PARALLEL CHIMERA COMPUTATIONS OF HELICOPTER FLOWS

G. Jeanfaivre, X. Juvigny, C. Benoit
ONERA, DSNA

Keywords: *Parallel, Chimera, Helicopter*

Abstract. *In the CFD field, chimera method is a meshing technique that enables to deal with complex geometries using overset structured grids. In the context of parallel scalar computers, its efficiency can be risen by balancing the overset grid blocks over the machine processors. In this paper, the chimera method has been parallelized in Onera's* **elsA** *software leading to efficient computations of steady and unsteady (with moving bodies) applications. Particular focus is put on helicopter applications.*

Introduction. *The numerical simulation of flows around complex bodies, and in particular those related to helicopters, requires a computing power and a memory size increasingly larger. For this reason, the use of parallel machines becomes more and more attended, even impossible to circumvent. A parallel implementation, by a better exploitation of the machines, makes possible the study of more complex geometries and the mesh resolution can be improved.*

In addition, the chimera method, frequently used for the numerical simulation of flows, in particular for applications to the helicopters, allows the calculation of multiblock meshes composed of overset structured grids. Thus, from the reduction in the constraints on the grid blocks, the generation of the mesh is simplified and, in the case of moving bodies, the mesh does not have to be regenerated at each time step.

This paper presents recent progress concerning the implementation of the chimera method in Onera's CFD grid structured software, called elsA, previously presented in [1]. This method has been parallelized for both steady and unsteady applications and applied to rotorcraft simulations. In the first subsection, the flow equations and the numerical scheme are recalled. Then, the chimera method is briefly described for a better understanding of the parallel algorithm, topic of the following paragraph. Finally, application to an isolated rotor in hover is presented, followed by applications to rotor in forward flight and rotor-fuselage interaction.

Equations and Numerical scheme. *In this paper, the 3D compressible Euler equations are solved by using a finite-volume discretization scheme on each structured grids. For rotors, the Euler equations are written in a blade attached rotating reference frame. In this frame, the system of equations is formulated in terms of absolute velocities, which is a necessary condition for an accurate treatment of the numerical fluxes and of the farfield boundary conditions.*

A Jameson like centred scheme of second order accuracy with artificial viscosity (Jameson et al., 1981) is used for discretizing the equations. They are advanced in time using a four-stage Runge-Kutta explicit scheme. A simplified implicit stage (Lerat et al., 1982) increases the stability domain.

Chimera method. *The chimera method [7] enables to solve the Euler or Navier-Stokes equations on a set of overset grids.*

It relies on two kind of transfers between overset grids : at the overlapping boundaries of grids and in the vicinity of blanked regions. Blanked regions are points of a grid that lie physically in a body, meshed by another overlapping grid. Those transfers are realized by interpolation of conservative variables. Since Jameson numerical scheme is used in **elsA**, *classically, two layers of interpolated cells are built around blanked points or overlap boundaries to enable the numerical scheme to be applied without any modification on discretized points. Nevertheless, in order to diminish the constraints on the size of overlap, one layer of interpolated cells [4] can be used in* **elsA**. *The numerical scheme is then modified on neighbouring cells.*

Here, the parallelization of chimera concerns one and two layers of interpolated cells.

Parallel algorithm. *When parallelizing the chimera method, one crucial point is minimizing the interpolation transfers between the grids.*

Let F_{int} the value of the flow field to be determinated for an interpolated point. F_{int} is obtained by linear combination of field values taken on edges of the interpolation cell (coefficients x_i are interpolation coefficients) :

$$F_{int} = x_1F_1 + x_2F_2 + x_3F_3 + x_4F_4 + x_5F_5 + x_6F_6 + x_7F_7 + x_8F_8. \quad (1)$$

So, interpolating a field in three dimensions implies its knowledge at the eight edges of the interpolation cell. If the two blocks concerned by a transfer (the interpolated block, B_D, and the interpolation block, B_N) are not managed by the same processor, this procedure imposes many exchanges. In order to minimize the number of communications, the storage of the interpolation coefficients is carried out by the interpolation block that contains the interpolation cell (see Fig. 1, 2).
Moreover, always to obtain optimal communications, the data are separated into two kinds : local data (the two blocks concerned by interpolation are on the same processor), and global data (on two different processors).

Thanks to bufferization, all data are sent to all interpolation or interpolated blocks at one go. The send and the receive are carried out by a key which allows to identify block sending and receiving. On the other hand, as data are often stored in array or lists, the key allows to the receive block to manage array data meant for it. Moreover, the management of communication graphs prevents deadlocks.

Only load balance by distribution (that is without splitting the blocks) is performed and only the number of points is taken as a criterion to find the best configuration. This will be improved in the future, following the work of [8].
Communications between processors are performed with MPI subroutine calls [5].

Implicit interpolations. *Sometimes when the overlap between grids is too short, a valid interpolation cell can not be found. When the interpolation cell is itself interpolated for instance, a special treatment is required. This treatment is called implicit interpolations [3]. For example, if edges 1 and 3 are also to be interpolated, equation (1) becomes :*

$$F_{int} - x_1F_1 - x_3F_3 = x_2F_2 + x_4F_4 + x_5F_5 + x_6F_6 + x_7F_7 + x_8F_8. \quad (2)$$

Points where solution is unknown are grouped together in the left side. Applying this principle to all implicit points leads to solve a linear system :

$$AX = B$$

The matrix A is composed of interpolation coefficients, the vector X represents the unknown interpolated field values on implicit points and B, field values obtained from numerical scheme. Coefficients of this sparse matrix are stored by row [6].
In parallel, new methods are implemented to take into account :
- the transfer of some data, like indexes of global interpolated cells or number of implicit points (to place matrix coefficients), which are on other processors;
- the creation of group containing only processors

which treat implicit points.
The implicit interpolation treatment is carried out in two stages : matrix construction and matrix system resolution. In the first stage, each processor stores one part of the matrix. Matrix part that concerns local interpolation blocks and global interpolation blocks are stored separately.

In the second stage, since the matrix is not symmetrical but the diagonal beeing strictly dominant, the system resolution is carried out by the BiCGStab method. For this method, two operations require communications : scalar product and matrix-vector product. For the first operation, each processor computes one part of the scalar product and then a collective operation combines values from all processes and distributes the result back to all processes. Concerning the matrix-vector product, a collective operation gathers data from all tasks and delivers it to all.

Steady applications. *Through all the paper, flow simulations are achieved solving the Euler equations. Numerical computations have been conducted on a SGI Origin 2000 and on a DEC HPC320.*

In this paper, the speed-up is defined by :

$$\text{Speedup} = \frac{\text{Min}(T_1, T_s)}{\text{Max}(T_n)_{n=1,N}} \qquad (3)$$

with T_s, *sequential cpu time*
T_1, *cpu time for parallel computation on one processor*
N, *processor number*
T_n, *cpu time for parallel computation on the processor* n.

The case of the the 7A fourbladed helicopter rotor in hovering flight ($M_t = 0.662$, $\theta_c = 7.5°$) *is first considered. The chimera grid system is made of a mesh around each blade and a background cartesian grid, locally refined in the region of the blades, and, for a better load balance, split into four blocks. Mesh of blades consists in* $140 \times 24 \times 17$ *points and mesh of cartesian grids is made of* $31 \times 31 \times 60$ *points (Fig.3). Two computations are carried out for this configuration, one with one layer of explicit interpolated points and the other with two layers of explicit and implicit interpolated points [3].*

The relative Mach number is shown in Fig. 5 and in Fig. 7, and is equivalent for the two cases. Results obtained by parallel and sequential algorithms are strictly identical as demonstrated in 4 and in 8. Furthermore, Fig. 7 shows a little difference between one layer and implicit case, probably due to the fact that implicit case exhibits extrapolated points (see table 3).

Speed-up plots are presented in Fig. 6 and Fig. 9 for different numerical experiments. For a best understanding, a table shows the repartition of number of computed points by processor (table 1 and the number of explicitly, implicitly interpolated and extrapolated points (table 2 and 3). It can be concluded that even if the number of computed points is well distributed other the processors, the speed-up remains not optimal. The chimera transfers must be taken into account in the load-balance.

Unsteady applications. *The case of an isolated rotor in forward flight is then considered.*

Blades are animated with an harmonic movement of pitch, flap and lead-lag. The tip Mach number is set to $M_t = 0.646$ *and the forward speed corresponds to an advance ratio of* $\mu = 0.4$.

The mesh topology is the same as for the hover case. The computation is carried out on a DEC computer. A result concerning the relative Mach number is given in Fig. 10 after one revolution. The method capability to predict some of the main characteristics of the rotor flowfield is clearly demonstrated. Especially the transonic pocket on the advancing blade side and the reverse circle on the retreating blade are two typical features of helicopter rotor flowfield.

As for the rotor in hover case, results obtained by parallel and sequential algorithms are strictly identical.

In this case, the interpolated blocks and interpolation blocks are more often on different processors than for the hover case resulting in a lower speed-up (see Fig. 11).

The case of the rotor-fuselage interaction has been also computed using the parallel algorithm.

The configuration is shown on Fig. 12. The rotor is the 7A model rotor and the fuselage is the simplified Helishape "Dauphin" of Eurocopter. The length of the blade is $14c$ *and the distance between the rotor rotation center and the top of the fuselage is* $0.66\,c$ *(with* c*, blade profile chord).*

The mesh is made of a curvilinear block around each blade, on intermediary cartesian grid split in four grids, and 14 *blocks for the fuselage for a total of* 1361904 *points. No special care was brought to those grids. The mesh is shown on Fig. 13.*

The rotor movement is the same as for the isolated rotor in forward flight test case.

Five processors on Compacq enables to carry out one revolution in 43 hours. Relative Mach number contours are given in Fig. 14. The result is overall as expected.

In the **elsA** *software, processors are attributed at blocks. Whereas, here, repartition of the number of points by block is completely unequal. So, the load balance is very poor, which explains bad results for the speed-up (see Fig. 15). For more precisions, we present a table (4) with the number of points by processor.*

At last, pressure coefficients on the blade surface for different sections and azimuthal angles ψ *are shown on Fig. 16-19. Results about isolated rotor and interaction rotor-fuselage are compared with experimental results provided by Onera S1 wind tunnel at the Modane center [2]. The difference between the two simulations shows that the fuselage's influence on pressure coefficient is clearly demonstrated, overall at* $\psi = 180^o$ *and* $\psi = 360^o$*, where blades are parallel to the fuselage. As expected, the fuselage influence decreases from the innermost to the outermost spanwise blade sections. As a general rule, the three curves are nearly joined, nevertheless, where differences are noteworthy, experimental results are more close by rotor-fuselage results.*

Conclusion. *The chimera method in* **elsA** *software has been parallelized. The results concerning steady applications are rather good. Nevertheless, results concerning unsteady applications demonstrate that load balance based on the volume of data exchange is required.*

Acknowledgments. *Authors like to thanks the french DGA and DPAC for their support to this study.*

References

[1] *L. Barrera, C. Benoit, M.-C. Le Pape, R. Houdeville, J. Peter, and J.-C. Jouhaud. Advanced Numerical Simulation on Structured Grids for Transport Aircraft Using an Efiicient Object Oriented Solver. In* proceedings of ICAS 2002, *ICAS 2002-2.5.3, 2003.*

[2] *P. Beaumier, M. Costes, and R. Glavériaux. Comparison betweenFP3D Full Potential Calculations and S1 Modane Wind Tunnel Test Results on Advance Fully Instrumented Rotors. In* proceedings of the 19^{th} European Rotorcraft Forum, *Cernobbio, Como (Italy), Septemnber 14-16, 1993.*

[3] *C. Benoit.* Méthode d'adaptation de maillages au moyen d'algorithmes génétiques pour le calcul d'écoulements compressibles. *PhD thesis, ENSAM, 1999.*

[4] *G. Jeanfaivre, C. Benoit, and M. C. Le Pape. Improvement of the Robustness of the Chimera Method. In* proceedings of the 32^{nd} AIAA Fluid Conference and Exhibit, *Saint Louis, Missouri, USA, 2002.*

[5] *MPI. The message passing interface (MPI) standart. In* http://www-unix.mcs.anl.gov/mpi/, *Published by University of Tenessee, Knoxville, 1995,1996,1997.*

[6] *Y. Saad.* Iterative methods for sparse linear systems. *PWS publishing company, 1996.*

[7] *J. Steger, F.C. Dougherty, and J.A. Benek. A Chimera Grid Scheme.* Advances in Grid Generation, K.N. Ghia and U. Ghia, eds., ASME FED-Vol. 5., *June 1983.*

[8] *A.M. Wissink and R.L. Meakin. On parallel implementation of dynamic overset grid methods. In* Technical report of the SC97, San Jose, *Nov 15-21 1997.*

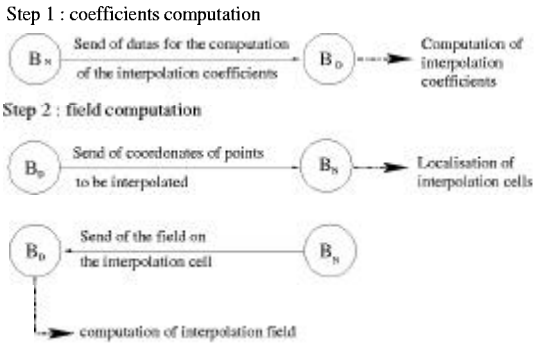


Fig. 1 Communications if interpolated block, B_D, stores interpolation coefficients

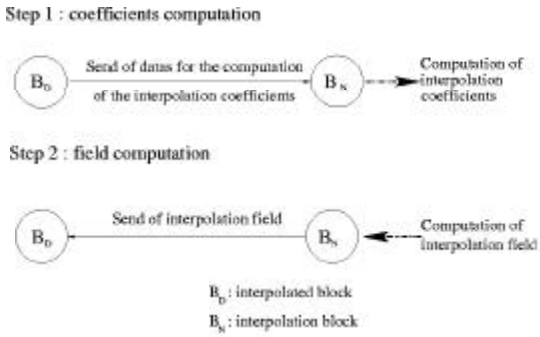


Fig. 2 Communications if interpolation block, B_N, stores interpolation coefficients

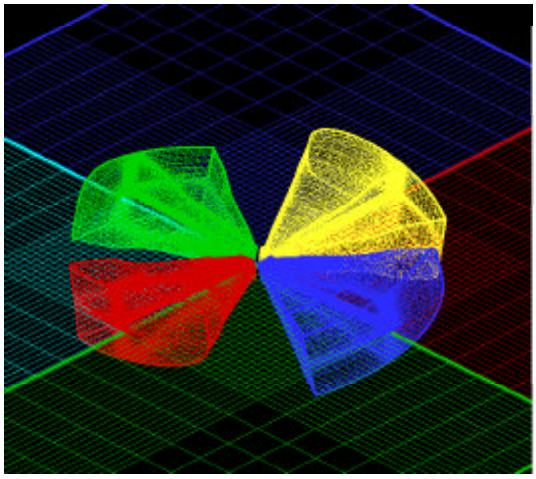


Fig. 3 Mesh for isolated rotor in hover

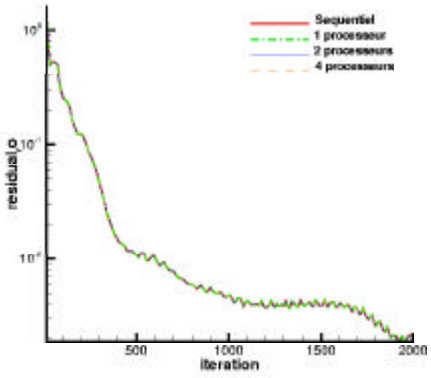


Fig. 4 Residual for isolated rotor in hover, one layer of interpolated points

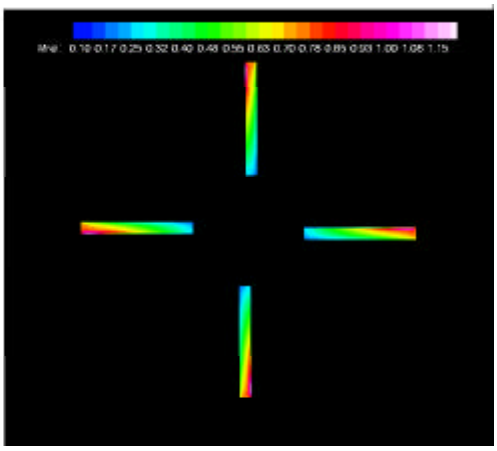


Fig. 5 Relative Mach number for isolated rotor in hover

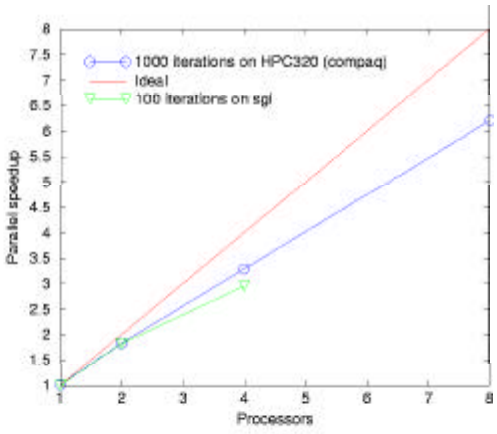


Fig. 6 Speed-up for isolated rotor in hover, one layer of interpolated points

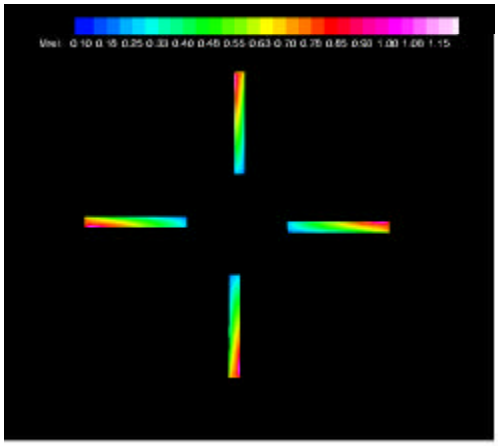


Fig. 7 Relative Mach number for isolated rotor in hover, implicit interpolations

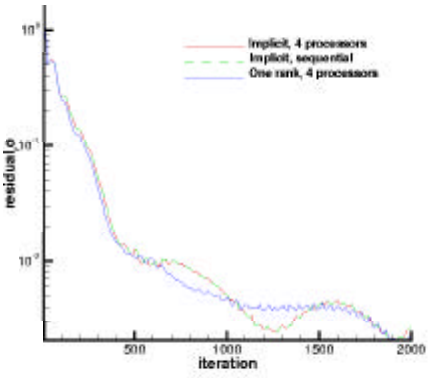


Fig. 8 Residual for isolated rotor in hover, implicit interpolations

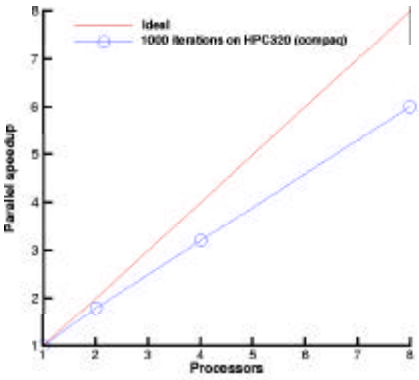


Fig. 9 Speed-up for isolated rotor in hover, one layer of interpolated points, implicit interpolations

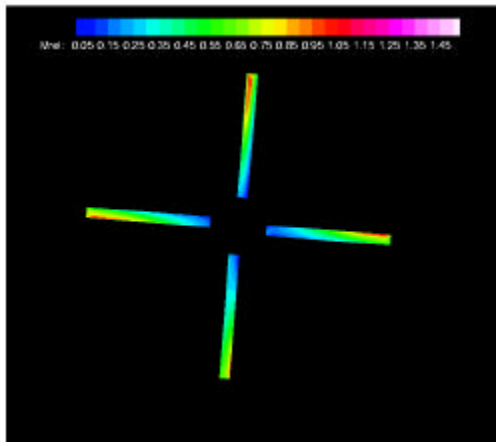


Fig. 10 Relative Mach number for an isolated rotor in forward flight at $\Psi = 90^o$

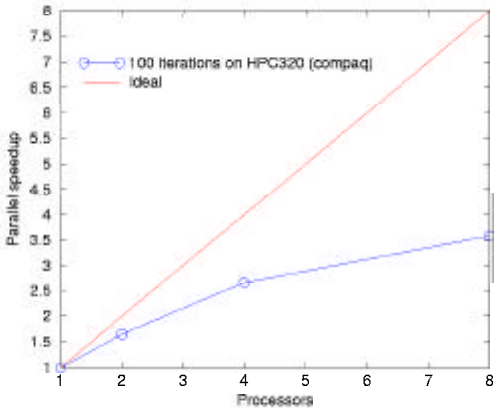


Fig. 11 Speed-up for isolated rotor in forward flight

Fig. 12 Rotor-fuselage consfiguration

1 Proc	862906							
2 Procs	431452	431452						
4 Procs	215726	215726	215726	215726				
6 Procs	137052	137052	147200	147200	147200	147200		
8 Procs	147200	68526	68526	147200	147200	147200	68526	68526

Table 1 Distribution of the number of points for the rotor in hover case

Type of points	Interpolated	Extrapolated
Blade 1	8888	0
Blade 2	8888	0
Blade 3	8888	0
Blade 4	8888	0
Box 1	125	0
Box 2	125	0
Box 3	125	0
Box 4	125	0

Table 2 Number of interpolated points for the rotor in hover case with one layer of interpolated points

Type of points	Interpolated	Implicit	Extrapolated
Blade 1	16858	142	0
Blade 2	16858	142	0
Blade 3	16858	141	1
Blade 4	16858	141	1
Box 1	251	2	0
Box 2	251	2	0
Box 3	251	2	0
Box 4	251	2	0

Table 3 Number of interpolated points for the rotor in hover case in implicit case

1 Proc	1361906					
2 Procs	680953	680953				
3 Procs	421057	520674	420175			
4 Procs	336777	349517	337120	338492		
5 Procs	270676	272244	272734	272244	274008	
6 Procs	153223	260337	264747	155379	260337	267883

Table 4 Distribution of the number of points for the rotor-fuselage case

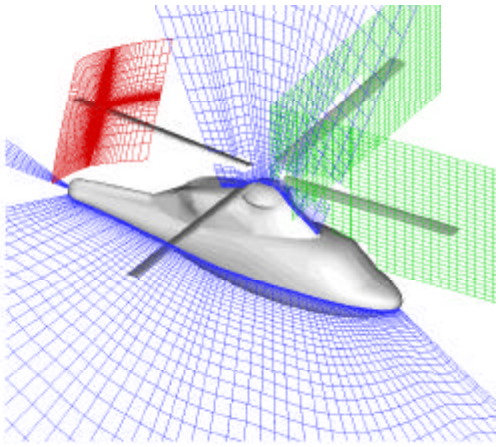


Fig. 13 Mesh for the rotor-fuselage in forward flight

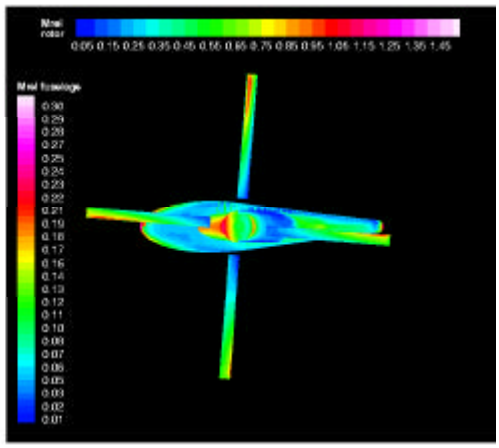


Fig. 14 Relative Mach number for the rotor-fuselage case at $\Psi = 360^o$, over view

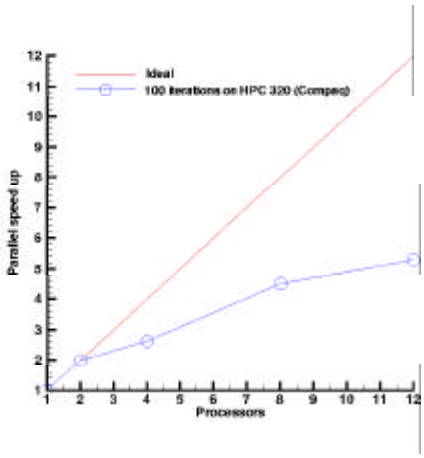
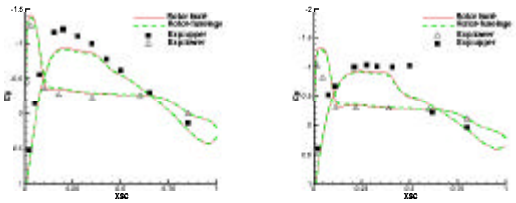
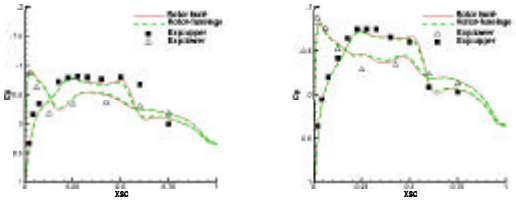


Fig. 15 Speed-up for a rotor-fuselage in forward flight

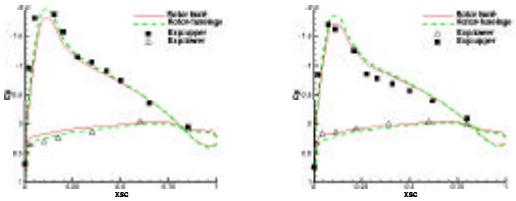


a) r/R=0.7 b) r/R=0.85

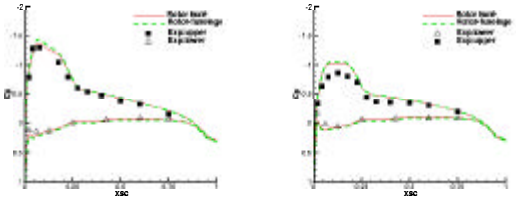


c) r/R=0.915 d) r/R=0.975

Fig. 16 Pressure coefficients around blade sections, $\Psi = 90^o$

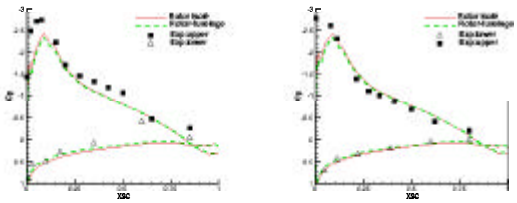


a) r/R=0.7 b) r/R=0.85

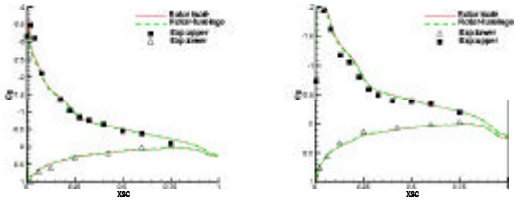


c) r/R=0.915 d) r/R=0.975

Fig. 17 Pressure coefficients around blade sections, $\Psi = 180^o$

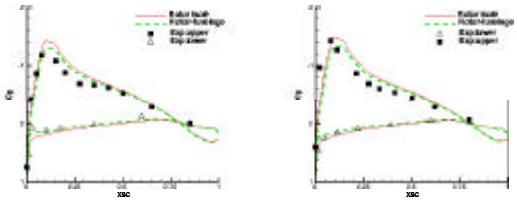


a) r/R=0.7 b) r/R=0, r/R=0.85

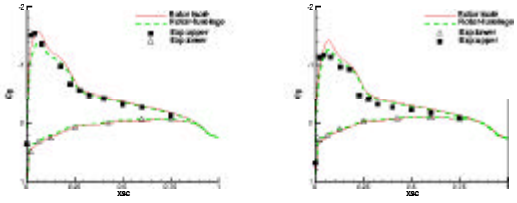


c) r/R=0.915 d) r/R=0.975

Fig. 18 Pressure coefficients around blade sections, $\Psi = 270^o$



a) r/R=0.7 b) r/R=0.85



c) r/R=0.915 d) r/R=0.975

Fig. 19 Pressure coefficients around blade sections, $\Psi = 360^o$