41st European Rotorcraft Forum
Munich, 1-4 September, 2015
Paper 133

COMPARISON BETWEEN DIFFERENT GRID METHODOLOGIES FOR ROTORCRAFT CONFIGURATIONS

Mark A. Woodgate
woodgate@liverpool.ac.uk
Research Fellow
CFD Laboratory, School of Engineering
University of Liverpool, L69 3GH, U.K.

George N. Barakos
G.Barakos@liverpool.ac.uk
Professor
CFD Laboratory, School of Engineering
University of Liverpool, L69 3GH, U.K.

Abstract

There is a wide variety of CFD grid types including Cartesian, structured, unstructured, and hybrids as well as numerous methodologies of combining grids together to reduce the time required to generate high quality grids around complex configurations. This paper presents a comparison of some of the different meshing techniques implemented within the Helicopter Multi-Block CFD method and highlights the merits of each. The paper also presents some of the challenges in developing CFD solutions that can cope with a variety of grids, and deliver efficient solutions.

1 INTRODUCTION

Structured, unstructured, or hybrid grids can be used for helicopter CFD. The grids may also use either overset or sliding interfaces to account for the relative motion between components of the helicopter or for reducing the complexity of the mesh generation process. In the literature, established helicopter CFD codes [1, 2, 3, 4, 5] have been used with several types of grids without a clear conclusion as to which mesh type is best for computations. In this paper, the idea of hybrid grids is put forward as an attempt to compromise between accuracy of solution and ease of mesh generation. The basic scheme considered, is to keep structured zones where accuracy is required (e.g. around the rotor blades) or in the rotor wake and use the unstructured parts to alleviate meshing difficulties in regions with complex geometries (e.g. complex fuselage shapes). Of course, the use of hybrid grids brings forward issues related to the solver (that now has to cope with different mesh types) and issues related to communication between different mesh types.

2 BACKGROUND

The choice of CFD grid type has a number of consequences beyond the ease of generating a high quality mesh. These include the computational cost within the CFD solver, the computational storage, and the accuracy and robustness of the employed iterative solution method.

2.1 Cartesian Grids

A Cartesian grid as shown in figure 1(a). It is a special case of a structured grid where the cells are all squares and aligned with the Cartesian axis system. They are trivial to generate and have faster flux evaluations compared to body fitted grids due to simplified mesh metrics. However, their disadvantage is the complexity in how the boundaries are treated; the cells at the boundary are cut by the real geometry, which then has to be handed via the flow solver.

2.2 Multi-Block Structured Grids

A multi block structured grid, as shown in figure 1(b), tries to take advantage of the flexibility of the unstructured grids in local refinement, and the simplicity and efficiency of structured meshes. In fact a multi-block mesh can be thought of as an unstructured hexahedral mesh. Hence the advantages of body fitted grid around the geometry can be maintained while reducing the grid generation time.

2.3 Unstructured Grids

An unstructured grid, as shown in figure 1(c) has an irregular connectivity and hence cannot be expressed as a three dimensional array meaning it requires more storage, in general. When a flow has large gradients in one direction with milder ones in the transverse directions like for example, in boundary layers, shear layers, and wakes, this can lead to highly stretched tetrahedra. This issue is normally overcome by using prism layers in these parts of the flow such that the height of the prism is much smaller than its base.

2.4 Patching Methods

There are a number of techniques for patching multiple grids together, however, none is totally satisfactory as shown in figure 2. One such method is the use of non conformal interfaces between the grids which has been used to good effect in rotorcraft aerodynamics by embedding the blades into a “drum” which allows movement with respect to the background grid [2]. In this case, the use of different point distributions on either side of a curved interface leads to gaps and overlaps, between the rotating drum grid and the stationary background grid. Hence, in general, non conformal interface schemes are only conservative when the interface is planar. The case for the non conformal interface abutting a curved surface is more problematic since it is now possible for the interpolation scheme to use information from the wrong part of the boundary. This issue was addressed in [6] by mapping the finer grids grids onto the the coarse grid representation of the curve when the interpolation weights are being calculated.

Another method is the grids to overlap one another as in the case of chimera or overset grids [7] which are also implemented in HMB2 [8]. An extension to this is the DRAGON (Direct Replacement of Arbitrary Grid Overlapping by Nonstructured) mesh is developed by Kao and Liou [9] which stitched non overlapping structured meshes for the different components using tetrahedra.

DRAGON grids have been extended by Wang et al [10] into the zipper layer method to increase the robustness in small gaps. This was without using an overlap or hanging nodes by adding a small number of tetrahedra and pyramids on either side of the interface to form a conformal mesh.

3 HMB2 HYBRID METHOD IMPLEMENTATION

The ultimate goal of any method that uses hybrid grids is to exploit the advantages while minimising the drawbacks. For example, a conformal hybrid mesh can be executed entirely within an unstructured solver with all the extra cost associated with these types of solvers. These drawbacks could be reduced by solving the body fitted grids using a multi block structured solver with the remainder solved in an unstructured solver. This methodology can be extended farther to include a high order method for solving cartesian background grids without having to extend the high order scheme to curvilinear grids. Also, it would be possible to solver a different set of governing equations in different parts of the flow. All this functionally is obtained though special treatment of the interface between two different zones so each separate solver has all the information required.

3.1 Block Structured Scheme

The Helicopter Multi-Block (HMB2) code [1, 11, 12], developed at Liverpool can solve the discretised Navier-Stokes equations in integral form using the arbitrary Lagrangian Eulerian (ALE) formulation for time-dependent domains with moving boundaries. The equations are discretised using a cell centred finite volume approach on a multi-block

grid. Osher's upwind scheme is used to discretise the convective terms and MUSCL [13] variable interpolation is used to provide third order accuracy. The Van Albada limiter [14] is used to reduce the oscillations near steep gradients. Temporal integration is performed using an implicit dual-time step method [15]. The linearised system is solved using the generalised conjugate gradient method with a block incomplete lower-upper (BILU) preconditioner.

3.2 Unstructured Scheme

The Helicopter Unstructured Method (HMB2/HUM), has been implemented using a similar scheme albeit with some differences. Firstly, the governing equations are discretised using a vertex-based approach on the unstructured mesh. Hence the solution unknowns are stored at the grid points instead of the cell centres and the cell control volume is now obtained using the dual mesh. With structured grids the second neighbours of each cell are well-defined and these are used in the spatial discretisation of the equations (e.g. in the MUSCL scheme). With unstructured grids this is not the case, and flow gradients are used instead to the same effect. The remainder of the functionally is the same but due to the differences outline above, a structured multi-block mesh will not produce identical results when run using the unstructured mode.

3.3 Coupling Methods

The method for coupling a block structured zone to a unstructured zone depends on the restrictions placed on the interface [16]. The most restrictive case is the one where the interface is required to be conformal. An example of a conformal interface is shown in figure 3(a). An interface is conformal if both the points and the edges coincide for each zone. This can either be achieved by splitting the face on the block structured side and calculate the flux on each triangle or by using a layer of pyramid on the unstructured side to transition into the tetrahedral mesh of which the later case was implemented. It should be noted that the accuracy of the interpolation could be enhanced if instead of transitioning straight away what the "halo" layer of the block structured mesh is also included in the unstructured pack. This would require that there is enough space for the halo layer and the halo mesh information and makes the generation of the unstructured zone more complex in corners as shown in figure 3(d).

A slightly less restrictive case is the one where the two interfaces coincide but the points do not as shown in figure 3(b). If the interface is curved this also leads to having both holes and overlaps in the mesh. This can be reduced greatly if approximately the same density of points are used in both interface when they have high curvature. Another option would be to pre process the grid by projecting the higher density interface onto the lower with a method similar to the one described in [6].

The least restrictive case is that of going to non matching interfaces and using a overset grid or CHIMERA method. It should be noted that not all the functionally of a CHIMERA method would be required if, for example, the grids were built in such a way that no hole cutting is required and its just a matter of finding the interpolation weights, which will now not be close to the edge of the interface, to connect the two zones together.

4 IMPLICIT FLOW SOLVER

The Navier-Stokes equations can be discretised using a vertex-centred finite volume approach. The computational domain is divided into a finite number of non-overlapping control-volumes, and the governing equations are applied to each cell in turn. The spatial discretisation of the NS equations leads to a set of ordinary differential equations in time,

$$\frac{d}{dt}\left(\mathbf{W}V\right) = -\mathbf{R}\left(\mathbf{W}\right). \tag{1}$$

where $\mathbf{W}$ and $\mathbf{R}$ are the vectors of cell conserved variables and residuals respectively. Using a fully implicit time discretisation and approximating the time derivative by a second order backward difference equation (1) becomes

$$\frac{3V^{n+1}\mathbf{W}^{n+1} - 4V^{n}\mathbf{W}^{n} + V^{n-1}\mathbf{W}^{n-1}}{2\Delta t} + \mathbf{R}(\mathbf{W}^{n+1}) = 0. \tag{2}$$

The equation is non-linear in $\mathbf{W}^{n+1}$ and cannot be solved analytically. This equation is defined to be the unsteady residual $\mathbf{R}^{\star}$. Following the original implicit dual-time approach introduced by Jameson [15] equation (2) is solved by iteration in pseudo-time $t^{\star}$. This permits the acceleration techniques of steady state flow solvers to be used to obtain the updated solution and allows the real time step to be chosen based on accuracy requirements alone

without stability restrictions. Using an implicit time discretisation on the pseudo-time $t^\star$

$$\frac{\mathbf{W}^{m+1} - \mathbf{W}^m}{\Delta t^\star} = -\frac{\mathbf{R}^\star(\mathbf{W}^{m+1})}{V^n} \tag{3}$$

where the superscript $m+1$ denotes the time level $(m+1)\Delta t^\star$ in pseudo-time. In this equation the flux residual on the right hand side is evaluated at the new time level $m+1$ and is therefore expressed in terms of the unknown solution at this new time level. The unsteady residual flux residual $\mathbf{R}^\star\left(\mathbf{W}^{m+1}\right)$ is linearised in the pseudo-time variable $t^\star$ as follows,

$$\begin{aligned} \mathbf{R}^\star\left(\mathbf{W}^{m+1}\right) &= \mathbf{R}^\star\left(\mathbf{W}^m\right) + \frac{\partial \mathbf{R}^\star}{\partial t^\star}\Delta t^\star + O(\Delta t^{\star 2}) \\ &\approx \mathbf{R}^\star\left(\mathbf{W}^m\right) + \frac{\partial \mathbf{R}^\star}{\partial \mathbf{W}}\frac{\partial \mathbf{W}}{\partial t^\star}\Delta t^\star \\ &\approx \mathbf{R}^\star\left(\mathbf{W}^m\right) + \frac{\partial \mathbf{R}^\star}{\partial \mathbf{W}}\Delta \mathbf{W}, \end{aligned} \tag{4}$$

where $\Delta \mathbf{W} = \mathbf{W}^{m+1} - \mathbf{W}^m$. and by using the definition of the unsteady residual

$$\frac{\partial \mathbf{R}^\star}{\partial \mathbf{W}} = \frac{\partial \mathbf{R}}{\partial \mathbf{W}} + \frac{3V}{2\Delta t}\mathbf{I} \tag{5}$$

Substituting the above in equation (3) and rewriting in terms of the primitive variables $\mathbf{P}$ then

$$\left[\left(\frac{V}{\Delta t^\star} + \frac{3V}{2\Delta t}\right)\frac{\partial \mathbf{W}}{\partial \mathbf{P}} + \frac{\partial \mathbf{R}}{\partial \mathbf{P}}\right]\Delta \mathbf{P} = -\mathbf{R}^\star(\mathbf{W}^m). \tag{6}$$

4.1 Jacobian Matrix

An approximate Jacobian is used to reduce the sparsity pattern of the Jacobian matrix as well as the stillness of the linear system. This is approximation take the form by differentiating the flux across the face with respect to the left and right MUSCL states instead of the solution variables. This scheme is similar to calculating the exact Jacobian matrix for a first order spatial discretisation, with the modification that the MUSCL interpolated values at the interface are used in the evaluation rather than the cell values that would be used for a first order spatial scheme. In fact these approximations are exact for a first order spatial discretisation.

In the multi-block structured case, this approximate Jacobian has 7 non zero entries per row. In the unstructured case for a hexhedral mesh this is still true since each vertex has 6 edges. However for a general unstructured mesh using a vertex centred scheme this will increase. For example the figure 4 shows a histogram of the number of edges per point for the ERICA fuselage case of section (5.4). Most have 9 non zero terms per row since this occurs from prisms which are grown from a surface mesh where 6 triangle met at a point. The average number of terms per row in this case is 10.52 with 6 rows having 24 non zero terms. This increase from 7 to 10.5 increase the computational cost of a matrix vector multiply by 50% however the cost is farther increased due to the non fixed distance between the columns of any given row which now have to be accessed from memory - since a sparse matrix vector multiply is memory bandwidth limited anything that increase loads from memory reduces the performance.

4.2 Low Memory Solver

The lower memory solver is based on splitting the Jacobian matrix into the lower diagonal and uppers parts.

$$A = L + D + U = (L + D)D^{-1}(U + D) - LD^{-1}U$$

where L only consists of block terms which are strictly below the diagonal, U only consists of block terms which are strictly above the diagonal and D is a block diagonal matrix. If it is assumed that the $LD^{-1}U$ is small and hence can be neglected then the following equation can be used to solve for the solution update.

$$(L + D)D^{-1}(U + D)\Delta W^n = -R(W^n). \tag{7}$$

Equation 7 can be inverted by using a forward and backward sweep procedure:

$$\begin{cases} D\Delta W^\star = -R(W^n) - L\Delta W^\star \\ D\Delta W^n = D\Delta W^\star - U\Delta W^n \end{cases} \tag{8}$$

where $\Delta W^\star$ is the solution vector updated in the forward sweep. The property of the Jacobian matrix has a strong influence on the performance of this scheme. There are a number of ways to set up the Jacobian matrix, in the original LU-SGS scheme, the matrix was constructed based on splitting the inviscid flux Jacobian according to its spectral radius. This treatment reduced both the computational complexity of the scheme and help to ensure the matrix is diagonally dominant [17, 18]. However in the case of HMB2 the Jacobian matrix is the same as the one used in section 4.1. This means that the memory required to store the preconditioner and the generalised conjugate residual solver bases are not required, this is at the expense of taking more iterations to solve the linear system, usually with a smaller CFL number to keep the system more diagonally dominant.

4.3 Parallel Implementation

The parallel implementation of the Helicopter Unstructured Method (HMB2/HUM) is done though partitioning the domain into smaller pieces using METIS [19]. This is a fast processes taking less than a 20 seconds for a mesh of 3.5 million cells. Since the process is linear in the number of cells even meshes of 100 million cells are not CPU depended. A larger limiting factor is the amount of memory needed to partition the mesh. This again is linear in the number of cells and currently about 1.4 Million cells per GByte. Hence a 100 million cell mesh would take around 10 minutes of CPU time but also around 70 GBytes of memory.

Since the grid has been partitioned into multiple domains information must be passed between processors. Unlike the structured solver in which is a double row of halo cells is passed the unstructured solver needs a single row of both the solution and its spatial gradients. This is double the amount of information since the gradient requires three times the information and the gradient of the temperature is also communicated.

4.4 Parallel Performance

The parallel performance is shown in table 1. As the number of processors doubles so is the number of iterations and hence if the code was 100% efficient the time would remain the same. The test problem had 3.55 million cell and was tested using the explicit two equation turbulence formation with MUSCL extrapolation, limiting was turned on and the gradients calculated using weighted least squares. There are two factors affecting the parallel performance. First, as the mesh is partitioned into smaller parts the ratio of internal points to halo points gets larger and so there is less computation per Mbyte of communication. The second factor is that as more processes are running on the same node there are less resources per processor. For example the fixed memory band width has to spread across more processors and hence can be a limiting factor in the parallel efficiency of the code. Taking both facts into account the 20% drop off is a good result. To assess the effect the limited memory bandwidth a second configuration was run where each node has just two processes and the messages sent via MPI where sent over a Gigabit Ethernet network instead of using the shared-memory Byte Transfer Layer which is used for communications within a node. This means that each process will have more memory bandwidth at the expense of a slower transport layer. As can be seen from table 1 there is a small increase in efficiency running with less processes per node. In an environment where processes within a node have much higher bandwidth and lower latency when communication across nodes the assignment of partitioned mesh to nodes is very important. Using the same test problem and running on 8 processes each on two nodes it possible to increase the wall clock time from the 1415 seconds from table 1 to 1740 seconds. In this case the amount of data communicated between the nodes increased a factor of 7 causing a marked slowdown. Ideally, the grid partitioning process should be done in two stages firstly the message between nodes should be minimised by partitioning on the number of nodes being used and then each of these partitions should be re-split based on the number of processes per node to minimise the amount of traffic over the slower communication layer.

5 NUMERICAL RESULTS

The following section first examines the difference performance between HMB2 and HMB2/HUM, the effectiveness of the Jacobian matrix for the unstructured scheme as well as two ad three dimensional computational results.

5.1 Performance Comparison Between HMB2 and HMB2/HUM

An important aspect of any CFD code is its computational performance and where the code spends most of its time. Codes that spend a large percent of their time in only a couple of subroutines are amenable to performance enhancement. The GCC compiler was used. The code was compiled with `-O2` optimization instead of the normal `-O3` so as to stop the compiler from in-lining certain small functions, making the timings within functions correct at the expense of an increased number of function calls. The table 2 below shows the top 20 function calls within the explicit part of the HMB2/HUM code. As can been seen some 70% of the runtime of the code is in 3 functions.

From the number of calls to any given function there are three different types. The ones that are called every iteration, for example, `nsSolverInit` which initialises the Navier Stokes solver, a function that are called number of cells times per iteration which is used to calculate the source term of the two equation turbulence

model, `sourceTermKOmega`, and finally, functions that are called number of edges times per iteration, for example the viscous flux calculation, `viscousGradResidual`. There is a minor difference between the number of calls to Osher's approximate Riemann solver and the number of viscous flux calls because the Osher solver is reused for some boundary conditions while the viscous fluxes has a specialised routine. It should be noted that a different type of mesh changes the ratio of the number of edges to the number of cells in the computations. The calculation of the gradient by least squares is the highest cost function since the current version recalculates the matrix every iteration. The limiter values are calculated and stored so that can be used in both the Navier-Stokes and two equation turbulence models. It only consists of two loops over the edges neither of which are very computationally expensive, however, there is a large number of conditional statements, four in each loop which have fairly random branching making the routine slow. After these two routines it is better to look at the cost of all the functions that make up the inviscid, viscous, and two equation turbulence model part of the computation. The inviscid part is the summation of the two functions shown in red in table 2 and most of the setup in `viscousResidualUns`. This totals around 14% if the runtime of the code. This is nearly exactly the same as the viscous parts shown in blue. While the two equation turbulence model parts (in green) total around 10%. This is a lot different to the structured multi-block code where the turbulence models take longer to calculate. Table 3 shows this via the relative timings of an explicit residual, of a $750K$ cell three dimensional sphere case. The evaluation of the second order Euler scheme in the structured code has been scaled to unity to obtain the relative performance of the structured and unstructured zones solving three different sets of governing equations. The increase in cost between the two unstructured versions is all down to the extra cost of the gradient calculation. There is a much larger increase in computation cost going to the Navier-Stokes equations in the structured method since now the gradient is needed where its already required for the second order unstructured MUSCL. The unstructured solver has a more effective viscous calculation since it done at the same time as the inviscid part where as it is done in two loops in the structured code. The increase in going to the k-ω turbulence model for the unstructured code is due to the use of extra storage which is used to reduce the repeated computation seen the structured code.

5.2 Performance of the Linear Solvers

The performance of the linear solvers were compared using a NACA0012 aerofoil at 10 degree angles of attack at a Mach number of 0.3 using the HMB2/HUM solved using the the first order spatial discretisation of the Euler equations. Figure 5(a) shows the rate of convergence of the generalised conjugate gradient with respect to the CFL number. As the CFL is increased the solution method tends towards Newton performance and takes very few iterations to converge to machine accuracy. Figure 5(b) shows the comparison of the Gauss-Seidel method of solving the linear system and the generalised conjugate gradient. With a tolerance of 0.1, the updates have converged to the first significant figure, the computational cost of the Gauss-Seidel method for 600 iterations was only two percent higher. Reducing the tolerance to 0.05 increased this cost by another 30%. It should noted that the computational cost of each Gauss-Seidel inner iteration is about one third of each inner iteration in the generalised conjugate gradient method. Adding in the fact that preconditioner is also not required around 4 to 5 times the number of Gauss-Seidel steps are possible compared to the generalised conjugate gradient method. It can also be seen even at the higher tolerance the drop in the residual is lightly smoother for the Gauss-Seidel method.

5.3 Two Dimensional Cases

Two two-dimensional cases are presented here. The first is subsonic flow around the NACA0012 aerofoil at zero angle of attack at a Mach number of 0.3. This flow was solved on three different computational grids, one comprising only of hexahedra, one of only tetrahedra and the final one a mixed mesh. This meshes were constructed so the grid points were always in the same position only the connectivity between them was changed. As can been seen from figure 6 all three answers are symmetric and very similar. The main differences can been seen at the interface as the hexahedra where cut in different directions and in the hybrid mesh at the intersection between the hexahedra and tetrahedra. The second example is the NACA0012 aerofoil at 1.25^o angle of attack and

mach number 0.8 and the results can be seen in figure 7. It shows both a limited and unlimited solution. The limiter clips with respect to all the gradients connected to a grid point and hence is overly dissipative since its also get turn on when the edge is parallel to the discontinuity. This is can seen by the change in the weak shock on the lower surface which was captured correctly with the unlimited solution.

The next test case is a coarse grid of 37 thousand nodes and was used to calculated the flow around a cylinder at 10^o degree angle of attack, Mach number of 0.3 and Reynolds number of 1000. Since the calculation was run second order the flow is in fact unsteady. Figure 8 shows the results from a steady state solve at the point where the flow has just become asymmetric and the vortex will start shedding behind the cylinder. One can clearly see the asymmetry in the V-Velocity while the pressure coefficient still looks nearly symmetric. Figure 9 shows the results of the same configuration run as an unsteady problem with a non-dimensional time step of 0.02. This size of non-dimensional time step was used as the grid is quite coarse. Even with this coarse grid it is possible to see the pair of vortices begin to be shed behind the cylinder. It is again easier to pick these up in the variation of the V-velocity than the pressure coefficient.

5.4 Three Dimensional Cases

The first fuselage case considers inviscid flow around the ROBIN [20, 21, 22] body. The mesh contained just over 1 million vertices and 5.7 million tetrahedra, there was no prism layer in this case. There were a total of 414 thousand surface triangles of which nearly all of them where on the fuselage since the grid was not a half model. The surface mesh density can be seen in figure 10 (a) and (b) for the dog house and nose respectively. The mesh was split onto 8 processors shown in figure 10 (c) with the surface pressure shown in (d).

The second case is for inviscid flow around the ERICA fuselage [23] shown in figure 11. The grid contains 5.3 million nodes and 15.9 million cells of which 8.5 million are tetrahedra and 7.3 million are prisms in the prism layer which is 12 cells high. There are 718K surface triangular elements, of which more than 600K are on the fuselage, and 37 surface quadrilateral elements which is the prism layer cells cut by the symmetry plane. As can been seen from the surface mesh in figures 11 (b) and (c) there are a large number of points on the fuselage which is one of the main advantages of using an unstructured mesh. The calculation was run on 16 processors and the CPU partition can be seen in figure 11 (d) while the pressure coefficient can be seen in figure 11 (e).

6 CONCLUSIONS AND FUTURE WORK

The ability to solver unstructured and hybrid mesh has been added to HMB2/HUM increasing the functionally of the Liverpool helicopter CFD code. Initial calculation have been carried out and the results and performance look promising. Many of the coding improvements found within the unstructured code will be back ported into the HMB2. The lower memory linear solve has already been implemented and is currently under testing while work on improving the accuracy of the gradient evaluation, and the performance of both the residual and Jacobian calculation will be carried out at a later stage.

7 ACKNOWLEDGMENTS

This work is funded under the HiperTilt Project of the UK Technology Strategy Board (TSB) and AugustaWestland (AW). The authors would like to acknowledge the use of the NICETRIP model geometry.

8 REFERENCES

[1] Barakos, G., Steijl, R., Badcock, K., and Brocklehurst, A., "Development of CFD Capability for Full Helicopter Engineering Analysis," *Proceedings of the Thirty-First European Rotorcraft Forum*, ERF, Florence, Italy, 13-15 September 2005.

[2] Steijl, R. and Barakos, G., "Sliding Mesh Algorithm for CFD Analysis of Helicopter Rotor-Fuselage Aerodynamics," *Int. J. Numer. Meth. Fluids*, Vol. 58, 2008, pp. 527–549.

[3] Puigt, G., Gazaixy, M., Montagnac, M., le Papey, M.-C., Platay, M., Marmignony, C., Boussuge, J.-F., and Couailliery, V., "Development of a new hybrid compressible solver inside the CFD elsA software," 20th AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, USA, June 27-30 2011, AIAA 2011-3379.

[4] Schwamborn, D., Gerhold, T., and Heinrich, R., "The DLR TAU-Code: Recent Applications in Research and Industry," European Conference on Computational Fluid Dynamics, TU Delft, Egmond aan Zee, The Netherlands, September 5-8 2006.

[5] Sitaraman, J., Potsdam, M., Wissink, A., Jayaraman, B., Datta, A., Mavriplis, D., and Saberi, H., "Rotor loads prediction using helios: A multisolver framework for rotorcraft aeromechanics analysis," *Journal of Aircraft*, Vol. 50, No. 2, 2013, pp. 478–492.

[6] Woodgate, M. and Barakos, G., "Rotor computations with active gurney flaps," 38th European Rotorcraft Forum, Amsterdam, Netherlands. 4-7 September, 2012.

[7] Steger, J. L., Dougherty, F., and Benek, J. A., "CHIMERA GRID SCHEME." Vol. 5, 1983, pp. 59–69, cited By 65.

[8] Jarkowski, M., Woodgate, M., Barakos, G., and Rokicki, J., "Towards consistent hybrid overset mesh methods for rotorcraft CFD," *International Journal for Numerical Methods in Fluids*, Vol. 74, No. 8, 2014, pp. 543–576, cited By 3.

[9] Kao, K.-H. and Liou, M.-S., "Advance in overset grid schemes: From Chimera to DRAGON grids," *AIAA journal*, Vol. 33, No. 10, 1995, pp. 1809–1815, cited By 28.

[10] Wang, Y., Qin, N., Carnie, G., and Shahpar, S., "Zipper layer method for linking two dissimilar structured meshes," *Journal of Computational Physics*, Vol. 255, No. 0, 2013, pp. 130 – 148.

[11] Badcock, K., Richards, B., and Woodgate, M., "Elements of computational fluid dynamics on block structured grids using implicit solvers," *Progress in Aerospace Sciences*, Vol. 36, No. 5, 2000, pp. 351–392.

[12] Steijl, R., Barakos, G., and Badcock, K., "A framework for CFD analysis of helicopter rotors in hover and forward flight," *International Journal for Numerical Methods in Fluids*, Vol. 51, No. 8, 2006, pp. 819–847.

[13] van Leer, B., "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101–136, cited By 2754.

[14] Van Albada, G., Van Leer, B., and Roberts Jr, W., "A comparative study of computational methods in cosmic gas dynamics," *Astronomy and Astrophysics*, Vol. 108, 1982, pp. 76–84.

[15] Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," 10^{th} *Computational Fluid Dynamics Conference*, Honolulu, HI, 24-26 June 1991, AIAA-1991-1596.

[16] Vanharen, J., Puigt, G., and Montagnac, M., "Theoretical and numerical analysis of non-conforming grid interface for unsteady flows," *Journal of Computational Physics*, Vol. 285, No. 0, 2015, pp. 111 – 132.

[17] Yoon, S. and Jameson, A., "Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations," *AIAA journal*, Vol. 26, No. 9, 1988, pp. 1025–1026.

[18] Chen, R. and Wang, Z. b., "Fast, block lower-upper symmetric Gauss-Seidel scheme for arbitrary grids," *AIAA journal*, Vol. 38, No. 12, 2000, pp. 2238–2245.

[19] Karypis, G. and Kumar, V., "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, Vol. 20, No. 1, 1998, pp. 359–392.

[20] Freeman, C. E. and Mineck, R. E., "Fuselage surface pressure measurements of a helicopter wind-tunnel model with a 3. 15-meter diameter single rotor," Tech. Rep. TM-80051, NASA, 1979.

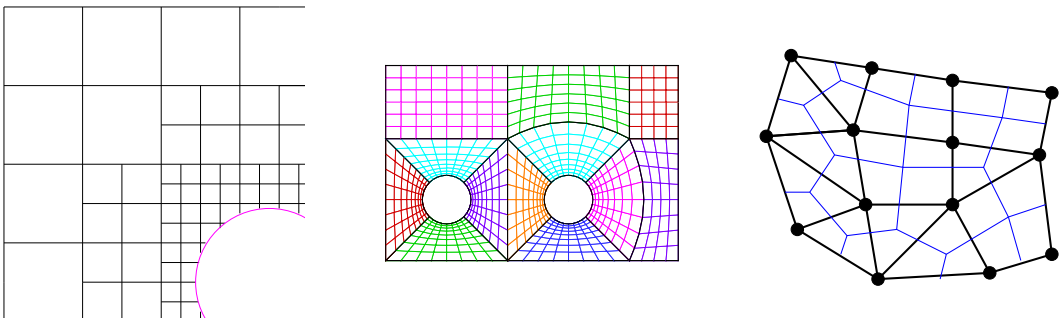
[21] Mark S., C. and John D., B., "Navier-Stokes and Potential Theory Solutions for a Helicopter Fuselage and Comparison With Experiment," Tech. Rep. TM-4566, NASA, 1994.

[22] Mineck, R. E. and Gorton, S. A., "Steady and periodic pressure measurements on a generic helicopter fuselage model in the presence of a rotor," Tech. Rep. TM-2000-210286, NASA, 2000.

[23] Stabellini, A., Verna, A., Ragazzi, A., Hakkaart, J., De Bruin, A., Hoeijmakers, A., Schneider, O., Przybilla, M., Langer, H.-J.,

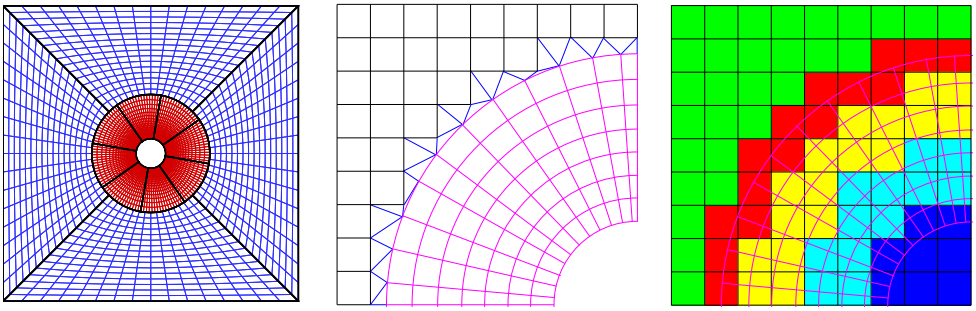
and Philipsen, I., "First NICETRIP powered wind tunnel tests successfully completed in DNW-LLF," Vol. 1, 2014, pp. 55–71.

[24] Venkatakrishnan, V., "Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters," *Journal of Computational Physics*, Vol. 118, No. 1, 1995, pp. 120 – 130.



(a) Cartesian Grid (b) Multi Block Structured Grid (c) Unstructured Grid

Figure 1: Different types of grid.

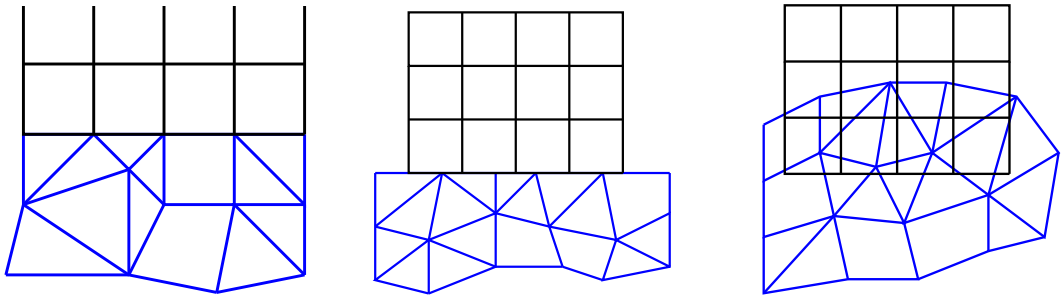


(a) Sliding planes Grid (b) DRAGON Grid (c) Chimera Grid

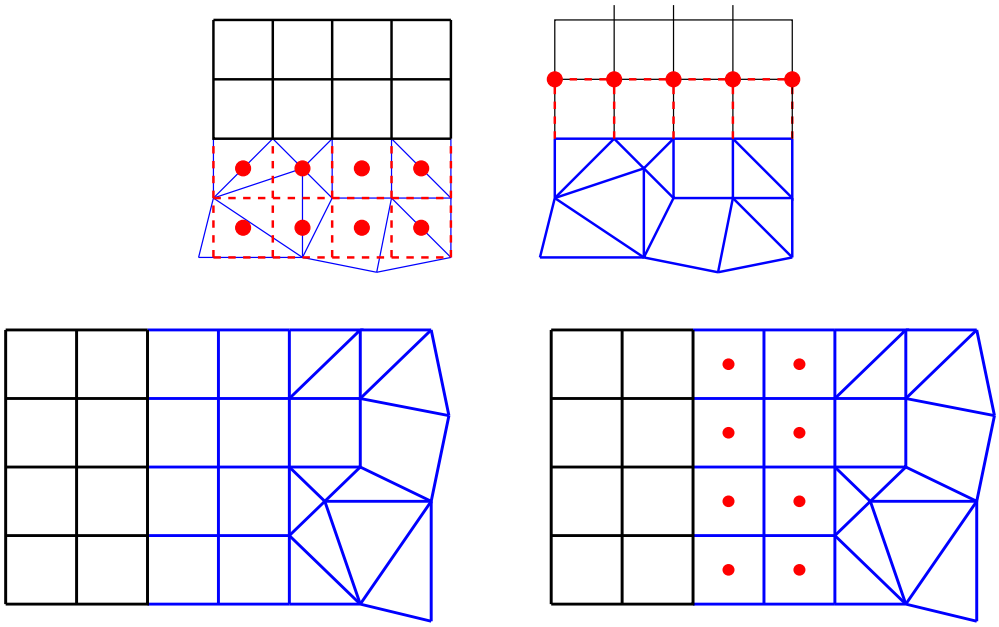
Figure 2: Different patching methods.

	One Node Only		Two processes per Node	
Number of Processors	Wall clock time	Efficiency	Wall clock time	Efficiency
1	19m 20s	100.0%	19m 20s	100.0%
2	19m 27s	99.4%	19m 27s	99.4%
4	19m 56s	97.0%	19m 37s	99.4%
8	21m 12s	91.2%	20m 41s	93.4%
16	24m 23s	79.3%	23m 35s	82.0%

Table 1: CPU time and memory usage for partitioning different size meshes onto 4 processors.



(a) Conformal Patching (b) Non Conformal Patching (c) Fully non matching



(d) Interpolation strategy with a double layer of hexahedrons in the unstructured grid.

Figure 3: Different Coupling Methods and interpolation strategies.

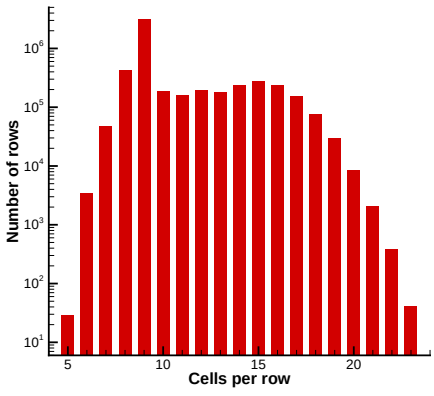


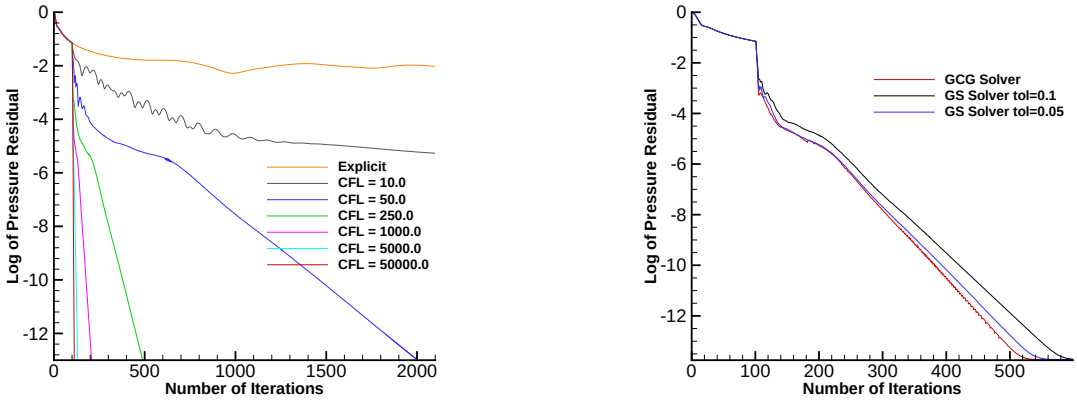
Figure 4: Histogram of the number of non zero per row in the approximate Jacobian matrix for a 5.3 million node unstructured Erica fuselage mesh used in section 5.4.

Name of Function	Number of Calls	Percentage time
leastSquaresGradsUns	600	40.09
calculateLimiter	600	16.04
viscousGradResidual	1423104000	11.35
osher	1437849600	7.97
viscousResidualUns	600	5.70
calcLocalTimeStepUns	601	4.95
viscousGradResidualKOmega	1423104000	3.17
turbulentResidualUns	600	2.92
upwindKOmega	1423104000	1.35
exp_flux	1437849600	1.35
sourceTermKOmega	475700400	1.03
updateSolutionUns	600	0.92
calcLamViscosity	602	0.64
subResidual	1423104000	0.56
calcEddyViscosity	602	0.38
scaleResidualUns	600	0.36
subResidual_2eq	1423104000	0.31
addResidual	1423104000	0.30
addResidual_2eq	1423104000	0.20
nsSolverInit	600	0.18

Table 2: CPU time for the top 20 subroutines for the explicit residual evaluation when modelling a turbulent flow with the $k - \omega$ two equation turbulence model.

Discretising equations	Structured	Unstructured Gauss Green	Unstructured Least Squares
Euler	1.00	1.31	2.16
Laminar NS	1.92	1.87	2.59
Turbulent NS	3.19	2.30	3.22

Table 3: Computation comparison between the structured and unstructured solver normalized with respect to the computational cost of the Euler equations in the structured code HMB2.



(a) (b)

Figure 5: Rate of convergence of the implicit scheme for Euler flow around a NACA0012 at 10^o angle of attack and mach number 0.3. Figure (a) shows the can in convergence of the generalised conjugate gradient scheme with respect to CFL number while (b) shows the comparisons of the generalised conjugate gradient scheme and the Gauss-Seidel method at CFL number 250.

.

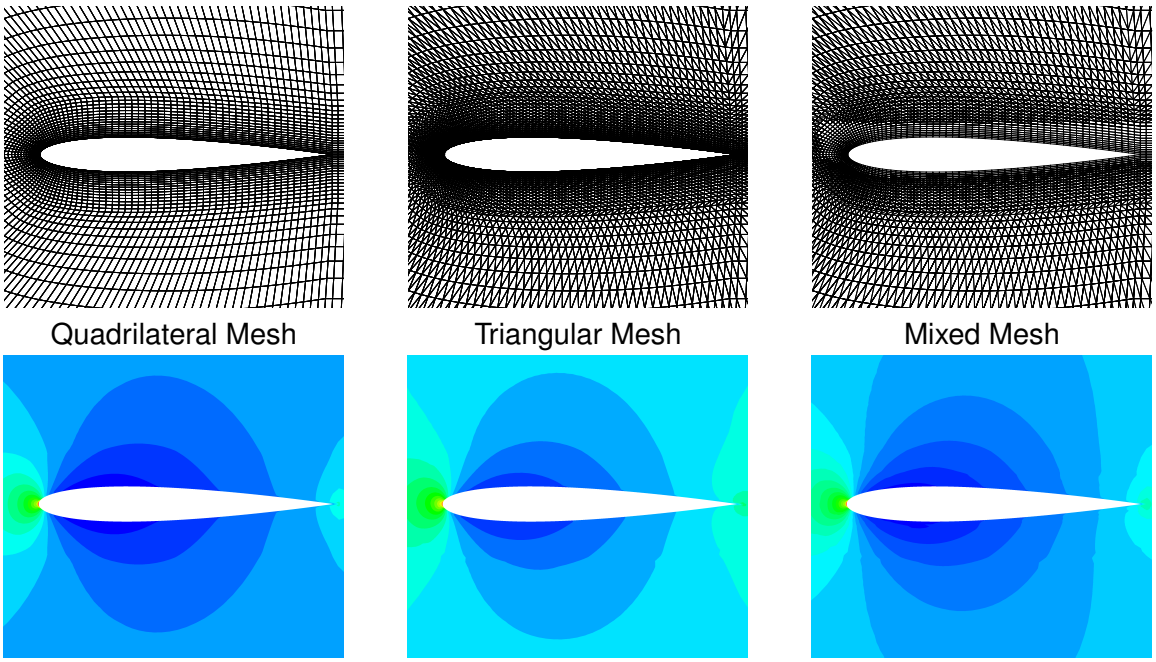
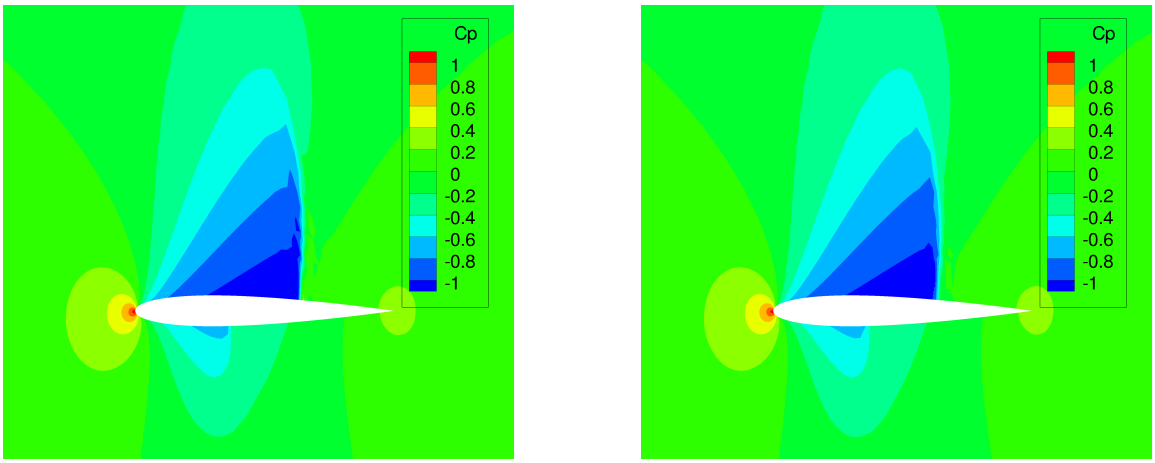


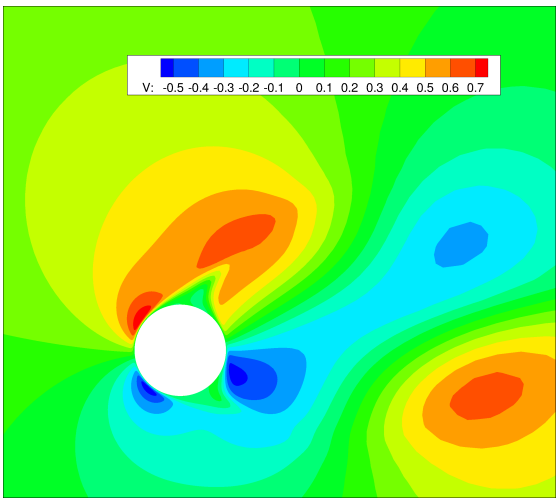
Figure 6: The Mesh and Cp for Euler flow around a NACA0012 at zero degrees incidence and Mach number 0.3.



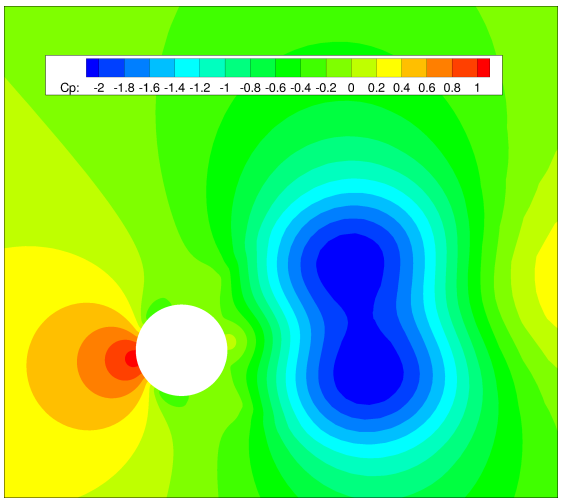
(a) (b)

Figure 7: Cp for Euler flow around a NACA0012 at 1.25^o angle of attack and mach number 0.8. Figure (a) has no limiter while (b) is limited with Venkatakrishnan limiter [24].

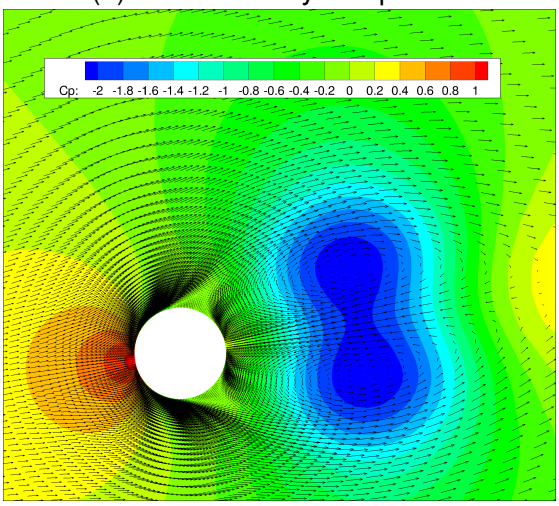
.



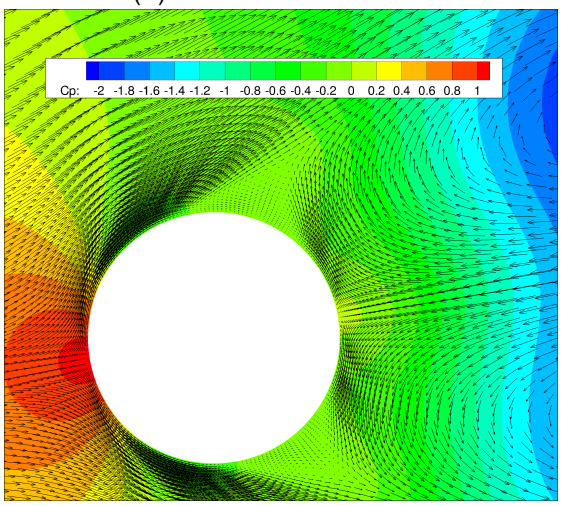
(a) The V-velocity component



(b) Pressure coefficient



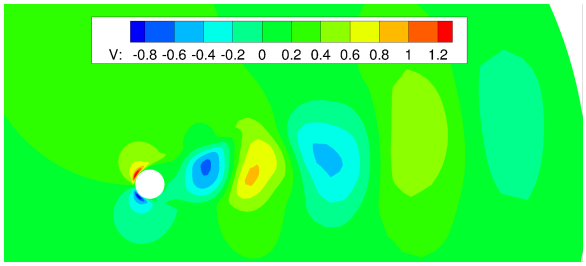
(c) Velocity vectors



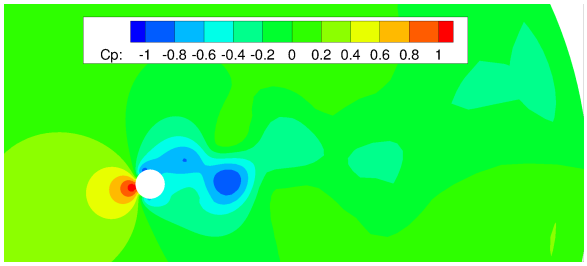
(d) Velocity vectors

Figure 8: Second order steady laminar flow past a cylinder with angle of attack 10^o and a Reynolds number of 1000.

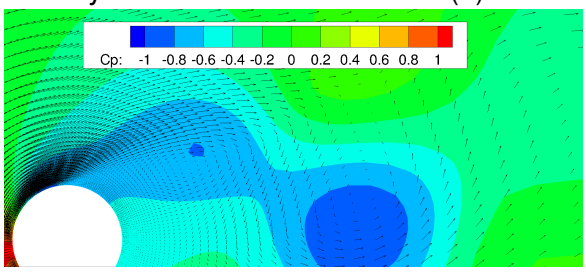
.



(a) The V-Velocity

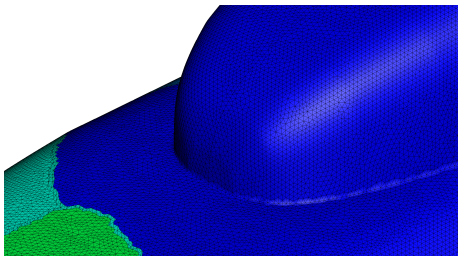


(b) Pressure coefficient

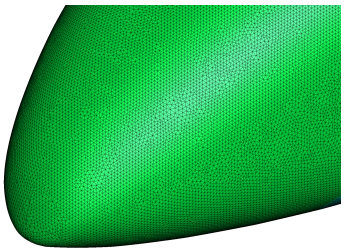


(c) Velocity Vectors

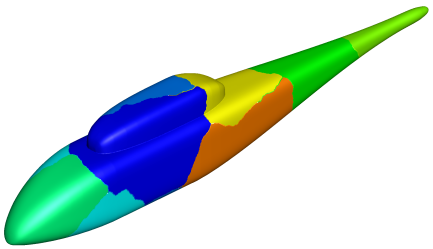
Figure 9: Second order unsteady laminar flow past a cylinder with angle of attack 10^o and a Reynolds number of 1000.



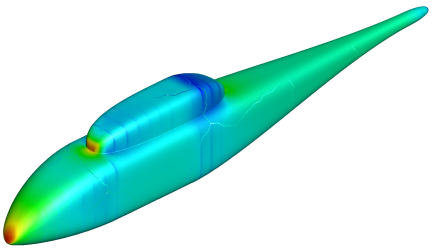
(a)



(b)

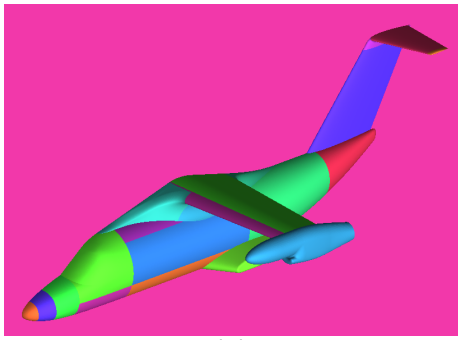


(c)

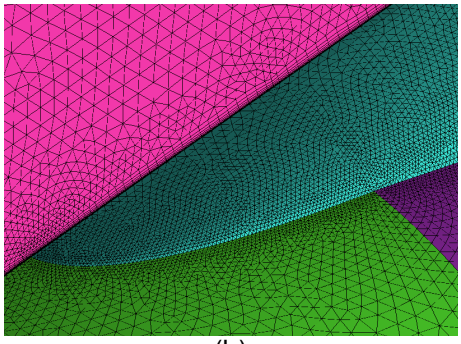


(d)

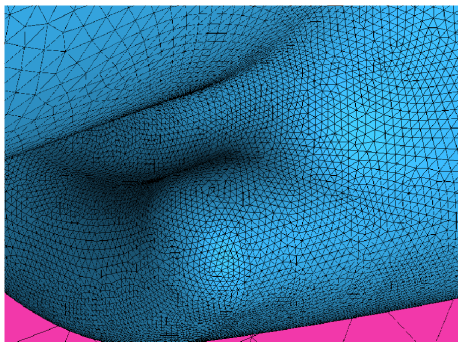
Figure 10: Surface mesh, processor partition and pressure solution for the ROBIN fuselage for zero angle of attack and 0.3 Mach number.



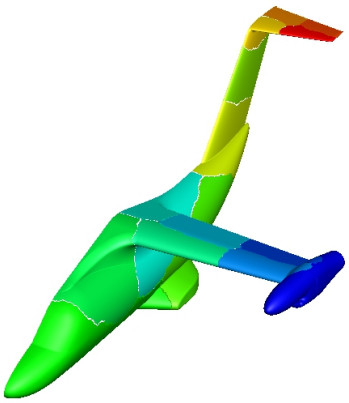
(a)



(b)



(c)



(d)

(e)

Figure 11: Surface mesh, processor partition and Cp solution for the ERICA fuselage for zero angle of attack and 0.3 Mach number.