORTRAN 90 and executed on a data parallel machine (CM-200) and on the CM-5. The cell related data (such as the values of U) as well as the face related data (i.e. $\vec{\mathbf{F}}$) are distributed among the virtual processors such that each processor "represents" an individual cell or face. A single instruction applied to all cells (or faces) is being performed simultaneously, reducing the total turnaround time.

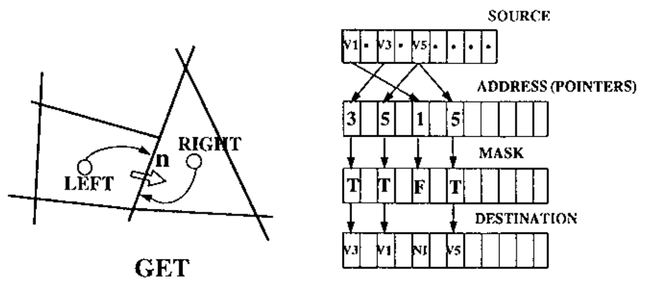

Figure 3: Schematic description of the GET instruction

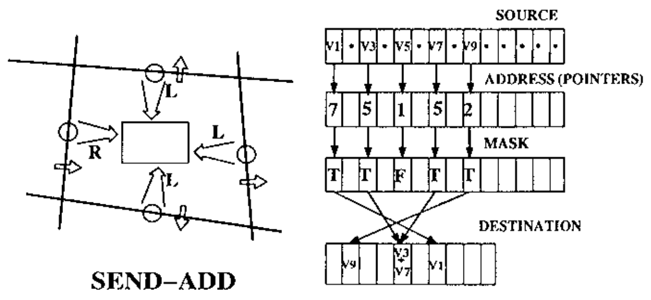

Figure 4: Schematic description of the SEND-WITH-ADDS instruction

3.1 Communication and Data Transfer

The inter-processor communication is invoked to transfer data between the cells and the faces. The communication is handled by the GET and SEND-WITH-ADDS instructions which are responsible for getting the data from the cells to the corresponding faces (GET - Figure 3) and collecting the sum of the data on the faces surrounding a given cell into a cell related array (SEND-WITH-ADDS - Figure 4).

The cell-faces communication path (ADDRESS array in Figures 3 and 4) is established by LEFT and RIGHT connectivity arrays (see Section 2.2). Using this type of connectivity makes the code very flexible in terms of the geometrical shape of the finite volume cells (all faces around any polygonal cell would have exactly two adjacent cell, regardless of the number of faces per cell).

4 Grid adaption

The computational grid in the current work can be adapted according to the computed flow field. The grid adaption enables starting a specific run with a relatively coarse grid and then refining it in the high error regions, reducing the overall error without paying the price of running initially with a fine grid and a large number of cells.

The grid adaption process consists of two main steps. First the cells in which the error indicator is large enough are flagged and then these cells are split, refining the grid in the high error regions.

4.1 Error indicator evaluation

The error indicator chosen in this work is based on the momentum gradient normal to the faces. This indicator gives better results then other variables proposed in the literature (i. e. total stress [11] density [12] or entropy gradients).

The advantage of using momentum gradients as an error indicator is twofold: first it is readily available in the code (as a dependent flow variable) so no time needs to be spent on its computation, and second, it "highlights" the appropriate computational cells whether they reside along a shock wave, inside a boundary layer or along a free shear layer.

The momentum gradient is computed in the direction normal to the faces by:

$$\nabla_n\left(\rho\vec{V}\right) \simeq \frac{\rho\vec{V}_l - \rho\vec{V}_r}{d_l + d_r} = \left(\rho\vec{V}_l - \rho\vec{V}_r\right) d^{-1} \qquad (15)$$

The distances d_l and d_r are defined in Figure 1.

Further modification to the error indicator equation is done by altering the power of the characteristic distance, d, in the right hand side of Equation 15. The purpose of this is to give more weight to the yet unrefined cells in the flow field smoothing out the adaption pattern. So the expression used to compute the momentum gradients is:

$$\nabla_n\left(\rho\vec{V}\right) = \Delta_{lr}\left(\rho\vec{V}\right) d^k \qquad (16)$$

where $\Delta_{lr} = (\cdot)_l - (\cdot)_r$. Experimenting with grid adaption for different problem types has shown that good results (i.e. shock , as well as boundary layer capturing) are obtained for k in the range:

$$0. \leq k \leq 0.4$$

The value for the error indicator is taken as the absolute value of the vector quantity given in Equation 16:

$$E = \left|\nabla_n(\rho\vec{V})\right| \qquad (17)$$

The error indicator threshold, above which the faces flagged for adaption, is determined based on its distribution density [13]. The threshold value is given by:

$$E_{th} = \overline{E} + \gamma\left(\sigma_E\right) \qquad (18)$$

where $\overline{E}$ and σ_E are the average value of the error indicator and its standard deviation. The coefficient multiplying the standard deviation, γ, is user defined.

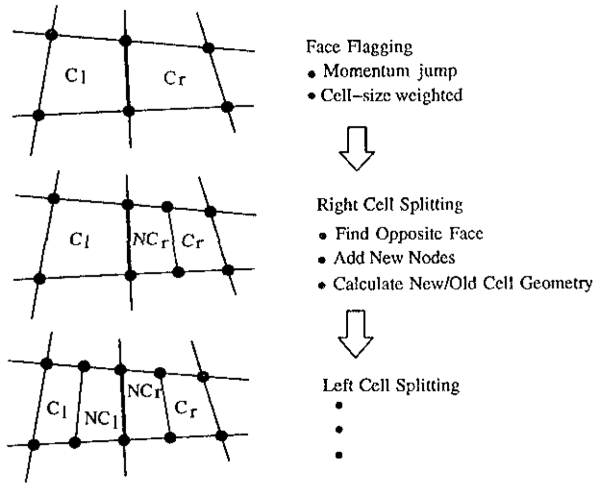

Figure 5: Basic cell refinement procedure

4.2 Adaption procedure

The basic procedure of the grid refinement is described in Figure 5. Each face which has error indicator above the threshold value causes its two corresponding cells (left and right) to be split into two "halves". As shown in Figure 5, first the "right" cell is considered, the opposite to the flagged face is identified and new nodes are added halfway between the nodes of the flagged face and the opposite. The "new" cell's geometry is calculated and the "old" one's is modified. The same procedure is then repeated for the "left" cell.

The scheme suggested here creates a directional adaption pattern. The refined cells get stretched in the direction normal to the flow gradients increasing the accuracy and the resolution of the solution.

Special care is taken, during the grid adaption, to maintain the smoothness of the grid. This is done by preventing the creation of multiply partial faces (MPFs). An example of such faces is given in Figure 6. Faces f1 trough f4, in this figure, are multiply partial when cell C1 in considered. Multiply partial faces make the grid unsmooth and increase the complexity of the adaption procedure.

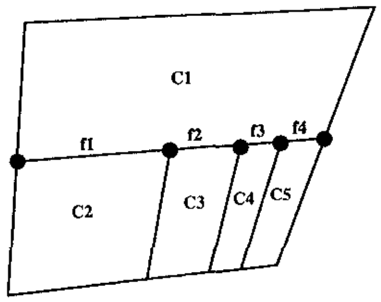

Figure 6: an example of multiply partial faces

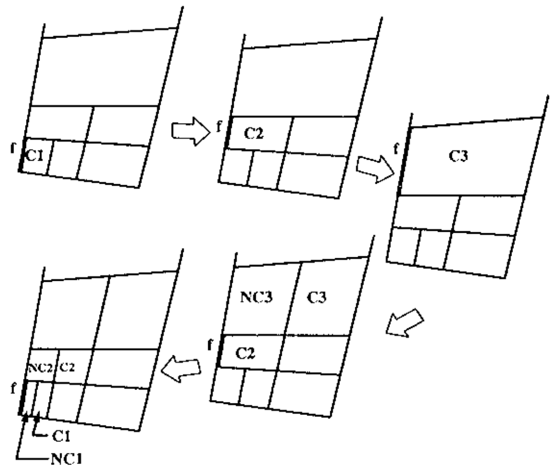

Figure 7: Schematic description of the adaption detour path

In order to avoid creation of MPFs the adaption procedure takes a "detour" whenever splitting a cell would result in MPF. A schematic description of the detour path is given in Figure 7.

An example of an adapted grid is given in Figure 8 which shows the adapted grid obtained after 900 time steps for supersonic flow (M=2.5) over a sharp wedge (Re = 10,000). As can be seen the grid is adapted around the oblique shock wave and the boundary layer. A closer look into the refined grid, Figure 8, shows the nonisotropy of the adaption method used here. While the refined boundary layer cells are highly stretched in the flow direction the shock wave cell pattern is aligned in the direction of the oblique shock.

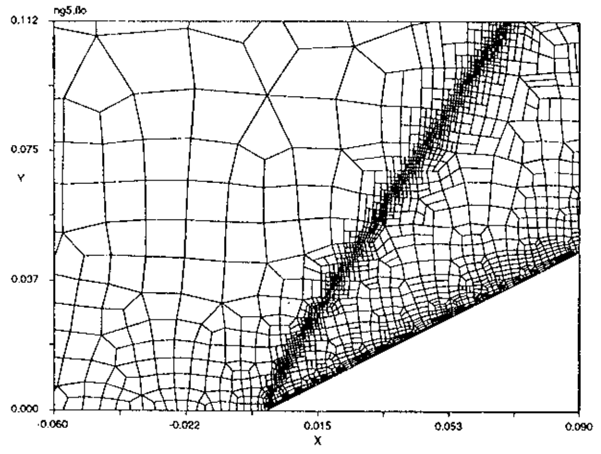

Figure 8: flow on a wedge M=2.5 - adapted grid

5 Results

5.1 Laminar boundary layer

This case was used to examine the interaction between the grid shape and the viscous terms of the governing equations.

Three level of grid fines have been used to perform grid convergence. These grids consist of cell sizes of 0.25 (grid1), 0.17 (grid2) and 0.125 (grid3) of the local thickness of the boundary layer at the location where the results are presented (x = 1m, 0.5m from the plate leading edge). Slices of the three grids are presented at Figure 9. The runs were performed at Mach=0.3, Re=2,000 which results in boundary layer thickness of approximately 0.1 at x=1.

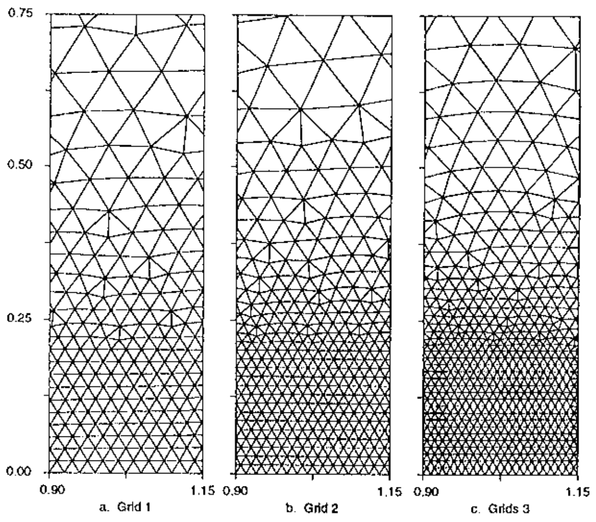

Figure 9: Slices of the three boundary layer grids

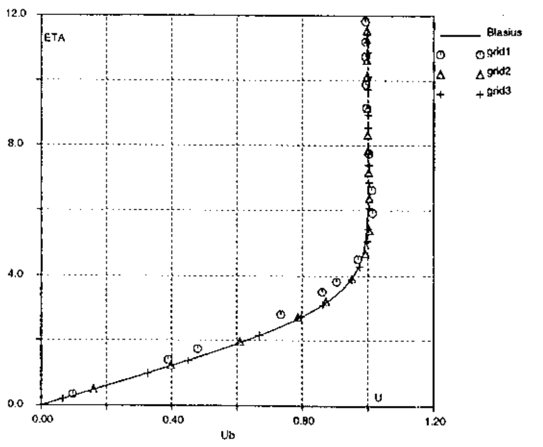

Figure 10: Computed nondimensional boundary layer velocity, as compare to the exact (Blasius) solution (M=0.3, Re=2000).

The computed nondimensional velocity in the three grids is presented in Figure 10 as a function of the nondimensional distance from the wall, η. As can be seen the results match the theory for grid 2 and 3. It should be noted that this excellent agreement is obtained with relatively few cells inside the boundary layer.

The results of the laminar boundary layer example also show that good evaluation of viscous flows is possible without using special grid geometry as was suggested is several reports ([14], [15],[16]).

5.2 Two dimensional airfoil

Transonic airfoil solutions have also been obtained using the RAE2822 airfoil. Figure 11 shows the pressure coefficient distribution on the airfoil surface at Mach=0.725, $\alpha = 2.92^0$. As can be seen the agreement between the experiment [17] and the present computational method is reasonable.

The shock wave location is obtained with good agreement to the experimental data. The differences between the computed results and the experiments (next to the lower side of the trailing edge and along the front part of the pressure side) are believed to stem from the lack of turbulence in the code.

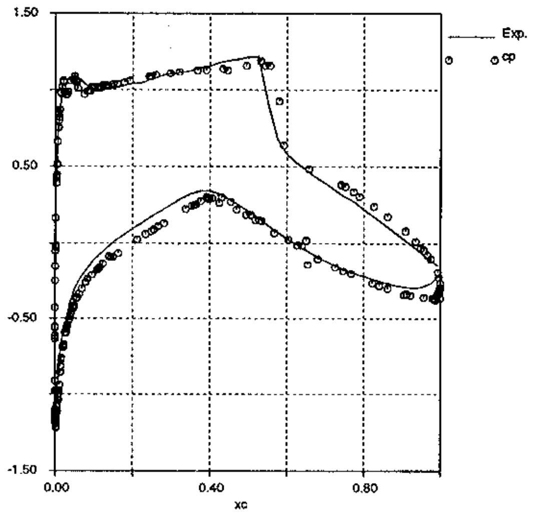

Figure 11: RAE2822 C_P distribution, $\alpha = 2.92^0$, Mach=.725, Re=5×10^6

The initial computational grid for this case consists of 5074 cells. During the run the grid is adapted seven times resulting in a final grid with 20,799 cells. The convergence history is presented in Figure 12. The adaption steps are evident in this figure by the relatively high values of residual arising from the fact that the flow field has to adjust itself to the newly adapted grid.

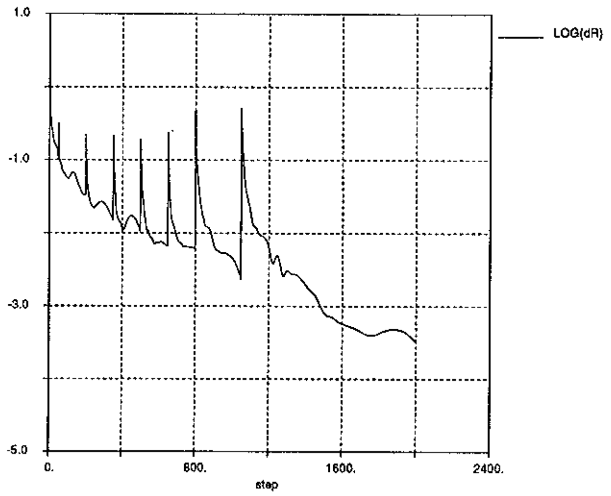

Figure 12: convergence history for RAE2822, M=.725, $\alpha = 2.92^0$

The adapted grid obtained in this computation is presented in Figure 13. The grid refinement in the high gradient regions is evident as well as the different natures of the refined regions: elongated cells inside the boundary layer, and isotropic cells near the leading edge.

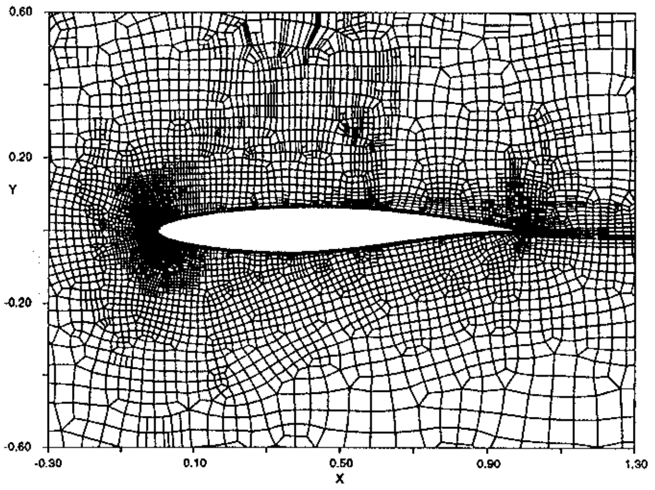

Figure 13: RAE2822 adapted grid

5.3 Three dimensional 76^0 sharp edge delta wing

The grid used in this case is tetrahedral, consisting of 11,000 cells and the simulated flow is subsonic (M=0.4). Figure 14 shows the normal force coefficient as a function of the angle of attack as compared to Polhamus' leading edge suction analogy [18]. As can be seen the normal force coefficients converge to a value close to the one predicted by Polhamus. Visualization of the

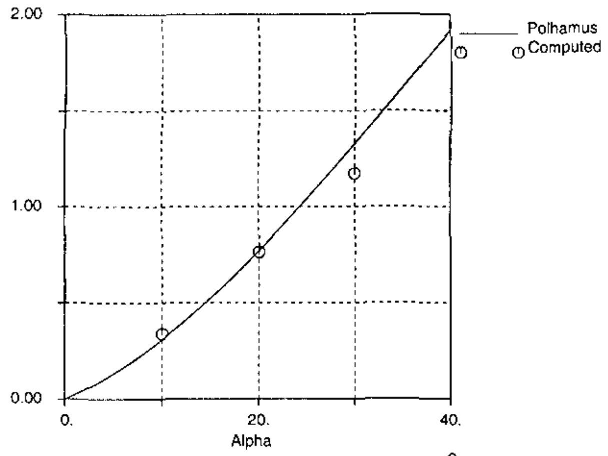

Figure 14: Normal force coefficient on a 76^0 delta wing

resultant flow field shows the well defined leading edge vortices in accordance with the experimental results.

6 Performance

Table 1 presents some of the performance characteristics obtained by the code on a 2048-node CM-200 machine (64 Weitek chips). The most striking data in this table is the large percentage of the total run time needed for inter processor communication (second row). The large communication time is expected to be reduced after the introduction of the more efficient gather/scatter library routines developed by Thinking Machines Corporation. Reduction of communication time is expected to have a significant effect on the overall performance.

Type of Run	Inviscid	Viscous	Viscous adaptive
mflops	33.5	26.5	22
%communication	77	82	69
flops/sec/cell	6600	5200	3000
μsec/cycle/cell	320	420	660

Table 1: Performance for various types of runs on the CM-200 machine

Another fact seen in Table 1, is the performance reduction as more tasks have to be carried out the (i.e. computation of viscous terms and adaption of the grid).

7 Summary and Conclusions

A robust three dimensional viscous flow solver has been developed for massively parallel machines. The spatial discretization in the code is unstructured and the grids introduced in this report were generated using the commercially available IDEAS package. The code uses a full fourth order Runge-Kutta scheme to enhance the convergence, as compared to the compact four stage scheme which is popular in time marching solvers.

A unique approach has been taken in grid adaption. The grid is refined in high momentum gradient regions and the refinement is being done in a direction normal to the local gradient by flagging the faces across which the gradient high highest and splitting the corresponding cells in "parallel" to the highlighted faces. The solid surface is reconstructed during the adaption by using second order interpolation between neighboring nodes whenever new surface node is added to the grid.

The results obtained so far show good agreement with analytic solution for flat plate boundary layer and experimental results over transonic airfoil (RAE2822). A resonable agreement with prediction method is also obtained for three dimensional flow over delta wing at high angles of attack.

Future runs are planned on more complex three dimensional configuration. These runs will be executed on the Thinking Machines CM5 due to large number of cell required in the more complex configurations.

References

[1] Long, L. N., Khan, M. M. S., Sharp, H. T. "Massively Parallel Three Dimensional Euler/Navier-Stokes Method" *AIAA Journal*, Vol. 29, No. 5, May 1991.

[2] Agarwal, R. K., "Parallel Computers and Large Problems in Industry", invited lecture, Computational Methods in Applied Science, Sept. 1992. Elsevier, 1992.

[3] Long, L. N., "Parallel Algorithms for Gas Dynamics", **Parallel Computational Fluid Dynamics '92**, Proc. Parallel CFD '92 Conference, New Brunswick, NJ, May 1992. North-Holland, 1993.

[4] Blake, K. R., Spragle, G. S. "Unstructured 3D Delauney Mesh Generation Applied to Planes, Trains and Automobiles", AIAA paper 93-0673, January, 1993.

[5] Pirzadeh, S. "Structured Background Grids for Generation of Unstructured Grids by Advancing-Front Method", *AIAA journal*, Vol 31, No.2, February 1993.

[6] Jameson, A., Baker, T. J. "Solution of Euler Equations for Complex Configurations",AIAA paper 83-1929-cp, 6th CFD conference, Danvers, Massachusetts, 1983.

[7] Jameson, A., "Transonic Flow Calculations", Princeton University, Princeton, NJ, MAE Report 1651, July 1983.

[8] Golub, Van Loan, "Matrix Computations", Second Edition, Algorithm 10.2.1 p. 523.

[9] Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", Journal of Computational Physics, Vol. 43, pp. 357 - 372 (1981).

[10] Gnoffo, P. A., "Application of Program LAURA to Three-Dimensional AOTV Flowfields", AIAA-86-0565, January 1986.

[11] Khan, M. M. S., "Adaptive Refinement of Tetrahedral Grids for Compressible Flow Computations on the Connection Machine", AIAA-91-0440, January, 1991.

[12] Rauch, R.D., Batina, J. T., Yang, H. T. Y. "Spatial Adaption of Unstructured Meshes for Unsteady Aerodynamic flow Computations" AIAA Journal, Vol 30, No. 5, May 1992.

[13] Kallinderis Y. G., Baron, J. R. "Adaption Methods for a New Navier-Stokes Algorithm", *AIAA journal,* Vol 27, No. 1, January 1989.

[14] Venkatakrishnan, V, Barth, T. J. "Applicatio of Direct Solvers to Unstructured Meshes for tha Euker and Navier-Stokes Equations Using Upwind Schemes", AIAA paper 89-0364, January 1989.

[15] Mavriplis, D. J., "Euler and Navier-Stokes Computations for Two-Dimwnsional Geometries Using Unstructured Meshes", ICASE Report No. 90-3, NASA CR-181977, 1990.

[16] Pirzadeh, S., "Unstructured Viscous Grid Generation by Advancing Layers Method", AIAA Paper 93-3453, August 1993.

[17] Cook, P. H., McDonald, M. A., Firmin, F. C. P "Airfoil RAE 2822 Pressure Distribution and Boundary Layer Measurements" AGARD AR 138, 1979.

[18] Polhamus, E. C., "Charts for Predicting the Subsonic Vortex Lift Characteristics of Arrow, Delta and Diamond Wings", NASA TN D-6243, 1971.