A PARALLEL ADAPTIVE FINITE ELEMENT EULER FLOW SOLVER FOR ROTARY WING AERODYNAMICS*

Carlo L. BOTTASSO†, Mark S. SHEPHARD‡

Rensselaer Rotorcraft Technology Center, and
Scientific Computation Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Abstract

We present a parallel adaptive finite element flow solver for rotary wing applications. The code adaptively refines and derefines the discretization for accurately resolving the different features of the flow, such as shocks and wakes. The code performs all the stages of the analysis —finite element solution, error indication, mesh adaptation— in a parallel MIMD environment making use of efficient scalable parallel algorithms. The main features of all the building blocks of the implemented approach are detailed and discussed in the paper. The capabilities and performance characteristics of the parallel automated adaptive procedure are demonstrated with regard to subsonic and transonic compressible flow problems. In particular, different hovering rotor problems are analyzed and the results are compared with experimental data showing good agreement.

Introduction

The aim of this paper is to present an adaptive procedure that the authors have recently developed for the automated aerodynamic analysis of helicopter rotors. Adaptive analyses on unstructured discretizations procedures represent an effective and accurate method to address complex physical phenomena, such as those that characterize rotorcraft systems. The problem of the accurate numerical simulation of these phenomena has recently stimulated a vigorous research effort in the scientific community, certainly prompted by the fact that rotor–body interactions, transonic effects, wake effects and blade stall, all have a major impact on the performance, stability and noise characteristics of helicopter rotors.

Such numerical simulations imply massive computations. Distributed memory parallel computers have recently been successfully employed for large–scale analysis of fluid flows [6][8]. These computers seem to offer the potential for satisfying the demands of high performance as well as providing large memories.

The effective use of the computational power of high performance MIMD machines requires the development of suitable techniques and implies a major restructuring of existing codes. It is then perfectly clear that the cost effective use of this new generation of computers requires the development of software tools of general applicability. In this work we report the development of a parallel adaptive code for rotating wing analysis that we have completed using a number of efficient software tools and algorithms for the parallel automated adaptive solution of PDE's on distributed memory computers that we and our Rensselaer colleagues have recently developed [4][15].

One of the most important characteristics and distinguishing features of the software here presented is that all the different phases of the analysis, namely the mesh partitioning, the finite element solution, the error indication, the mesh adaptation and the subsequent load balancing, are realized without leaving the parallel environment. In contrast with other procedures that perform only part of the analysis in parallel, as for example just the finite element solution phase, our approach has the advantage of making a better use of the power of a distributed memory architecture, leading to an integrated software environment, reducing the i/o and avoiding the bottlenecks that are always present when one tries to solve certain phases of the analysis in serial, especially when very large problems are addressed.

This integrated approach to the parallel adaptive solution of PDE's has lead us to select the message passing paradigm as our method of choice for the parallel programming. This is in contrast with the trend shown by some recent publications [6][7][8], where parallel finite element methodologies on fixed meshes have been developed based on data parallel techniques. In fact, we believe that the software development is more easily accomplished in a message passing programming model when one has to deal with adaptive strategies and mesh modification techniques. With the idea of developing a uniform software environment, we have used portable message passing protocols in each stage of the analysis. The implementation has been carried out using the message passing library standard MPI and it has been tested on IBM SP–1 and SP–2 systems.

In the first section of this work we present a stabilized finite element formulation which is valid for forward flight and for hovering rotor problems, as well as for general unsteady and steady compressible flow problems. The linear algebra is solved by means of a scalable implementation of the standard and matrix–free GMRES algorithms. Simple techniques are used for estimating regions of high error with the purpose of driving the adaptive procedures.

The second section briefly reviews the parallel mesh data

† Post–Doctoral Research Associate, Member AIAA.

‡ Director, Scientific Computation Research Center, Johnson Chair of Engineering, Associate Fellow AIAA.

structures that we have developed, the partitioning of the discretized computational domain and the parallel adaptation of the mesh.

A third section is devoted to the discussion of the treatment of the far–field and symmetry boundary conditions for a hovering rotor, which are fundamental for an accurate and efficient analysis of this class of problems.

The paper is concluded by a section dedicated to the analysis of the results gathered during a number of numerical experiments. The aim is here twofold: first, we show that using an adaptive methodology we can accurately numerically simulate complex engineering problems, such as the development of shock waves on the blades of hovering rotors in transonic conditions. Second, we address the problem of giving measures of efficiency and scalability of the parallel adaptive procedures that we have developed, making use of a classical problem in transonic CFD.

Finite Element Formulation

The initial/boundary value problem can be expressed by means of the Euler equations in quasi–linear form as

$$\boldsymbol{U}_{,t} + \boldsymbol{A}_i \cdot \boldsymbol{U}_{,i} = \boldsymbol{E}, \quad (i = 1, \ldots, n_{sd}) \tag{1}$$

plus well posed initial and boundary conditions. In equation (1), n_{sd} is the number of space dimensions, while $\boldsymbol{U} = \rho\,(1, u_1, u_2, u_3, e)$ are the conservative variables, $\boldsymbol{A}_i \cdot \boldsymbol{U}_{,i} = \boldsymbol{F}_i = \rho\, u_i(1, u_1, u_2, u_3, e) + p\,(0, \delta_{1i}, \delta_{2i}, \delta_{3i}, u_i)$ is the Euler flux, and $\boldsymbol{E} = \rho\,(0, b_1, b_2, b_3, b_i u_i + r)$ is the source vector. In the previous expressions, ρ is the density, $\boldsymbol{u} = (u_1, u_2, u_3)$ is the velocity vector, e is the total energy, p is the pressure, δ_{ij} is the Kronecker delta, $\boldsymbol{b} = (b_1, b_2, b_3)$ is the body force vector per unit mass and r is the heat supply per unit mass.

The Time–Discontinuous Galerkin Least–Squares finite element method is used in this effort [12][13]. The TDG/LS is developed starting from the symmetric form of the Euler equations expressed in terms of the entropy variables $\boldsymbol{V}$ and it is based upon the simultaneous discretization of the space–time computational domain. A least–squares operator and a discontinuity capturing term are added to the formulation for improving stability without sacrificing accuracy. The TDG/LS finite element method takes the form

$$\int_{Q_n} \left(-\boldsymbol{W}^h_{,t} \cdot \boldsymbol{U}(\boldsymbol{V}^h) - \boldsymbol{W}^h_{,i} \cdot \boldsymbol{F}_i(\boldsymbol{V}^h) + \boldsymbol{W}^h \cdot \boldsymbol{E}(\boldsymbol{V}^h) \right) \mathrm{d}Q$$

$$+ \int_{\mathcal{D}(t_{n+1})} \boldsymbol{W}^{h^-} \cdot \boldsymbol{U}(\boldsymbol{V}^{h^-})\,\mathrm{d}\mathcal{D} - \int_{\mathcal{D}(t_n)} \boldsymbol{W}^{h^+} \cdot \boldsymbol{U}(\boldsymbol{V}^{h^-})\,\mathrm{d}\mathcal{D}$$

$$+ \int_{P_n} \boldsymbol{W}^h\, \boldsymbol{F}_i(\boldsymbol{V}^h) \cdot \mathrm{d}\boldsymbol{P}$$

$$+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \left(\mathcal{L}\boldsymbol{W}^h \right) \cdot \tau \left(\mathcal{L}\boldsymbol{V}^h \right) \mathrm{d}Q$$

$$+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \nu^h \hat{\nabla}_\xi \boldsymbol{W}^h \cdot \mathrm{diag}\,[\tilde{\boldsymbol{A}}_0] \hat{\nabla}_\xi \boldsymbol{V}^h\, \mathrm{d}Q = 0. \tag{2}$$

Integration is performed over the space–time slab Q_n, the evolving spatial domain $\mathcal{D}(t)$ of boundary $\Gamma(t)$ and the surface P_n described by $\Gamma(t)$ as it traverses the time interval $I_n =]t_n, t_{n+1}[$. $\boldsymbol{W}^h$ and $\boldsymbol{V}^h$ are suitable spaces for test and trial functions, while τ and ν^h are appropriate stabilization parameters. $\tilde{\boldsymbol{A}}_0 = \partial \boldsymbol{U}/\partial \boldsymbol{V}$ is the metric tensor of the transformation from conservation to entropy variables. Refer to [12] for additional details on the TDG/LS finite element formulation.

We have implemented two different three dimensional space–time finite elements. The first is based on a constant in time interpolation, and, having low order of time accuracy but good stability properties, it is well suited for solving steady problems using a local time stepping strategy. The second makes use of linear–in–time basis functions and, exhibiting a higher order temporal accuracy, is well suited for addressing unsteady problems, such as for example forward flight. In this cases, moving boundaries are handled by means of the space–time deformed element technique [17].

For efficiently solving hover problems we have also developed a formulation starting from the Euler equations written in a rotating frame. This allows to treat a hovering rotor as a steady problem assuming that the unsteadiness in the wake can be neglected, thus allowing the use of the less computationally expensive constant–in–time formulation.

Assuming that the axis of rotation is coincident with the z axis and that the angular velocity is Ω, the compressible Euler equations in a rotating frame can be expressed in terms of the absolute flow variables $\boldsymbol{U}$ as

$$\boldsymbol{U}_{,t} + (\boldsymbol{A}_i - v_i \boldsymbol{I}) \cdot \boldsymbol{U}_{,i} = \boldsymbol{E} + \boldsymbol{E}_G, \tag{3}$$

where $v_1 = -\Omega y$, $v_2 = \Omega x$, $v_3 = 0$ and $\boldsymbol{E}_G$ can be defined as

$$\boldsymbol{E}_G = \boldsymbol{C}\boldsymbol{U} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Omega & 0 & 0 \\ 0 & -\Omega & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{U},$$

or, in terms of entropy variables, $\boldsymbol{E}_G = \tilde{\boldsymbol{C}}\boldsymbol{V}$, $\tilde{\boldsymbol{C}} = -\rho T\,\boldsymbol{C}$. Clearly, by the nature of the gyroscopic terms, we have that $\boldsymbol{C}^T = -\boldsymbol{C}$.

We remark that the rotating frame formulation of the compressible Euler equations in terms of absolute flow variables is formally equivalent to a change of variables (modification of the jacobians $\boldsymbol{A}_i$ into $\boldsymbol{A}_i - v_i \boldsymbol{I}$) plus the introduction of a source term $\boldsymbol{E}_G$.

From the formulation expressed in equation (3), a TDG/LS finite element formulation can be easily constructed along the lines of equation (2). In an inertial frame, a definition of τ that results in full upwinding on each mode of the system [12] is given by

$$\tau = \tilde{\boldsymbol{A}}_0^{-1} \left(\hat{\boldsymbol{A}}_\xi^T \,\mathrm{diag}(\tilde{\boldsymbol{A}}_0^{-1}) \hat{\boldsymbol{A}}_\xi \tilde{\boldsymbol{A}}_0^{-1} \right)^{-\frac{1}{2}}, \tag{4}$$

where

$$\hat{\boldsymbol{A}}_\xi = \left(\frac{\partial \xi_0}{\partial x_0} \tilde{\boldsymbol{A}}_0, \frac{\partial \xi_1}{\partial x_i} \tilde{\boldsymbol{A}}_i, \ldots, \frac{\partial \xi_{n_{sd}}}{\partial x_i} \tilde{\boldsymbol{A}}_i \right)$$

and ξ_i are the local element coordinates, x_0 and ξ_0 referring to the time dimension. In a rotating frame, we redefine $\hat{\boldsymbol{A}}_\xi$ as

$$\hat{\boldsymbol{A}}_\xi = \left(\tilde{\boldsymbol{C}}, \frac{\partial \xi_0}{\partial x_0}\tilde{\boldsymbol{A}}_0, \frac{\partial \xi_1}{\partial x_i}(\tilde{\boldsymbol{A}}_i - v_i\tilde{\boldsymbol{A}}_0), \ldots, \frac{\partial \xi_{n_{sd}}}{\partial x_i}(\tilde{\boldsymbol{A}}_i - v_i\tilde{\boldsymbol{A}}_0)\right).$$

Solution to (4) can be obtained based upon the eigenproblem

$$\left(\hat{\boldsymbol{A}}_\xi^T \operatorname{diag}(\tilde{\boldsymbol{A}}_0^{-1})\hat{\boldsymbol{A}}_\xi - \lambda^2\tilde{\boldsymbol{A}}_0^{-1}\right) \cdot \boldsymbol{T}_i = 0. \qquad (5)$$

The eigenproblem is simplified by means of a similarity transformation $\boldsymbol{S}$ that diagonalizes $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ and symmetrizes $\boldsymbol{A}_3$ [18]. However, the term arising from $\boldsymbol{E}_G$ remains non–symmetric. We have implemented both the non–symmetric and a symmetric form obtained by dropping the contribution of $\boldsymbol{E}_G$ from (5) and have found that for the hovering rotors that we have studied in our numerical simulations, the symmetric form gives results indistinguishable from those of the non–symmetric form at a lower computational cost.

Discretization of the weak form implied by the TDG/LS method leads to a non–linear discrete problem, which is solved iteratively using a quasi–Newton approach. At each Newton iteration, a non–symmetric linear system of equations is solved using the GMRES algorithm. We have developed scalable parallel implementations of the preconditioned GMRES algorithm and of its matrix–free version [6]. This latter algorithm approximates the matrix–vector products with a finite difference stencil with the advantage of avoiding the storage of the tangent matrix, thus realizing a substantial saving of computer memory at the cost of additional on–processor computations. Preconditioning is achieved by means of a nodal block–diagonal scaling transformation.

In this work we have implemented a simple error indicator based on the norm of the gradient of the flow variables and a slightly more sophisticated one [10] for linear elements which takes the basic form

$$e_i = \frac{h^2 \mid \text{Second Derivative of } \Psi \mid}{h \mid \text{First Derivative of } \Psi \mid + \varepsilon \mid \text{Mean Value of } \Psi \mid}, \qquad (6)$$

where e_i is the error indicated at node i, h is a mesh size parameter, Ψ is the solution variable being monitored, ε is a tuning parameter. The second derivative of Ψ is computed using a variational recovery technique.

The edge values of the error indicator are computed by averaging the corresponding two nodal values. These edge-wise error indicator values are then used for driving the mesh adaptation procedure. Appropriate thresholds are supplied for the error values, so that the edge is refined if the error is higher than the maximum threshold, while the edge is collapsed if the error is the less than the minimum threshold.

Parallel Mesh Data Structures, Mesh Partitioning, Load Balancing and Mesh Adaptive Procedures

The parallel mesh data structures developed at Rensselaer for supporting the solution of PDE's, the partitioning of the discretized computational domain and the parallel adaptation of it, have been discussed in [1][2][4][15]. In the following, we briefly mention the most important characteristics and ideas behind the implemented approaches.

The data structures used in a parallel adaptive finite element solver must provide fast query and update of partition boundary information. Besides the queries, update procedures must be available to the refinement/coarsening and load balancing components of the parallel finite element solver. Efficient computation requires updated entities be inserted or deleted from the partition boundary within constant time, or at most time proportional to the number of adjacent processors. To implement these fast query and update routines, we have made use of a topological entity hierarchy data structure [1], which provides a two-way link between the mesh entities of consecutive order, i.e. regions, faces, edges and vertices. From this hierarchy, any entity adjacency relationship can be derived by local traversals. The entities on the partition boundary are augmented with links which point to the location of the corresponding entity on the neighboring processor. This data structure is shared by all the building blocks of the code —flow solver, adaptation, balancing and partitioning algorithms— achieving in this way a uniform software environment.

The parallel adaptive analysis begins with the partitioning of the initial mesh which is performed using the orthogonal RB algorithm or its variant, moment of inertia RB (IRB). The whole mesh is first loaded into one processor and then recursively split in half and sent to other processors in parallel.

The mesh is then adapted based on the information provided by the error indication performed on the converged finite element solution. The mesh adaptive algorithm combines derefinement, refinement and triangulation optimization using local retriangulations [5]. The derefinement step is based on an edge collapsing technique. This approach does not require storage of any history information and it is therefore not dependent on the refinement procedure.

The implemented refinement algorithm makes use of subdivision patterns. All possible subdivision patterns have been considered and implemented to allow for speed and anihilate possible over-refinement.

The adaptive procedure includes also a triangulation optimization scheme, which is particularly important when the snapping of refinement vertices on curved model boundaries can potentially create invalid or poorly shaped elements. The idea is to iteratively consider the local retriangulation of simple and well defined polyhedra.

As all the other building blocks of the code here discussed, also the mesh adaptation algorithm has been completely parallelized [15].

In a parallel distributed memory environment, adaptivity performed on the mesh in general destroys load balancing. Therefore procedures are needed to redistribute the mesh in order to achieve a balanced situation. With regard to this problem, we have implemented two techniques. The first performs a parallel repartition of an already distributed mesh using the IRB algorithm [15]. The second is a load balancing scheme that iteratively migrates elements from heavily loaded to less loaded processors [4]. To decide which processors should be involved in load migration, we use a heuristic based on the Leiss and Reddy approach [9] of letting each processor request load from a heavily loaded neighbor to even out the load imbalance. Once the directions of load migration have been calculated, the elements on the partition boundary are migrated slice by slice, each slice of elements forming a peeling of the partition boundary.

Both these techniques present interesting characteristics and each one has its own advantages and disadvantages. We are currently in the process of evaluating these two approaches, as well as implementing improved migration techniques for achieving better quality of the partitions.

Boundary Conditions for Hovering Rotors

The imposition of the correct far–field boundary conditions is a critical issue in the analysis of hovering rotors, when one wants to give an accurate representation of the hovering conditions within a finite computational domain. For determining the inflow/outflow far–field conditions we have adopted the methodology suggested by Srinivasan *et al.* [16], where the 1–D helicopter momentum theory is used for determining the outflow velocity due to the rotor wake system. The inflow velocities at the remaining portion of the far–field are determined considering the rotor as a point sink of mass, for achieving conservation of mass and momentum within the computational domain.

Another important condition that must be considered for the efficient simulation of hovering rotors is the periodicity of the flowfield. This allows to consider a reduced computational domain given by the angle of periodicity $\psi = 2\pi/n_b$, n_b being the number of rotor blades.

The introduction of the periodicity conditions in the rotating wing flow solver has been implemented treating them as linear 2–point constraints applied via transformation as part of the assembly process. This approach has the double advantage of being easily parallelizable and of avoiding the introduction of Lagrange multipliers. On the other hand, it requires the mesh discretizations on the two symmetric faces of the computational domain to match on a vertex by vertex basis. Since this is not directly obtainable with the currently used unstructured mesh generator, a mesh matching technique has been developed for appropriately modifying an existing discretization.

In order to simplify the discussion, define one of the symmetric model faces as "master" and the other as "slave". The face discretization of the slave model face is deleted from the mesh, together with all the mesh entities connected to it. The mesh discretization of the master model face is then rotated of the symmetry angle ψ about the axis of rotation and copied onto the slave model face, yielding the required matching face discretizations. The matching procedure is then completed filling the gap between the new discretized slave face and the rest of the mesh using a face removal technique followed by smoothing and mesh optimization.

The imposition of the constraints can be formalized in the following manner. Consider the partition of the unknowns $\boldsymbol{V}$ in internal ($\boldsymbol{V}_i$), master ($\boldsymbol{V}_m$) and slave ($\boldsymbol{V}_s$), as

$$\boldsymbol{V} = (\boldsymbol{V}_i, \boldsymbol{V}_m, \boldsymbol{V}_s).$$

The slave unknowns $\boldsymbol{V}_s$ can be expressed symbolically as functions of the master unknowns $\boldsymbol{V}_m$ as

$$\boldsymbol{V}_s = \boldsymbol{G} \cdot \boldsymbol{V}_m$$

or, for the j–th master–slave pair of nodes as

$$\boldsymbol{V}_s^j = \boldsymbol{G}^j \cdot \boldsymbol{V}_m^j,$$

where

$$\boldsymbol{G}^j = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \boldsymbol{R} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$\boldsymbol{R}$ being the rotation tensor associated with the rotation of the symmetry angle ψ about the axis of rotation.

The minimal set of unknowns $\bar{\boldsymbol{V}} = (\boldsymbol{V}_i, \boldsymbol{V}_m)$ is related to the redundant set $\boldsymbol{V}$ by

$$\boldsymbol{V} = \boldsymbol{\Gamma} \cdot \bar{\boldsymbol{V}} = \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \\ 0 & \boldsymbol{G} \end{bmatrix} \cdot \bar{\boldsymbol{V}}.$$

The unconstrained linearized discrete equations of motion read

$$\boldsymbol{J} \cdot \Delta \boldsymbol{V} = \boldsymbol{r},$$

where $\boldsymbol{J}$ is the tangent matrix and $\boldsymbol{r}$ is the residual vector. Applying the transformation $\boldsymbol{\Gamma}$ to the unconstrained system yields the constrained reduced system

$$\boldsymbol{\Gamma}^T \boldsymbol{J} \boldsymbol{\Gamma} \cdot \Delta \bar{\boldsymbol{V}} = \boldsymbol{\Gamma}^T \cdot \boldsymbol{r}. \tag{7}$$

Refer to [14] for implementational details of this technique.

Numerical Experiments

In this section we present results gathered during a number of numerical experiments. The goal is to show the effectiveness of the proposed parallel adaptive automated procedure, both in terms of quality of the aerodynamic data and in terms of numerical performance.

Subsonic and Transonic Hovering Rotors

Caradonna and Tung [3] have experimentally investigated a model helicopter rotor in several subsonic and transonic hovering conditions. These experimental tests have been extensively used for validating CFD codes for rotating wing analysis. The experimental setup was composed of a two–bladed rotor mounted on a tall column containing the drive shaft. The blades had rectangular planform, square tips and no twist or taper, made use of NACA0012 airfoil sections and had an aspect ratio equal to six.

Figure (1) shows the experimental and numerical values of the pressure coefficients at different span locations for three subsonic test cases investigated by Caradonna and Tung, namely $\theta_c = 0^o$ and $M_t = 0.520$, $\theta_c = 5^o$ and $M_t = 0.434$, $\theta_c = 8^o$ and $M_t = 0.439$. The agreement with the experimental data is good at all locations, included the section close to the tip. Only two pressure distributions are presented for each case for space limitations, however similar correlation with the experimental data was observed at all the available locations. Relatively crude meshes have been employed for all the three test cases, with the coarsest mesh of only 101,000 tetrahedra being used for the $\theta_c = 0^o$ case, and the finest of 152,867 tetrahedra for the $\theta_c = 8^o$ test problem.

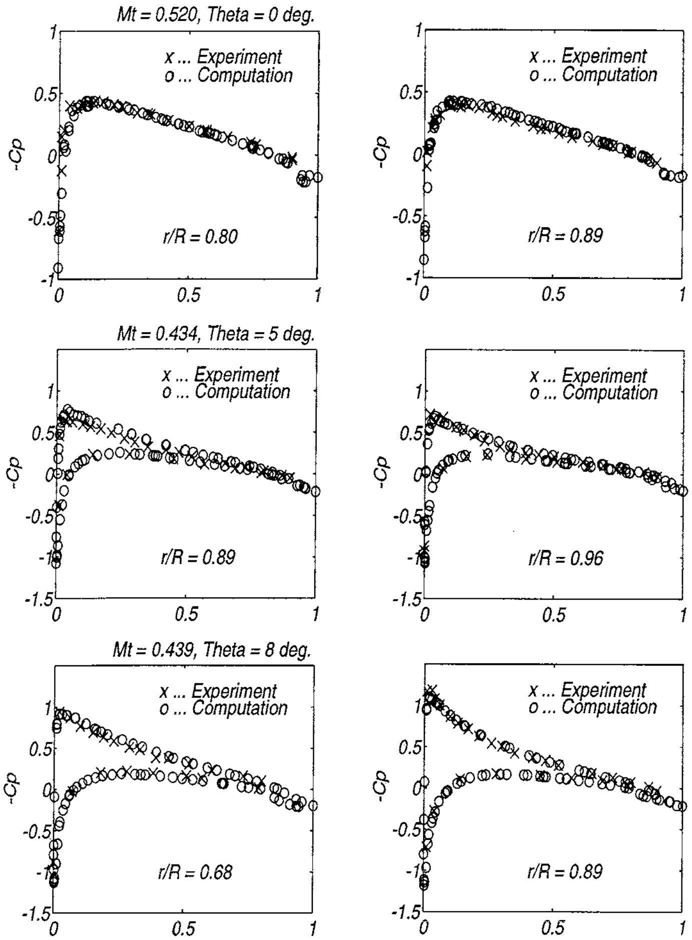

Figure 1: Computed and experimental pressure coefficients on the blade at different span locations, for the three subsonic cases $\theta_c = 0^o$, $M_t = 0.520$; $\theta_c = 5^o$, $M_t = 0.434$; $\theta_c = 8^o$, $M_t = 0.439$.

The analysis was performed on 32 processing nodes of an IBM SP–2. Reduced integration was used for the interior elements for lowering the computational cost, while full integration was used at the boundary elements for better resolution of the airloads, especially at the trailing edge of the blade. The GMRES algorithm with block–diagonal preconditioning was employed, yielding an average number of GMRES iterations to convergence of about 10. The analysis was advanced in time using one single Newton iteration per time step and a local time stepping strategy denoted by CFL numbers ranging from 10 at the beginning of the simulation to 20 towards convergence, yielding a reduction in the energy norm of the residual of almost four orders of magnitude in 50 to 60 time steps. The symmetric form of the least–squares stabilization was employed, and the discontinuity capturing operator was not activated.

Figure (2) shows the experimental and numerical values of the pressure coefficients for a transonic case denoted by $\theta_c = 8^o$ and $M_t = 0.877$. The first two plots of Figure (2) present the pressure distributions obtained using an initial crude grid consisting of 142,193 tetrahedra. Three levels of adaptivity were applied to this grid in order to obtain a sharper resolution of the tip shock, yielding a final mesh characterized by 262,556 tetrahedra. The pressure distributions obtained with the adapted grid are shown in the third and fourth plots of the same picture. Note that the smearing present in the first two plots and due to the numerical viscosity introduced in the formulation with the purpose of stabilizing it, has disappeared. Consistently with the nature of the Euler equations, the shocks appear as jumps and are resolved in only one or two elements. Note also the appearance of the analytically predicted overshoot just aft of the shock which is typical of the transonic Euler solutions.

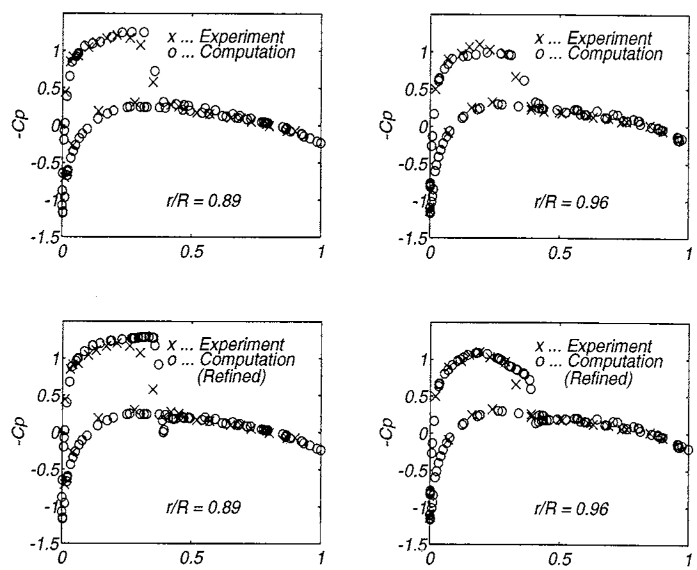

Figure 2: Computed and experimental pressure coefficients on the blade, at two different span locations close to the tip, $\theta_c = 8^o$, $M_t = 0.877$. Top two plots: initial coarse 142,193 tetrahedron grid. Bottom two plots: adapted (three levels) final 262,556 tetrahedron grid.

The effect of the adaptation of the mesh on the resolution of the shock is clearly demonstrated in Figure (3), where the density isocontour plots at the upper tip surface are presented for the initial and adapted meshes. The effect noted in Figure (2) can be more fully appreciated here.

The parallel adaptive analysis was conducted on 32 processing nodes with the GMRES algorithm, using once again reduced integration for the interior elements and full integration at the boundary elements. The symmetric form of the least–squares stabilization was employed, together with the discontinuity capturing term for improved shock confinement. After partitioning of the initial coarse mesh using the IRB algorithm, the simulation was performed for 60 implicit time steps with CFL condition equal to 10 in the initial 20 steps and equal to 15 for the remaining steps. The results

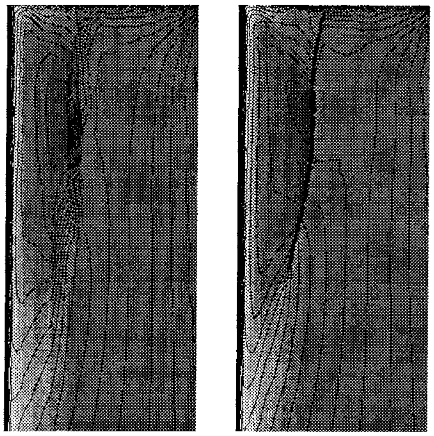

Figure 3: Density isocontour plots on the upper surface of the blade tip, $\theta_c = 8^o$, $M_t = 0.877$. At left: initial coarse grid. At right: final adapted grid.

gathered at convergence were used for computing an error indicator based on density and Mach number, which was employed for driving the parallel adaptation of the mesh. For the new vertices created by the adaptation process, the solution was projected from the coarser mesh using simple edge interpolation. The solution obtained in this way was used for restarting the analysis, which was advanced for 60 time steps with a CFL number of 15. Similarly, a second adaptation was performed, yielding the final mesh for which other 40 time steps were performed at a CFL of 20, until convergence in the energy norm of the residual. The average number of GMRES cycles per time step throughout the analysis was 8.

Figure (4) shows the mesh at the upper face of the blade tip, before and after refinement. The different grey levels indicate the different subdomains, i.e. elements assigned to the same processing node are denoted by the same level of grey. Note the change in the shape of the partitions from the initial to the final mesh, change generated by the mesh migration procedure for re–balancing the load after the refinement procedure has modified the discretization. Note also how the mesh nicely follows the shock.

Parallel–Adaptive Performance Results

The evaluation of the efficiency and performance of a parallel adaptive analysis is a task complicated by the numerous aspects that must be considered. In the following we will try to address at least some of them with the help of a classical problem in CFD, namely that of the ONERA M6 wing in transonic flight, that we have used in the early stages of development of our code for validation purposes. This wing has been studied experimentally by Schmitt and Charpin [11] and it has been employed by numerous researchers for validating both structured and unstructured flow solvers. The wing is characterized by an aspect ratio of 3.8, a leading edge sweep angle of 30^o, and a taper ratio of 0.56. The airfoil section is an ONERA D symmetric section with 10% maximum

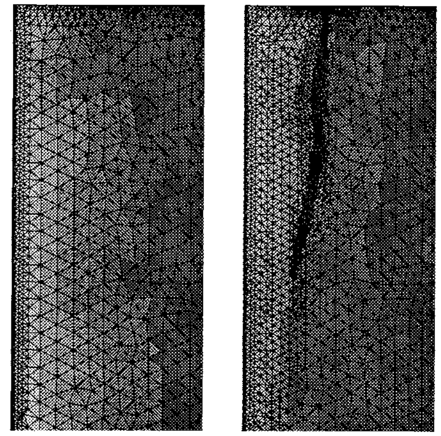

Figure 4: Meshes with partitions on the upper surface of the blade tip, $\theta_c = 8^o$, $M_t = 0.877$. At left: initial coarse grid with IRB partitions. At right: final adapted grid with partitions obtained by migration.

thickness–to–cord ratio.

We consider a steady flow problem characterized by an angle of attack $\alpha = 3.06^o$ and a value of $M = 0.8395$ for the freestream Mach number. In such conditions, the flow pattern around the wing is characterized by a complicated double–lambda shock on the upper surface of the wing with two triple points.

We first address the scalability of the parallel solver on a fixed mesh, i.e. we analyze the speed–ups attained by the code using one single mesh and varying the number of processing nodes. This is a classical measure of efficiency, and it is important to show that the implemented procedure performs well with respect to it before measuring other properties that are more pertinent to an adaptive analysis.

The simulation was performed using a mesh consisting of 128,172 tetrahedra, using the matrix–free GMRES algorithm with reduced integration of the interior elements and full integration of the boundary elements. A local time stepping strategy was employed with one single Newton iteration per time step, using a CFL condition of 5 in the first 20 time steps and a CFL equal to 10 for other 80 time steps, attaining a drop in the residual of three orders of magnitude. The mesh was partitioned using a parallel implementation of the IRB algorithm. The time for partitioning, even if small when compared with the time needed for achieving convergence in the finite element analysis, is not considered in the following. The analysis was run on 4, 8, 16, 32, 64, 128 processors of an IBM SP–2 and the results are presented in Figure (5) in terms of the inverse of the wall clock time versus the number of processing nodes. The highly linear behavior of the parallel algorithm shows the excellent characteristics of scalability of the code.

The same problem was then adaptively solved in order to more accurately resolve the complicated features of the flow. An initial coarse mesh of 85,567 tetrahedra was partitioned with the IRB algorithm on 32 processing nodes and the anal-

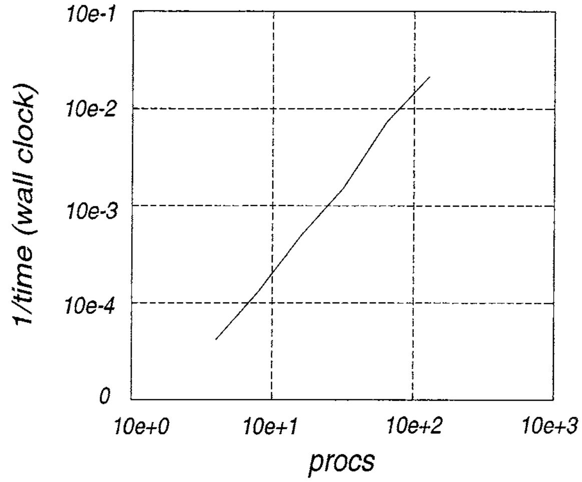

Figure 5: Parallel efficiency evaluated at fixed mesh for the ONERA M6 wing in transonic flight. 128,172 tetrahedra, IRB partitions.

ysis was carried on to convergence as previously explained. The results obtained were then used for computing an error indicator based on density and Mach number, which was employed for performing a first level of refinement, bringing the mesh to 131,000 tetrahedra. The solution was projected on the new vertices using a simple edge interpolation technique, and the analysis was then performed on the refined mesh for 80 time steps at a CFL number of 10. Similarly, other two levels of refinement followed by subsequent analysis were performed, obtaining an intermediate 223,499 tetrahedron mesh and a final 388,837 tetrahedron mesh.

Figure (6) shows the density isocontour plots on the upper surface of the wing corresponding to the initial and the final mesh discretizations. Note that the forward shock is barely visible in the results obtained with the initial coarse mesh, the aft shock presents significant smearing and the lambda shock located at the tip of the wing is not resolved. As expected, considerable improvement in the resolution of the shocks can be observed when mesh adaptation is employed.

Figure (7) shows the initial and final meshes. Once again, elements assigned to the same subdomains are denoted by the same grey level. For the final mesh, the partitions shown are those obtained with the iterative load balancing algorithm.

The fact that the analysis is conducted in parallel doesn't modify the convergence characteristics of a classical h refinement technique, such as the one here considered. However, while in a serial environment essentially only the accuracy of the solution versus the size of the problem and its computational cost enter into the picture, in a parallel environment other factors must be considered. In particular, we consider here the evolution during the analysis of two fundamental parameters: *(i)* the surface–to–volume ratio for the subdomains, *(ii)* the number of neighbors of each subdomain. The first of these two parameters essentially dominates the volume of communication in terms of the size of the messages to exchange, while the second parameter dominates the number

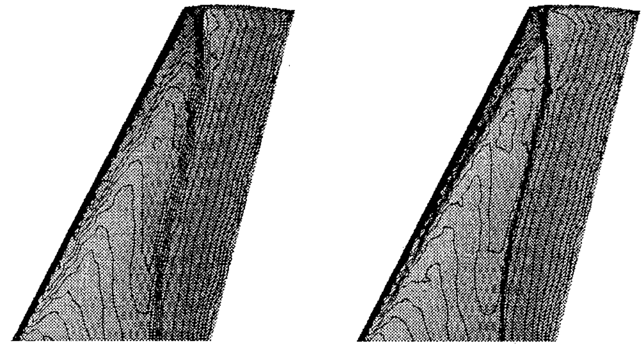

Figure 6: ONERA M6 wing in transonic flight, $\alpha = 3.06^o$, $M = 0.8395$. Density isocontour plots for the initial and final meshes.

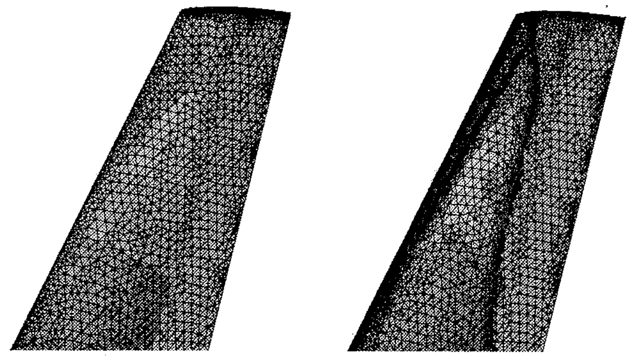

Figure 7: ONERA M6 wing in transonic flight, $\alpha = 3.06^o$, $M = 0.8395$. Initial and final meshes. Different grey levels indicate processor assignment (iterative load balancing partitions).

of messages that each processor must send and receive.

In a parallel adaptive environment, the issue is then: given certain repartitioning algorithms, which is the quality of the partitions that they produce compared to their relative cost? It is well known that certain classes of partitioning algorithms, such as the Spectral Bisection method, produce very high quality partitions. However, the cost associated with spectrally bisecting increasingly larger meshes during an adaptive analysis would be prohibitive. Therefore in this work we consider two relatively low cost approaches to the problem, the previously mentioned parallel IRB repartitioning and the iterative load migration scheme.

Two distinct runs were made, the only difference between them being the repartitioning strategy adopted. In both cases, all the stages of the analysis —initial IRB partitioning, flow solution, error sensing, adaptation and load balancing— were performed automatically in parallel on 32 processing nodes, i.e. without ever leaving the parallel environment. The load balancing algorithm was activated three times during the adaptation of each of the meshes, after the refinement, after the snapping of the newly generated vertices to the curved boundaries of the model and after the local retriangulation[§]. At every call, the algorithm was requested to perform only approximately eight migration iterations, yielding a maximum out of balance number of elements per processing node equal to one at the end of each refinement level. This strategy allows better efficiency of the various stages of the adaptive algorithm that can then operate on balanced or nearly balanced meshes. This "incremental" rebalancing capability represents an advantage of the iterative load balancing scheme over other algorithms. The parallel repartitioning algorithm was instead activated just once at the end of each adaptive step.

The meshes obtained during the two previously mentioned parallel adaptive simulations of the ONERA M6 wing were analyzed for gathering data on the overall performance of the analysis. Figure (8) reports plots of the boundary faces and neighbor statistics. The quantities plotted are defined as:

(i) Surface–to–volume measures:

$$S_{\max} = \max_i (\text{Boundary Faces}_i/\text{Faces}_i),$$
$$S_{\text{glob}} = \text{Boundary Faces}/\text{Faces}.$$

(ii) Neighbor measures:

$$N_{\max} = \max_i (\text{Neighbors}_i/(\text{Procs} - 1)),$$
$$N_{\text{avrg}} = (\textstyle\sum_i \text{Neighbors}_i/(\text{Procs} - 1))/\text{Procs}.$$

All these quantities are reported in Figure (8) versus the number of tetrahedra in the mesh at a certain adaptive level normalized by the number of tetrahedra in the initial mesh. The solid line represents the values of the parameters obtained for the parallel adaptive analysis where the iterative mesh migration procedures were employed. The dashed line corresponds to the parallel adaptive analysis where the refined meshes were repartitioned after each adaptive step using the parallel IRB algorithm.

From the analysis of the first two plots at the top of Figure (8), it is clear that the migration procedures implemented in this work control very effectively the surface–to–volume ratios, which in fact remain constant and fairly similar to the ones obtained with the IRB partitioning for the whole simulation. On the other hand, the second two plots of the same figure show that the number of neighbors of each subdomain tends to increase with the number of adaptive steps performed. A more detailed analysis shows that in general each subdomain is connected by a significant amount of mesh entities (vertices, faces, edges) only with a reduced number of neighbors, while it shares a very limited numbers of mesh entities with the other neighbors. We are investigating ways of removing such small contact area interconnections, in order to achieve a better control on the number of neighbors.

The different partition statistics provided by the two rebalancing algorithms and shown in the previous figure clearly

[§]We remark that in the current implementation, also snapping can cause load imbalance since it makes use of local triangulation.

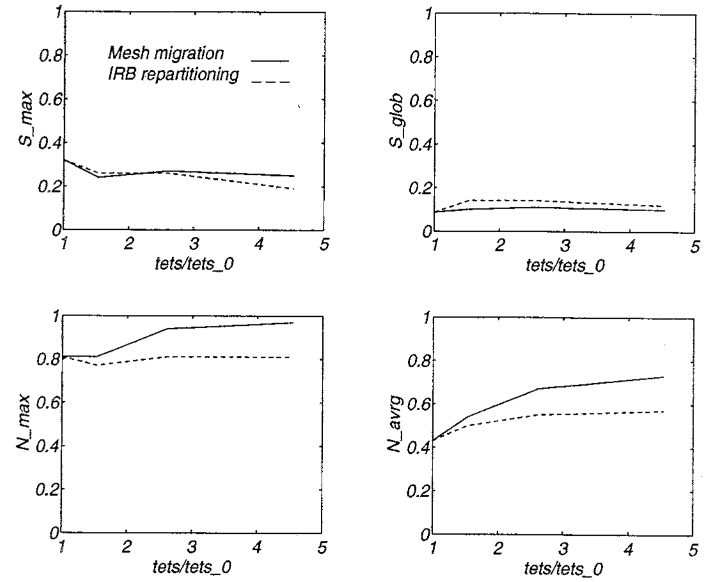

Figure 8: Boundary faces and neighbor statistics for the parallel–adaptive analysis of the ONERA M6 wing in transonic flight using the mesh migration and IRB rebalancing schemes.

have an impact on the performance of the flow solver. For example, the ratio of the wall clock timings for the flow solutions performed on the final adapted mesh was found to be 0.83, in favor of the repartitioning algorithm. It should be pointed out that this is not an objective measure of efficiency of the rebalancing strategy, in the sense that it depends on the algorithm used for the flow solution. On the contrary, $S_{\max}$, S_{glob}, $N_{\max}$ and N_{avrg} are objective measures.

The two approaches were also compared in terms of relative wall clock timing cost. The repartitioning algorithm outperformed the migration scheme at each adaptive step. The ratio of the iterative migration to the rebalancing wall clock timings was found to be 4.07 at the first level (131,000 tetrahedron mesh), 4.41 at the second (223,499 tetrahedron mesh) and 2.21 at the third (388,837 tetrahedron mesh).

These preliminary test results seem to indicate that the iterative load migration scheme tends to be more computationally expensive than the parallel IRB algorithm, and at the same time does not yield the same quality of the partitions, at least with the currently implemented heuristics. However, it must not be forgotten that these tests are certainly not as exhaustive as one might desire for ruling in favor of one approach over the other. Moreover, it is clear that this result is partially due to the low cost of the IRB partitioning, and comparing the migration scheme with other more expensive partitioning algorithms might lead to opposite conclusions. For example, if an algorithm with better control over the number of neighbors could be devised, then the migration scheme used in conjunction with a high quality initial partition (such as the one provided by a spectral partitioning) could yield an overall better performance than a repartitioning scheme. A more complete analysis of the relative merits of the two approaches will be the subject of future work.

Conlusions

The major motivation for the development of automated adaptive techniques is their ability to effectively and accurately resolve intricate features of the solution, such as those that characterize the flow around rotors in hover and forward flight. Although the idea of using such techniques for improving the simulation capabilities of rotary wing codes is certainly not new, our contribution is original in its effort to bridge adaptivity with parallelism on MIMD machines.

We have developed a methodology that, due to its generality, is not restricted to the problems or the examples discussed in this work. We have shown that not only can we accurately determine the airloads of hovering rotors in a variety of situations, but also that we can do it with efficient scalable parallel algorithms. The final aim of our efforts is the analysis of more complex rotary wing problems, such as forward flight with strong blade–vortex interactions and aeroelastic coupled systems. We are confident that the adaptive capabilities of the code will provide a viable tool for the accurate numerical simulation of those effects, while the parallel algorithms that support the analysis in all its phases will make this computations feasible with the resources offered by the current generation of parallel computers.

Acknowledgments

The authors would like to acknowledge the contribution of Zvi Rusak for the many helpful discussions, Can Özturan and Joe E. Flaherty in the development of the data structures, Hugues L. De Cougny for the adaptive routines, Ramamoorthy Ravichandran for the mesh matching procedures.

The authors also gratefully acknowledge the Army Research Office for funding this research through the ARO Rotorcraft Technology Center at Rensselaer Polytechnic Institute (DAAH04–93–G–0003, G. Anderson project monitor).

References

[1] BEALL, M.W. and SHEPHARD, M.S., Mesh Data Structures for Advanced Finite Element Applications, Scientific Computation Research Center, RPI, Troy, NY, in preparation for submission.

[2] BOTTASSO, C.L., DE COUGNY, H.L., FLAHERTY, J.E., ÖZTURAN, C., RUSAK, Z., SHEPHARD, M.S., Compressible Aerodynamics Using a Parallel Adaptive Time–Discontinuous Galerkin Least–Squares Finite Element Method, 12th AIAA Applied Aerodynamics Conference, Colorado Springs, CO, June 20–22, 1994.

[3] CARADONNA, F.X. and TUNG, C., Experimental and Analytical Studies of a Model Helicopter Rotor in Hover, USAAVRADCOM TR–81–A–23, 1981.

[4] DE COUGNY, H.L., DEVINE, K.D., FLAHERTY, J.E., LOY, R.M., ÖZTURAN, C. and SHEPHARD, M.S., Load balancing for the Parallel Solution of Partial Differential Equations, *Appl. Num. Math.*, 16:157–182, 1994.

[5] DE COUGNY, H.L. and SHEPHARD, M.S., Refinement, Derefinement and Local Retriangulations in Three-Dimensions, Scientific Computation Research Center, RPI, Troy, NY, in preparation for submission.

[6] JOHAN, Z., Data Parallel Finite Element Techniques for Large–Scale Computational Fluid Dynamics, Ph.D. Thesis, Stanford University, July 1992.

[7] JOHAN, Z., HUGHES, T.J.R., MATHUR, K.K. and JOHNSSON, S.L., A Data Parallel Finite Element Method for Computational Fluid Dynamics on the Connection Machine System, *CMAME*, 99:113-134, 1992.

[8] KENNEDY, J.G., BEHR, M., KALRO, V. and TEZDUYAR, T.E., Implementation of Implicit Finite Element Methods for Incompressible Flows on the CM–5, Army High–Performance Computing Research Center, University of Minnesota, Rep. 94–017, April 1994.

[9] LEISS, E. and REDDY, H., Distributed Load Balancing: Design and Performance Analysis, Tech. Report vol. 5, W. M. Keck Research Computation Lab, 1989.

[10] LOEHNER, R., An Adaptive Finite Element Scheme for Transient Problems in CFD, *CMAME*, 61:323–338, 1987.

[11] SCHMITT, V. and CHARPIN, F., Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers, AGARD R-702, pag. B1-1, B1-44, 1982.

[12] SHAKIB, F., Finite Element Analysis of the Compressible Euler and Navier–Stokes Equations, Ph.D. Thesis, Stanford University, CA, 1988.

[13] SHAKIB, F., HUGHES, T.J.R., and JOHAN, Z., A New Finite Element Formulation for Computational Fluid Dynamics: X. The Compressible Euler and Navier Stokes Equations, *CMAME*, 89:141-219, 1991.

[14] SHEPHARD, M.S., Linear Multipoint Constraint Applied Via Transformation as Part of a Direct Assembly Process, *Int. J. Num. Meth. Eng.*, 20:2107–2112, 1984.

[15] SHEPHARD, M.S., FLAHERTY, J.E., DE COUGNY, H.L., ÖZTURAN, C., BOTTASSO, C.L. and BEALL, M.W., Parallel Automated Adaptive Procedures for Unstructured Meshes, *Parallel Computing in CFD*, AGARD, 7 Rue Ancelle, 92200 Neuilly-Sur-Seine, France, 1995.

[16] SRINIVASAN, G.R., RAGHAVAN, V., and DUQUE, E.P.N., Flowfield Analysis of Modern Helicopter Rotors in Hover by Navier–Stokes Method, presented at the International Technical Specialist Meeting on Rotorcraft Acoustics and Rotor Fluid Dynamics, Philadelphia, PA, Oct. 15–17, 1991.

[17] TEZDUYAR, T.E., BEHR, M., MITTAL, S. and LIOU J., A New Strategy for Finite Element Computations Involving Moving Boundaries and Interfaces – The Deforming–Spatial–Domain/Space–Time Procedure: I. The Concept and the Preliminary Tests, *CMAME*, 94:353–371, 1992.

[18] WARMING, R.F., BEAM, R.M., HYETT, B.J., Diagonalization and Simultaneous Symmetrization of the Gas–Dynamic Matrices, *Math. of Comp.*, 29:1037–1045, 1975.