# PARALLEL ADAPTIVE FINITE ELEMENT SOLUTION OF HELICOPTER ROTORS IN HOVER

Carlo L. BOTTASSO, Mark S. SHEPHARD

Dipartimento di Ingegneria Aerospaziale
Politecnico di Milano
Via C.Golgi 40, 20133, Milano, Italy
and
Scientific Computation Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

## Abstract

This paper summarizes the technical developments in the area of adaptive CFD for rotary wing aerodynamics that the authors have realized during the past years. We discuss the implementation of parallel adaptive stabilized finite element procedures on distributed memory computers, and present a collection of results related to helicopter rotors in hover.

## Introduction

This paper reports the status of our research efforts in the area of CFD analysis of rotorcraft systems. The emphasis of our work has been devoted to unstructured adaptive parallel procedures. Adaptivity performed on unstructured discretizations allows the accurate capture of different features of the solution, even in complex geometries, while parallel computing offers the potential for satisfying the demand of high performance as well as providing large memories.

A successful general purpose code for rotary wing aerodynamics must be equipped with (i) modeling flexibility, (ii) numerical robustness, (iii) accuracy and (iv) high performance. The procedures outlined herein represent our approach at addressing these issues.

Modeling flexibility is required for handling complex geometries, general boundary conditions, etc. Modeling flexibility is crucial for being able to move from academic test cases to the solution of problems of industrial relevance. The problem of selecting appropriate data structures and procedures for supporting the analysis on arbitrary domains is a non–trivial task, especially in three dimensions. We address the modeling issue by means of geometry based procedures that interact with a CAD representation of the computational domain. This allows us to perform mesh generation and adaption in arbitrarily complex domains. Moreover, the physical attribute information required to support the analysis is tied to the geometric model definition, rather than to the discrete model. This offers distinct advantages in an automated analysis environment like the one used here.

To address the issue of robustness of the numerical scheme, a Time–Discontinuous Galerkin Least Squares (TDG/LS) finite element method is used [8]. This scheme is particularly well suited for the incorporation in an automated adaptive environment. In fact the method has a firm mathematical foundation, and its stability and accuracy properties have been rigorously established.

A general automated procedure should deliver a solution to a prescribed level of accuracy. This result can be achieved through the interaction of two tools: error estimates that determine which regions of the computational domain are over–refined or are not providing sufficient accuracy, and a mesh adaption procedure that optimizes the discretization based on the information provided by the error indicator. We have developed general adaptive strategies that can be used for locally adapting an existing discretization in complex geometries. At present, these mesh adaption routines are driven by simple error indicators. Although error indicators do not necessarily measure the real discretization errors, they do correctly indicate, when carefully chosen, the local regions of the computational domain associated to high errors.

An important property of the present adaptive code is that all the phases of the analysis are performed in parallel, thus avoiding potential bottlenecks. This includes the steps of element formation, equation solution, error

estimation, mesh refinement and load balacing for the next parallel solution [10, 3].

The paper is organized as follows: first, the main features of the procedures are described, sketching the implemented finite element formulation, the data structures used for supporting the various phases of the analysis, special additions to the code needed for efficiently dealing with rotary wing problems, and the mesh adaption scheme. Then we present a collection of test problems to determine blade pressure load measurements, and wake and acoustic wave tracking, and include comparisons with relevant experimental data.

# Parallel Adaptive Finite Element Technology

## Finite Element Formulation

The three–dimensional Euler code has been described in detail in References [3, 4]. The scheme adopts the TDG/LS finite element method [8]. The TDG/LS is developed starting from the symmetric form of the Euler equations expressed in terms of the entropy variables 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 code implements 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. We have developed a new formulation of the deforming element methodology, that incorporates an integration by parts that ensures flux consistency [2].

The code can perform analyses both in a fixed and in a rotating Cartesian frame. This latter option allows to treat a hovering rotor as a steady problem, assuming that the unsteadiness in the wake can be neglected. This implies that the less computationally expensive constant–in–time formulation can be used, together with a special local time stepping strategy that ensures fast convergence to the solution. To this aim, we have extended the stabilized Galerkin Least–Squares formulation to the Euler and Navier–Stokes equations written in a rotating frame [3]. The resulting formulation inherits all the features and properties of the original inertial form.

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 [3, 10]. 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. The code performs an automatic switching to the matrix-free algorithm when the size of the problem prevents the storage of the tangent matrix in core memory. Preconditioning is achieved by means of a nodal block–diagonal scaling transformation.

## Geometry–Based Data Structures

An important highlight is that the code is built on top of a geometry–based database [1]. The database stores complete knowledge of the relations (usually termed "classification") of each mesh entity with the underlying geometric model. The procedures are directly interfaced with several general purpose geometric modelers that provide access to the geometric and topological description of the domain. The understanding of these relationships allows to guarantee the validity of the generated discretizations during any mesh modification operation, to determine when local modifications of the mesh would create invalid elements and to guarantee that refinement improves the geometric approximation that a mesh gives of the true geometry through the positioning of newly generated vertices on the model boundaries.

A wire–frame sketch of a typical solid model for a two–bladed hovering rotor is given in Figure (1). Given the symmetry of the geometry and of the flowfield, the computational domain is represented by a half cylinder that encloses one of the two rotor blades. The diametral face of the cylinder has been split in two faces, that are used for applying the symmetric flowfield conditions as explained below. The bottom face has been inscribed with an edge that is used for defining an outflow to the computational domain.

The specification of the physical attributes of the anal-

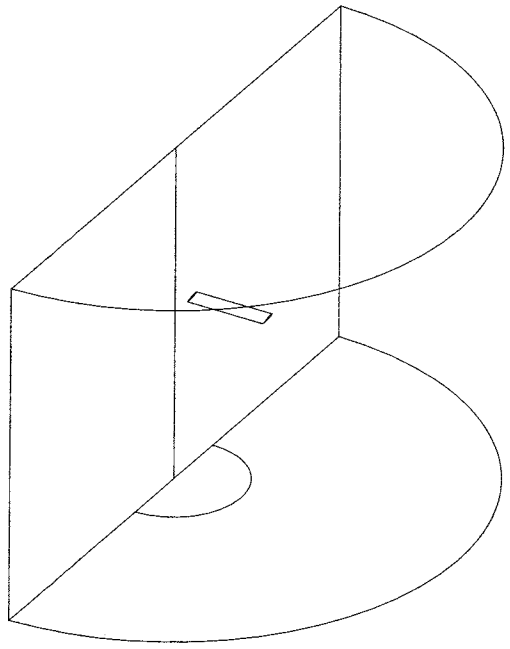

Figure 1: Wire–frame sketch of the solid model for a two–bladed hovering rotor.

ysis (boundary conditions, initial values, etc.) is handled by means of a general model–based procedure. The user associates the attributes to the appropriate entities of the geometric model, using a limited set of macro operators. The code retrieves this information when necessary and associates it with the finite element entities. This ensures completely automatic data management in arbitrarily complex computational domains with no user intervention, even when mesh adaption is used. For additional generality and flexibility of the code, non–uniform attribute distributions are supported.

## Special FEM Techniques for Rotary Wing Aerodynamics

We have incorporated in our code a set of procedures to allow flexibility and efficiency in the analysis of 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. The inflow/outflow far–field conditions are based on the methodology suggested by Srinivasan et al. [11], 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. Figure (2), gives a graphical representation of the resulting far–field inflow and outflow velocities.

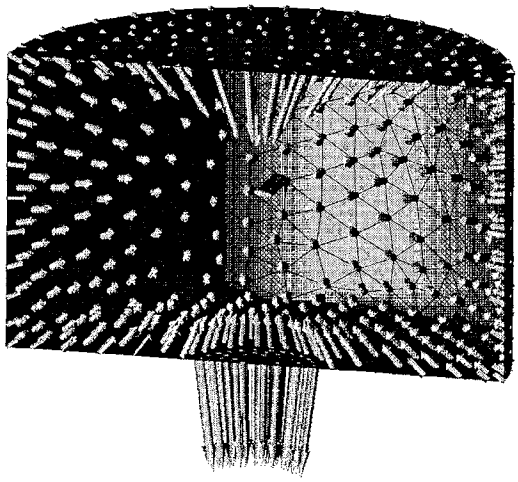

Figure 2: Far–field boundary conditions for a hovering rotor (Note: inflow/outflow velocities not to scale.)

Another important condition that must be considered for the efficient simulation of hovering rotors is the periodicity of the flowfield. 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 [9]. 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.

Figure (3) illustrates the mesh matching process for a two–bladed rotor. 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 about the axis of rotation through the symmetry angle 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.

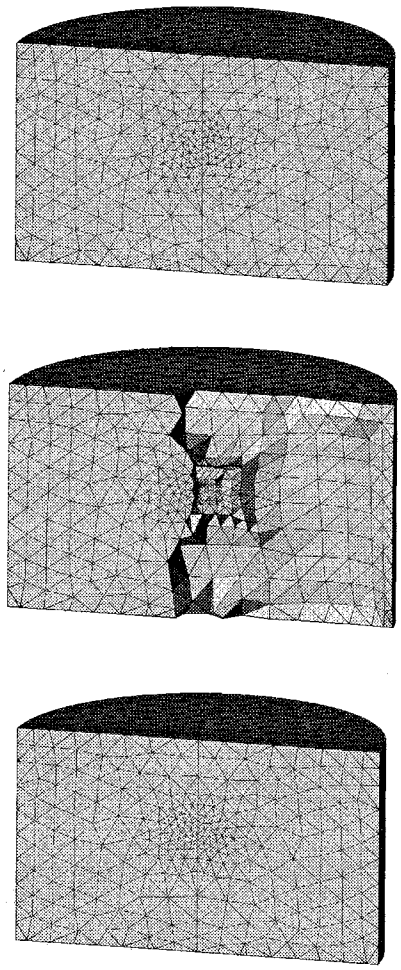

Figure 3: Mesh matching of the periodic face discretizations. Top: initial non–matching mesh. Center: elements connected to one of the periodic faces are deleted. Bottom: matched mesh.

## Mesh Adaption Scheme

The parallel mesh data structures for supporting the solution of PDE's, the partitioning of the discretized computational domain, and the parallel adaptation of it, are discussed in References [10, 6].

The code implements a topological entity hierarchy data structure, 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 can also be loaded into the memories of the individual processors and repartitioned using a distributed IRB routine.

The mesh is then adapted based on the information provided by the error indication performed on the converged finite element solution. Two types of error indicators are currently implemented in the code. Each error indicator can be simultaneously applied to one or more physical variables (density, pressure, temperature, entropy and Mach number). The first indicator is the magnitude of the gradient of the key variable. This indicator is not based on solid mathematical ground, but it is useful in practice when relatively small local regions of large gradients of the solution variables are present. This is typically the case of many rotary wing aerodynamic problems. The second is based on an estimate of the curvature of the solution for the selected key variables [3]. The nodal values of the error indicators are mapped to the finite element edges, and used for driving the edge based mesh adaption process. The evident drawback of the use of such indicators, is that the user must specify high and low thresholds for the error indicators. Each edge in the mesh is checked for its associated error indicators, and refined if these are found to be higher than the high threshold, or collapsed if they are lower then the low threshold. Although nice results can be obtained with this technique as shown in the numerical results section, it is our experience that this poses a heavy work load on the user that has to find through a trial and error process, values of the thresholds that lead to suitable meshes. True error estimators are required to eliminate this trial and error process.

The mesh adaptive algorithm combines derefinement, refinement and triangulation optimization using local retriangulations. 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 eliminate over-refinement. As all the other building blocks of the code here discussed, the mesh adaptation algorithm has been completely parallelized [10, 6].

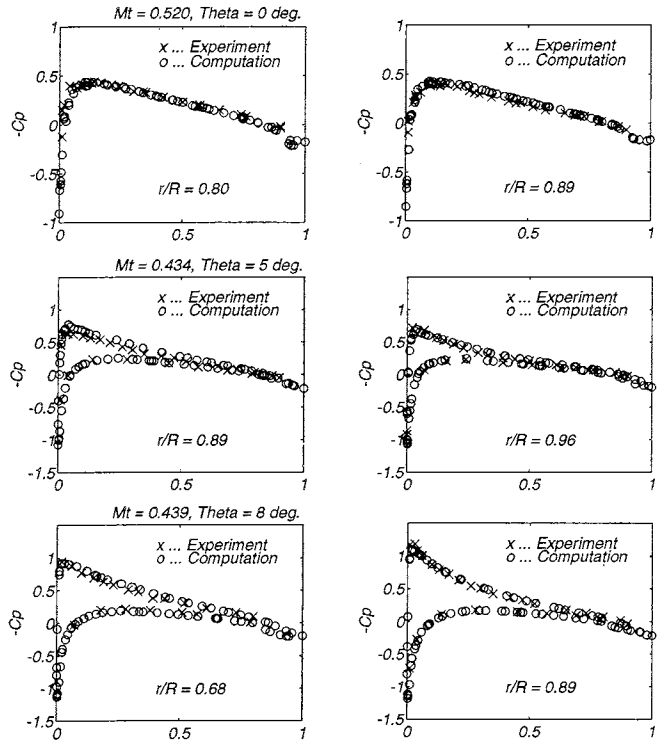


Figure 4: Computed and experimental pressure coefficients on the blade at different span locations, for the three subsonic cases $\theta_c = 0^o$, $M_t = 0.520$; $\theta_c = 5^o$, $M_t = 0.434$; $\theta_c = 8^o$, $M_t = 0.439$.

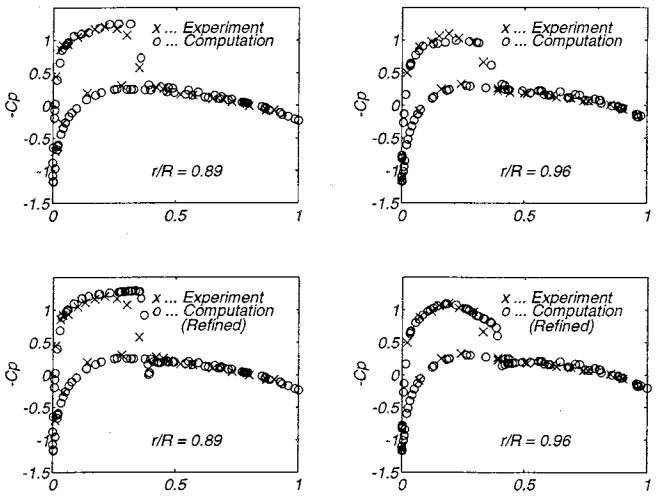


Figure 5: Computed and experimental pressure coefficients on the blade, at two different span locations close to the tip, $\theta_c = 8^o$, $M_t = 0.877$. Top two plots: initial coarse 142,193 tetrahedron grid. Bottom two plots: adapted (three levels) final 262,556 tetrahedron grid.

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. The second is a load balancing scheme that iteratively migrates elements from heavily loaded to less loaded processors.

## Numerical Experiments

In this section we present results gathered during a number of numerical experiments related to subsonic and transonic hovering rotors.

We first address the prediction of blade pressure distribution, with the help of some numerical tests discussed also in Reference [3]. Caradonna and Tung [5] have experimentally investigated a model helicopter rotor in several subsonic and transonic hovering conditions. 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 (4) 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, including the section close to the tip. 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.

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 per time step. 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

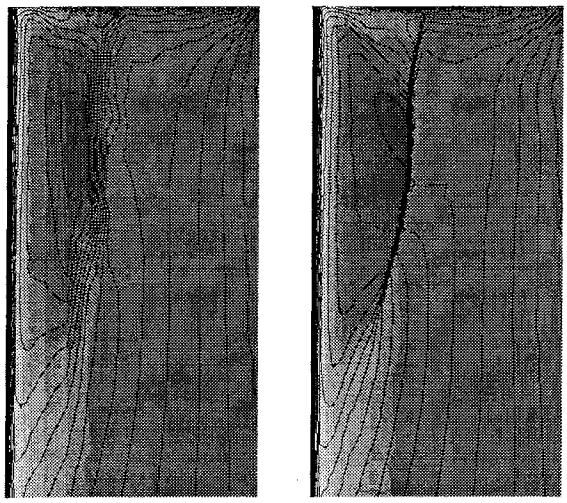

Figure 6: Density isocontour plots on the upper surface of the blade tip, $\theta_c = 8^o$, $M_t = 0.877$. At left: initial coarse grid. At right: final adapted grid.

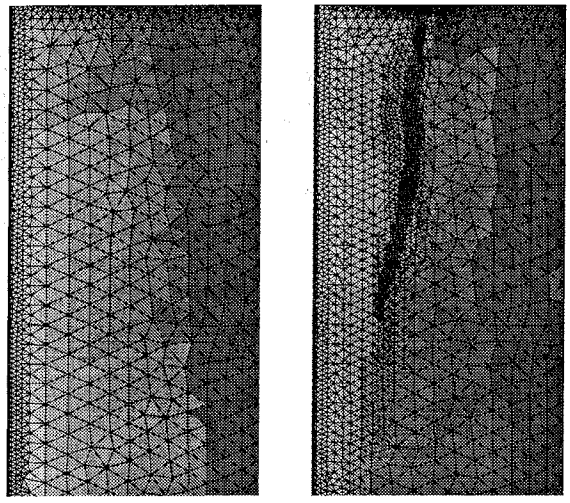

Figure 7: Meshes with partitions on the upper surface of the blade tip, $\theta_c = 8^o$, $M_t = 0.877$. At left: initial coarse grid with IRB partitions. At right: final adapted grid with partitions obtained by migration.

norm of the residual of almost four orders of magnitude in 50 to 60 time steps.

Figure (5) 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 (5) 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. Consistent 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.

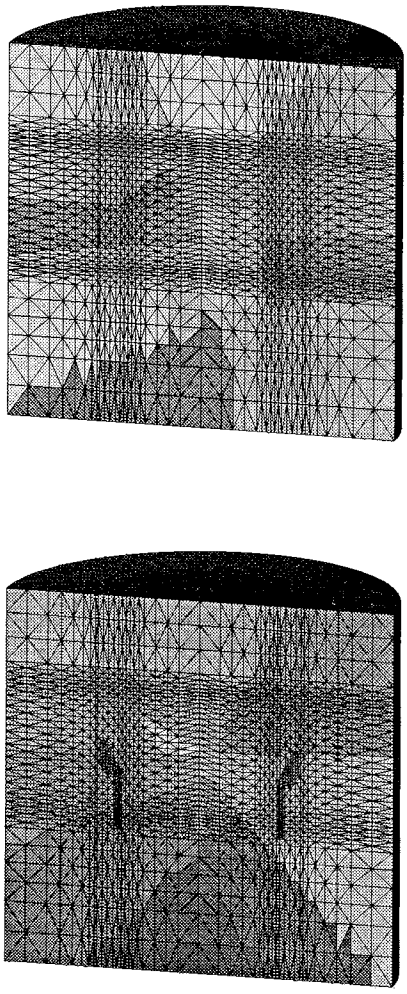

Figure 8: Meshes with partitions for the wake confinement problem, $\theta_c = 8^o$, $M_t = 0.439$. Top: initial coarse grid with IRB partitions. Bottom: adapted grid (1 level) with IRB partitions.

The effect of the adaptation of the mesh on the resolution of the shock is clearly demonstrated in Figure (6), where the density isocontour plots at the upper tip surface are presented for the initial and adapted meshes.

Figure (7) 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.

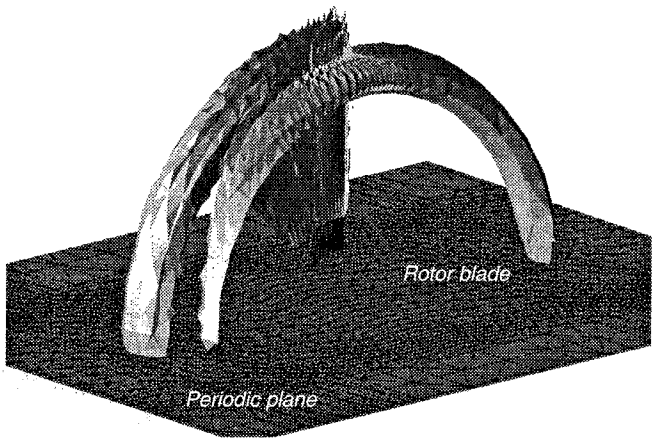


Figure 9: Error indicator for the wake confinement problem, $\theta_c = 8^o$, $M_t = 0.439$.

The test case denoted by $\theta_c = 8^o$, $M_t = 0.439$ and $C_t = 0.00459$ was then selected for testing the ability of the code in performing wake modeling by h–adaptivity. The goal of the exercise is that of capturing the vortical structures shed by the blades and their mutual interactions, without any ad–hoc wake confinement formulation but simply by h-adapting the mesh.

For this purpose, an initial mesh denoted by 284,342 tetrahedra was partitioned in 32 subdomains using the IRB algorithm (the initial mesh is courtesy of Roger Strawn, NASA Ames Research Center). The parallel implicit analysis was performed in the usual way, obtaining a four–order of magnitude reduction in the energy norm of the residual in 50 time steps. The solution was then used to compute an error indicator based on vorticity, with limiters on the minimum edge length in order to control the size of the mesh. Figure (8) shows the far field and periodic boundaries of the initial and refined meshes. In order to maintain the vertex matching at the periodic faces, the adaptive code was modified in order to guarantee that the refined face discretizations satisfy the matching requirement.

Figure (9) shows an isosurface representation of the error indicator used for targeting the elements in the wake region for refinement. The view is taken from below the plane of the rotor.

Another two levels of refinement were applied to the mesh, each level followed by an implicit solution to convergence. The final mesh is denoted by 1,074,112 tetrahedra, mainly clustered at the tip of the blade and in the wake region. For the final mesh, about 720 implicit iterations at a CFL of 20 were necessary for reducing the energy norm of the residual of 2.5 orders of magnitude. We remark that the convergence on the refined mesh is somewhat problematic: not only a large number of implicit time steps are necessary, but we also observed that the wake undergoes an undamped oscillatory motion of small amplitude about its reference position.

The effect of the mesh adaption is clearly shown in Figure (10). The left part of the picture shows a three–dimensional vorticity isosurface plot, obtained on the initial coarse mesh. The wake appears to be tracked for almost $270^o$, it interacts with the blade and it is deviated downwards, dissipating shortly after. The right part of the same picture shows the same vorticity isosurface plot obtained with the final refined mesh. Note that now the wake is tracked for almost $360^o$. The diameter of the isosurface at the location where the wake interacts with the blade is also significantly increased.

Figure (11) shows a comparison of the wake geometry with the experimental data of Caradonna and Tung. We report the vertical position of the wake core, as well as the radial position of the core versus the azimuth angle. Both quantities show excellent correlation with the experimental data. The position of the wake core was determined by computing the centroid of the isosurface of Figure (10) at intervals of $25^o$ of the azimuth angle.

The code was also tested in its ability to perform adaptive High Speed Impulsive (HSI) noise computations, in order to evaluate the feasibility of interfacing it with a Kirchhoff integral solver. Such a coupled CFD–Kirchhoff procedure would allow a solution adaptive unstructured CFD simulation of the acoustic field close to the rotor, while the Kirchhoff formulation would be responsible for propagating the acoustic signal to the far field.

For this purpose, a test case experimentally investigated by Purcell [7] was selected. The same problem has been numerically simulated by a number of researchers, including Strawn et al. [12]. The problem is that of a rectangular blade hovering rotor with NACA0012 airfoil sections and aspect ratio of 13.71, denoted by a tip Mach number $M_t = 0.95$. The test case is characterized

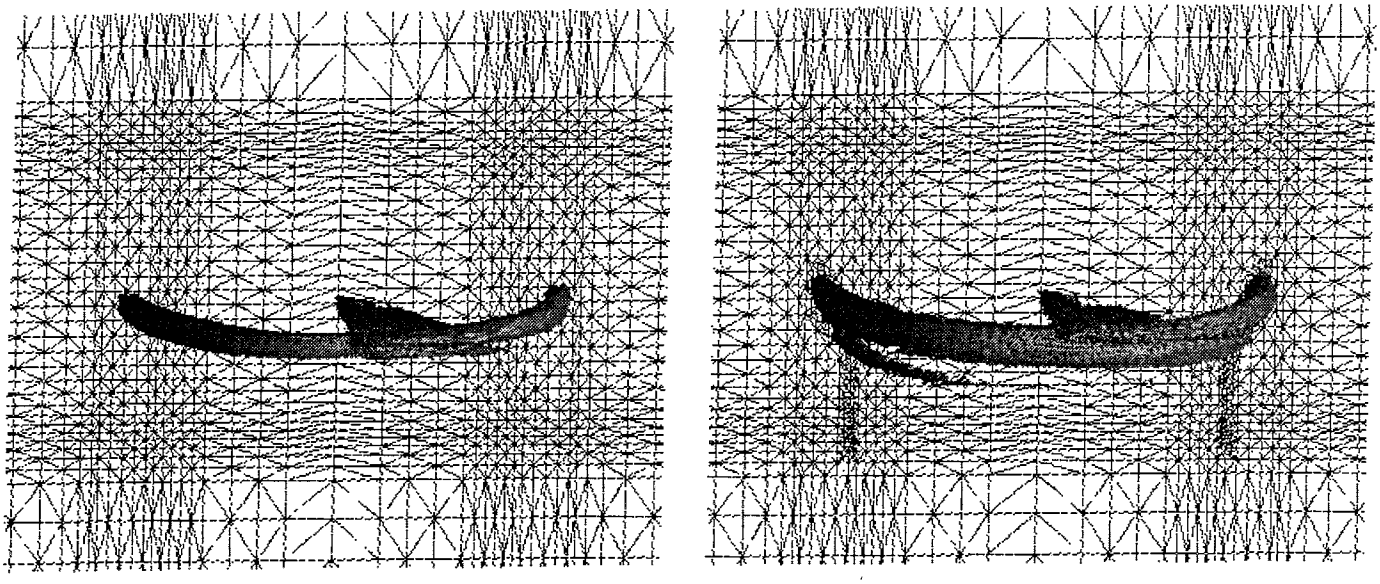

Figure 10: Vorticity isosurface plots for the wake confinement problem, $\theta_c = 8^o$, $M_t = 0.439$. At left: vorticity isosurface before refinement (284,342 tetrahedra). At right: vorticity isosurface after refinement (3 levels, 1,074,112 tetrahedra).

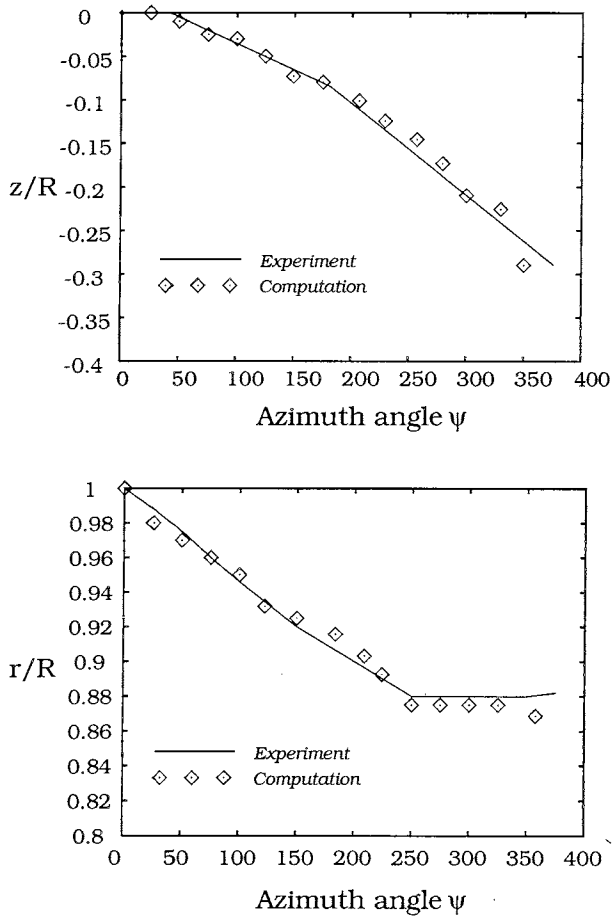


Figure 11: Wake geometry for the wake confinement problem, $\theta_c = 8^o$, $M_t = 0.439$.

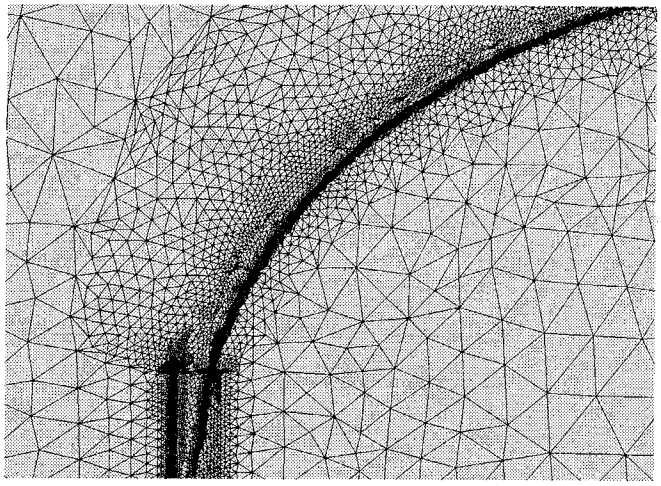

Figure 12: Final refined mesh (4 levels) for the acoustic wave problem, $\theta_c = 0^o$, $M_t = 0.95$.

by a marked tip delocalization. The simulation is conducted for the non–lifting case, therefore the problem is symmetric about the plane of the rotor and the computational domain extends only on one side of the plane itself. The far field boundaries are located at 1.5 radii above the plane and at 3 radii from the hub. Periodic boundary conditions are applied at the symmetric faces, while slip conditions are applied at the rotor disk plane to account for symmetry.

The parallel adaptive analysis was conducted with four refinement levels, each followed by subsequent anal-

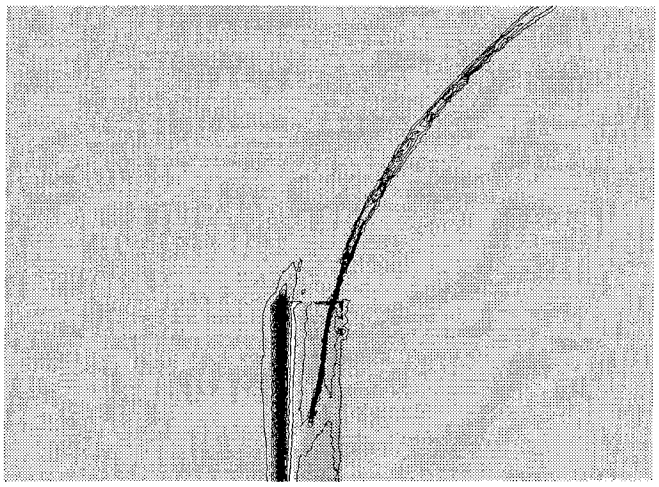

Figure 13: Isosurface plot of the error indicator for the acoustic wave problem, $\theta_c = 0^o$, $M_t = 0.95$.

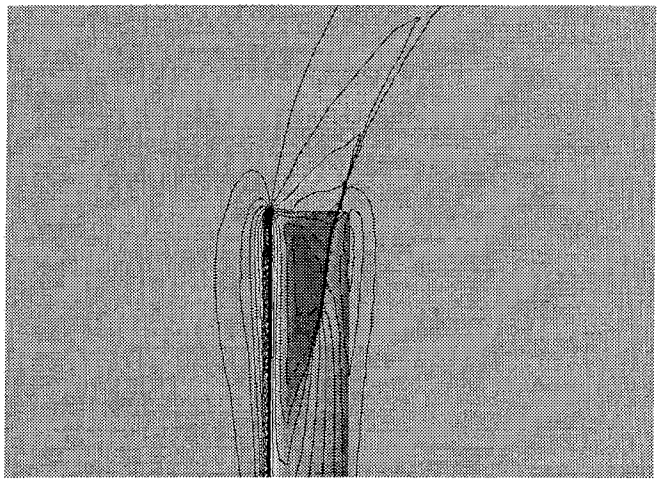

Figure 14: Pressure distribution on the blade and on the plane of the rotor for the acoustic wave problem, $\theta_c = 0^o$, $M_t = 0.95$.

ysis to convergence. Figure (12) shows the final mesh, characterized by 575,026 tetrahedra. Note the refinement along the tip blade shock and along the acoustic wave.

The error indicator selected in this case is based on the norm of the gradient of pressure, with limiters on the minimum allowable edge lengths. The principal scope of the limiters is to prevent the excessive refinement of the leading edge of the blade tip, that otherwise would lead to excessively large meshes. Clearly, the use of a true error estimator —as opposed to the the simple error indicator adopted here— that vanishes when the local mesh size reaches the appropriate value, would eliminate the need of such artificial devices. Figure (13) shows an isocontour plot of the error indicator.

The pressure distribution on the blade surface and on the plane of the rotor is presented in figure Figure (14). Note that the shock wave is very nicely captured, resulting in a very sharp jump.

The encouraging results obtained in this preliminary simulation in targeting for refinement different features of the flow field, such as the strong blade shock and the acoustic signal, seem to indicate that a combined CFD–Kirchhoff procedure can achieve a high level of reliability and efficiency. The development of such a software tool is currently underway.

## Conlusions

We have reported some recent advances on parallel adaptive finite element procedures for the solution of rotary wing compressible flow problems. The code has been successfully validated against a number of interesting problems in rotor CFD, from subsonic and transonic blade pressure predictions, to wake confinement through h–adaption and acoustic computations.

These results indicate that unstructured adaptive techniques are indeed able to accurately capture the flow features of the solution in a wide range of problems of helicopter interest. Clearly, the choice of appropriate error measures is crucial to the success of any automated adaptive scheme. Although the simple error indicators used in this work have been successful in driving the adaption towards efficient meshes for the problems considered, more robust and reliable error estimators need to be incorporated in the code.

This work shows that the bridging of adaptive techniques, stabilized finite element formulations, advanced data structures and parallelism, constitutes a viable modern approach to the solution of this class of challenging problems.

## Acknowledgments

The authors 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, submitted to *Int. J. Num. Meth. Eng.*

[2] BOTTASSO, C.L., On the Computation of the Boundary Integral of Space–Time Deforming Finite Elements, *Comm. Num. Meth. Eng.*, to appear.

[3] BOTTASSO, C.L. and SHEPHARD, M.S., A Parallel Adaptive Finite Element Euler Flow Solver for Rotary Wing Aerodynamics, *AIAA J.*, under review. Presented: 12th AIAA Computational Fluid Dynamics Conference, San Diego, CA, USA, June 19–22, 1995.

[4] 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.

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

[6] DE COUGNY, H.L. and SHEPHARD, M.S., Parallel Mesh Adaptation by Local Mesh Modification, Scientific Computation Research Center, RPI, Troy, NY, in preparation for submission.

[7] PURCELL, T.W., CFD and Transonic Helicopter Sound, Paper No. 2, 14th European Helicopter Forum, Milan, Italy, September 20–23, 1988.

[8] 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.

[9] 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.

[10] 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, Vol. R–807, 6.1–6.49, 1995.

[11] 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.

[12] STRAWN, R., GARCEAU, M., and BISWAS, R., Unstructured Adaptive Mesh Computations of Rotorcraft High–Speed Impulsive Noise, 15th AIAA Aeroacustics Conference, Long Beach, CA, USA, October 25–27, 1993.