Development of a new hybrid compressible solver inside the CFD elsA software

Guillaume Puigt* Michel Gazaix† Marc Montagnac* Marie-Claire Le Pape†

CERFACS ONERA CERFACS ONERA

Marta de la Llave Plata† Claude Marmignon† Jean-François Boussuge*

ONERA ONERA CERFACS

Vincent Couaillier†

ONERA

A new fully integrated environment for compressible flow simulation on structured and unstructured zones coexisting within the same hybrid mesh is presented. This environment originates from the existing elsA a CFD software tool, developed by ONERA and CERFACS for structured grids, and widely used both in industry and research.

Some configurations can not be easily addressed with a fully structured mesh approach. An efficient way to overcome this drawback is to combine structured and unstructured zones in a single hybrid mesh. Structured zones are kept for sake of accuracy in boundary layers, wall clock efficiency and low memory consumption. Unstructured zones enable an easier mesh generation / adaptation process.

This paper describes the extension of elsA to unstructured multi-element zones composed of hexahedra, tetrahedra, prisms and pyramids. A key feature is the treatment of structured / unstructured zone interfaces, performed with a mismatched abutting grid interface algorithm.

We demonstrate that the Object-Oriented (OO) programming approach is useful and efficient to integrate unstructured data structures and numerical methods into the original software to form a single computational CFD kernel.

Software design, numerical algorithm, structured/unstructured block interface treatment, CPU efficiency and parallel scalability, are discussed. Finally, validation examples demonstrating the project status are given.

I. Introduction

Most Computational Fluid Dynamics (CFD) codes may be classified into two branches depending on the meshing methodology b, either structured or unstructured. Both mesh types have their specific advantages.

The main advantages of structured grids are, first, their ability to capture the anisotropy of boundary layer regions, where high-aspect ratio cells are required, and, secondly, structured numerical techniques helping to improve performance and robustness (for example implicit smoothing operator, or the structured way of coarsening meshes for the multigrid procedure). Over the years, engineers have defined best practices to build such meshes on relatively complex CAD model. Since the mesh generation can be complex and time-consuming, large efforts have been dedicated to ease the meshing process. Widely used approaches are chimera grids,1 or mismatched abutting interfaces2 (in CGNS3 notation4) between blocks. These techniques

* CERFACS, Computational Fluid Dynamics, 42 avenue Coriolis, 31057 TOULOUSE CEDEX (FRANCE).
† ONERA, Computational Fluid Dynamics and Aeroacoustics, BP72 29 avenue de la Division Leclerc 92322 CHÂTILLON CEDEX (FRANCE).
1 ensemble logiciel pour la simulation en Aérodynamique
2 a mesh is defined as the complete set of individual grid associated to each zone (CGNS notation)
3 or overset
4 also called patched grid
are now mature, and a huge amount of experience is available. Note also that several recent workshops\textsuperscript{4,5} have demonstrated faster mesh convergence when using structured mesh, specially for high-Reynolds number flows. However, the structured grid generation for complex geometries\textsuperscript{6} is a time-consuming task due to the possible need of breaking the domain manually into several blocks.

For really highly complex CAD models, a common belief inside the CFD community is that is is easier to switch to unstructured mesh technology. However, the process being generally more or less automatic, solution accuracy may be reduced due to the presence of skewed elements in sensitive regions like boundary layers. A way to overcome this drawback is to authorize unstructured grids composed of different element shapes with hexahedra and prisms in the anisotropic flow region, tetrahedra elements where the flow is isotropic, and pyramids in the buffer region from four-node faces to three-node element faces. In the following, such multi-element grids are called mixed-element grids\textsuperscript{7}. Their main drawbacks, however, are the memory overhead associated with storing grid connectivity information and the computer time associated with indirect addressing. In addition, due to the lack of grid structure, it is difficult to implement simple implicit schemes such as approximate factorization, while explicit schemes suffer from slow convergence.

It is therefore tempting to try to associate both techniques, in order to benefit from the advantages of both techniques, and hopefully avoiding most of the deficiencies of each approach. Moreover, to reduce development costs as well as to minimize end user’s learning time, it would be desirable to have a single code being able to handle structured, unstructured and hybrid meshes.

Thus, this paper presents a new unstructured solver, together with the coupling algorithm with the structured regions. This unstructured solver is incorporated inside the CFD tool els\textsuperscript{A},\textsuperscript{6–9} developed by ONERA since 1997 and co-developed by CERFACS since 2001,\textsuperscript{10} and used by a large user community in both research and industry. The extended els\textsuperscript{A} tool can take into account simultaneously both structured and unstructured zones. The structured and unstructured solvers share the same CFD kernel, avoiding most code duplication with the associated maintenance nightmare. Structured zones benefit from the higher accuracy, numerical efficiency and better usage of modern processor memory cache, while unstructured zones allow to keep the meshing process in the most complex regions manageable.

The development in els\textsuperscript{A} of hybrid mesh capabilities relies on a strong cooperative effort between ONERA and Cerfacs.\textsuperscript{11}

To the authors’s knowledge there are up to now no fully satisfying mesh generation software handling fully hybrid structured/unstructured mesh. Moreover, based on our experience in structured mesh generation, it is likely that the different mesh regions will be generated nearly independently in order to manage the geometric complexity. That means that at some point these different parts will have to be glued. In that case, ensuring node coincidence along the interface would be an unbearable constraint, since giving a node distribution coming from a structured mesh generation software to an unstructured one is generally not possible. Conversely, the latter can easily use a typical cell dimension given by the structured mesh generation software. Besides, huge meshes will hardly be generated in one shot due to memory consumption of the generation tool, even if this is likely to evolve with advances in these tools. Either they will be generated on the fly by a mesh refinement process or some parts will again be generated independently. All these reasons have confirmed the strategy of using mismatched abutting grid boundaries. It has also been preferred to the chimera technique to ensure the fundamental finite volume flux conservation property\textsuperscript{8}. The mismatched abutting interface will be also the backbone of the forthcoming sliding mesh functionality.

The remainder of this paper is organized as follows. In section II, we review some related works. Section III describe the numerical unstructured solver. Details of the software design are given in section IV, followed by a discussion of the mismatched abutting interface algorithm, section V. Section VI discuss the solver efficiency and parallel scalability. Finally several validation examples are given in section VII.

### II. Related work

Our choice of a dual-mesh paradigm, implemented with a fully-integrated software environment, has some common features with recently published work. The main differences between these computational tools are the coupling algorithm between grids – exact matching interface, abutting interface or chimera –, and the

---

\textsuperscript{6}such as a landing gear configuration, or a turbine blade with hundreds of cooling holes  
\textsuperscript{7}we prefer mixed-element grid to hybrid grid in order to avoid confusion with hybrid mesh  
\textsuperscript{8}it is also planned in future release to implement the chimera technique between structured and unstructured grids to allow even more flexibility in the meshing process, in cases where conservativity is not essential
software strategy – fully integrated tool versus loosely-coupled solvers.

The pioneering work of Lefebvre\textsuperscript{12} at ONERA demonstrated the versatility of hybrid adaptive methods for the simulation of turbulent flows.

Soetrisno et al.\textsuperscript{13} described an explicit parallel coupling between matching structured and unstructured (tetrahedra only) grids.

At DLR, a prototype, ”TAU-ijk”, has been developed from the widely used unstructured DLR TAU code,\textsuperscript{14–16} and includes ingredients from the block-structured FLOWer code such as implicit smoothing techniques, which are applied in the structured regions of the computational domain.

Also at DLR, the TRACE code\textsuperscript{17,18} was built by coupling an existing structured solver with a newly developed unstructured-grid module via a conservative hybrid-grid interfacing algorithm; TRACE is mostly used for turbomachinery applications.

Katz et al\textsuperscript{19} investigated a hybrid mesh strategy that couples unstructured grids for the ”near-body” region, cartesian structured grids for the ”off-body” region, and a meshless solver in between.

Helios,\textsuperscript{20,21} a high-fidelity rotorcraft simulation software, utilizes a heterogeneous meshing paradigm consisting of unstructured near-body grids and Cartesian off-body grids. A Python-based infrastructure is used to combine the different solver components: the off-body structured solver, SAMARC (a combination of the SAMRAI meshing software and the Cartesian version of the ARC3D solver), and near-body unstructured solver, NSU3D; the two mesh systems are overlaid on top of each other using an overset domain connectivity formulation.

DRAGONFLOW\textsuperscript{22,23} is a programme suite coupling two existing successful flow solvers: OVERFLOW and USM3D, operating on ”DRAGON” meshes; both OVERFLOW and USM3D are presented in form of module libraries, and a master module controls the invoking of these individual modules.

III. Numerical description of the new unstructured solver

In this section, we first briefly recall \textit{elsA} features for structured grids. The unstructured discretization techniques for convection and diffusion are then discussed. Finally, the numerical implementation of RANS turbulence models is discussed.

A. Summary of \textit{elsA} features available for structured grids

\textit{elsA} solves the inviscid (Euler), and viscous (laminar, Reynolds Averaged Navier-Stokes -RANS- and Large Eddy Simulation -LES-) compressible fluid equations in a steady or unsteady formulation on a structured multi-block mesh. \textit{elsA} includes multi-physics for aero-elasticity, shape optimization with an adjoint module. \textit{elsA} has been validated on a very broad range of configurations, including aircraft (civil and military), turbomachine, helicopter and missile.

From the numerical point of view, \textit{elsA} is a cell-centered finite volume solver: it solves the RANS equations in a conservative formulation over each finite volume. For convection, all the schemes are written in a fully structured way: to compute fluxes at cell interfaces, one uses the cells which share the interface and if needed, the second row of neighboring cells.

B. Unstructured convection scheme

A first order upwind Roe’s scheme\textsuperscript{24} only needs for each interface the left and right hand-side conservative fields and the normal vector to the interface. Therefore, the extension to unstructured grids is straightforward. The principle of the MUSCL approach\textsuperscript{25} for second-order accuracy is to replace the zeroth order extrapolation of the conservative variables at the mesh interfaces by a first order approximation that involves gradients of the variables. A slope limiter must be applied to guarantee the maximum principle.

A classical 2nd order MUSCL implementation for structured grids needs four cells for the extrapolation, and following a one-dimension formulation for an interface $i + 1/2$ located between cells $i$ and $i + 1$ in one dimension, it writes:

$$\begin{align*}
W^L_{i+1/2} &= W_i + 0.5\psi(W_i - W_{i-1}, W_{i+1} - W_i) \text{ for the left hand side} \\
W^R_{i+1/2} &= W_{i+1} - 0.5\psi(W_{i+1} - W_i, W_{i+2} - W_{i+1}) \text{ for the right hand side,}
\end{align*}$$

(1)

where $\psi$ is the slope limiter.
For unstructured grids, looking for a second row of cells is a complex geometry problem since it involves shape elements, elements surrounding cells and nodes. Therefore, it is more convenient to modify the MUSCL extrapolation procedure; the approach is based on a combination of centered and upwind differences, closely following the implementation for cell-vertex CFD codes:26–28

\[
\begin{align*}
W_{i+1/2}^L &= W_i + 0.5\psi(\nabla W_i \cdot \overrightarrow{CLCR}, W_{i+1} - W_i) \\
W_{i+1/2}^R &= W_{i+1} - 0.5\psi(\nabla W_{i+1} \cdot \overrightarrow{CLCR}, W_{i+1} - W_i)
\end{align*}
\]

(2)

Up to now, only van Albada, van Leer, minmod and Superbee slope limiters have been implemented. As a consequence, the cell barycenter is a new geometric quantity needed for unstructured grids for all computations while, on structured grids, it was only necessary for the computation of the distance to the wall in the case of turbulent RANS computations.

C. Diffusion scheme

1. Structured grid

In elsA, two kinds of diffusion schemes are implemented for structured grids: a pure centered scheme based on Green-Gauss theorem and defined as for a cell-vertex framework,29 and a scheme based on a dual control volume built around the considered mesh interface.

2. Unstructured grid

The pure centered approach has been implemented for unstructured grids; in order to avoid odd / even decoupling, a modified-centered approach has also been considered.30 Let Σ be a mesh interface located between the left L and right R volumes and let \( \nabla W_\Sigma \) be the centered gradient on Σ of a quantity W. This gradient is corrected in the direction linking the left and right cell centers as:

\[
\nabla W_\Sigma = \nabla W_\Sigma - \left( \nabla W_\Sigma \cdot \overrightarrow{CLCR} - (W_R - W_L) \right) \frac{\overrightarrow{CLCR}}{||\overrightarrow{CLCR}||^2},
\]

where \( C_L \) and \( C_R \) are the centers of left and right cells. A numerical analysis of the scheme has shown that it is second order accurate on squares, and that the accuracy decreases on regular triangles (approx. order 1.2).31

A discretization similar to the scheme based on a dual control volume for structured grids (section III.C.1) is planned for future implementation. It is based on a diamond-like scheme.32 This scheme leads to a constant accuracy on all shape elements. However, this approach is memory and CPU consuming since extra data is necessary at the interface mesh nodes. For a node i, the needed quantities are computed from a first order extrapolation coupled with a least-square formulation based on cells sharing node i.

D. Temporal integration

The temporal integration is either explicit - 4-stage Runge-Kutta scheme - or implicit - LUSSOR relaxation scheme.33 For unsteady computations, Gear and Dual Time Stepping (DTS) algorithms are available. The formulations for steady and unsteady flows are derived directly from the initial structured implementation, and almost no modifications have been required for unstructured zones, thus avoiding many code duplications.

E. Implementation of turbulence models for RANS simulations

Two classical RANS turbulence models, namely the Spalart-Allmaras and Wilcox \( k - \omega \) turbulence models, are available for computations on unstructured grids. The computation of the distance to the wall is based on the minimum distance between each cell and any wall face centers. The convective scheme is derived from the structured implementation, using decoupled approach and the Roe’s scheme. The diffusive term is computed exactly as for the mean flow quantities. Source and dissipation terms are volume contributions and they are therefore not computed within a face-based approach.
F. From algorithms to data structure for unstructured zones

The choice of the data storage format is driven by the necessity to secure conservation at mesh interfaces. For a cell-centered formulation, the best way to implement the unstructured approach is to store data within a face-based format. The principle is therefore to define the list of mesh interfaces and the corresponding cell references. The definition of left and right hand sides of the interface is done according to the definition of the normal vector direction: the normal vector is directed from left to right. The conversion of the element-based connectivity to the face-based connectivity is done outside by a preprocessing tool, and the element-based connectivity is neither read nor stored by elsA in order to minimize memory.

Even if face-based algorithms are trivial for convection or diffusion, the classical algorithms for metric have been modified to deal with face-based formulation. As an example, let consider the definition of the cell barycenter (necessary for the MUSCL approach). For control volumes with a single kind of faces (triangles or quadrangles), the face-based algorithm for the computation of the cell center is obvious. As an example, consider a tetrahedron $ABCD$ with a center $G$. Let $G_1$ (resp. $G_2$, $G_3$ and $G_4$) be the center of the face $ABC$ (resp. $ABD$, $BCD$ and $ACD$). Then:

$$G = \frac{A + B + C + D}{4} = \frac{G_1 + G_2 + G_3 + G_4}{4}. \quad (3)$$

The situation becomes worse for elements that blend triangle and quadrangle faces, since the weights associated with each vertex depend on the number of vertices of the face.

If one limits the element shapes to tetrahedra, prisms, pyramids and hexahedra, it is possible to derive a face-based formulation for the cell center computation. The idea is to associate a weight of unity for the triangle face center and a weight of 2 for a quadrangle face center. For sake of clarity, the formulation is presented only for pyramids. Let $ABCDE$ a pyramid with the quadrangle face $BCDE$. Let $G_1$, $G_2$, $G_3$, $G_4$, $G_5$ and $G$ be the barycenters of faces $ABC$, $ACD$, $ADE$, $ABE$, $BCDE$ and of the volume respectively. We have:

$$3G_1 = A + B + C, \quad (4)$$
$$3G_2 = A + C + D, \quad (5)$$
$$3G_3 = A + D + E, \quad (6)$$
$$3G_4 = A + B + E, \quad (7)$$
$$4G_5 = B + C + D + E, \quad (8)$$
$$5G = A + B + C + D + E. \quad (9)$$

From Eq. 4, Eq. 5, Eq. 6, Eq. 7, one can deduce:

$$3G_1 + 3G_2 + 3G_3 + 3G_4 = 4A + 2B + 2C + 2D + 2E,$$

and therefore, using Eq. 8 and Eq. 9, we have:

$$3G_1 + 3G_2 + 3G_3 + 3G_4 + 2 \times 4G_5 = 4(A + B + C + D + E) = 20G. \quad (10)$$

An important remark concerns the integer “20” in Eq. 10: it is the number of nodes counted for determining $G$ since $20 = 3 + 3 + 3 + 3 + 2 \times 4$. The same procedure applied to tetrahedra, prisms and hexahedra leads to final weights of 12, 30 and 48 respectively.

IV. Software design

A. Object-Oriented architecture

The hybrid structured-unstructured RANS solver is developed taking advantage of the elsA Object-Oriented (OO) framework. elsA kernel is based on the definition of (C++) classes which define the main objects used during the computation. For each essential computational class of elsA an abstract class has been designed, providing the common interface to structured and unstructured technology. Specialized methods
Figure 1. UML diagram of the hierarchy of Boundary class. BndBase is the root of the inheritance hierarchy; from BndBase are derived two abstract classes, for physical and topological boundary conditions. BndPhys is the generic class for structured (S) and unstructured (U) true (concrete) implementations of the boundary conditions.

are then developed twice for structured and for unstructured zones. Fig 1 illustrates the class hierarchy for a subset of the C++ classes implementing boundary conditions.

In practice, in the implementation of numerical algorithms, for example the time integration, the hybrid solver only manipulates polymorphic objects in order to avoid to take care explicitly of structured and unstructured zone treatments: due to inheritance, when a pure virtual method is called by a polymorphic object, the concrete method is used in the corresponding derived class. Therefore, this technique allows implementing unstructured features without duplicating nor modifying the existing structured code. This demonstrates the advantage of a fully-integrated approach versus coupling of separate codes (or libraries) by a master module, where code duplication would be unavoidable.

Somewhat surprisingly, it turns out that even non object-oriented code elements can be used by both solvers, provided that a small number of basic programming rules are respected. Let us give as an example the implementation of many Fortran computational loops:

```
DO n = n0, nf
   ...
END DO
```

In many cases, we have been able to keep a single (Fortran) implementation used by both solvers; the only difference is in the C++ calling code, where n0 and nf are set.

B. Input / Output strategy

The desire to handle simultaneously structured and unstructured grids at the same time has driven a study in order to analyze the mesh format outputs from mesh generation tools. It appeared that the CGNS Library is able to handle structured and unstructured zones, and provides sufficient flexibility for the implementation of data not “normalized” by the CGNS standards (which are called UserDefinedData). A CGNS database is in practice a tree, whose leaves have specific meanings. For unstructured grids, the mesh format is based on the definition of element connectivity, which cannot be used directly by our face-based unstructured solver. As for most of unstructured CFD codes, the preprocessing step is not done inside the CFD kernel and a CGNS database preparation tool called PREPCGNS adds the specific data to the CGNS database.

Note that numerical and physical parameters are not stored in the CGNS database; instead, it is much more convenient to define them using an advanced Python scripting interface, which provides many useful features to end-users, such as coherency checks and default mechanism.44,35

C. Numerical aspects of elsA and their consequences for unstructured grids

The cell-centered approach used in elsA consists in a flux balance at each mesh interface. This choice for structured grids had a lot of consequences on the treatment of unstructured zones. Many unstructured CFD...
codes use a node centered formalism and define a dual mesh whose facets are build on cell centers, face centers and edge midpoints. For such codes, a coupled finite volume / finite element formalism is generally used: the finite volume formalism for convection defined at dual mesh facets is combined with a finite element approach which provides the definition of gradient inside mesh elements through the use of finite element shape functions. On multi-element shapes, the computation of the gradient for diffusion is a key point, and a new approach has been proposed recently for non simplex elements.

For coherence with the initial kernel, it has been decided to give up all methods not based on a cell-centered approach. Consequently, the derivation of a gradient from finite element shape functions is not anymore allowed for the diffusion scheme. The definition of the diffusion scheme is theoretically known as one of the difficulty of the approach.

V. Mismatched abutting interface for hybrid mesh

The main idea behind the development of this framework is to keep structured zones where accuracy is needed and unstructured zones to alleviate meshing constraints in some regions for complex geometries. Therefore, the connection between structured and unstructured zones must be addressed in an efficient way.

Let us stress that both solvers use a cell-centered finite volume formulation – the cells of the primary grid itself are used as the control volumes.

A. Matched (1-to-1) abutting interface

The simplest interface between structured and unstructured zones is the 1-to-1 abutting interface to refer to the CGNS naming convention.

There are two ways to compute conservative fluxes on such interface. Either fluxes of one zone are sent to the other zone, which ensures automatically the flux conservation property, or fluxes are computed independently in the two zones and the conservation property results from the application of numerical schemes on the same data in both zones. The latter method is used so, if the metric is computed with the same algorithms on both sides of the multizone interface, then the surface vector is ensured to be the same for each face. Therefore, the numerical schemes applied at the multizone interface must be adapted since their formulation may not be equivalent in structured and unstructured zones.

B. Mismatched abutting interface

The mismatched abutting technique has been chosen to patch structured and unstructured grids in elsA for two main reasons. An example of a mismatched multizone interface is shown on Fig. 2. First, it has proven very effective in the structured context in terms of meshing process flexibility. Moreover, the technique of 1-to-1 abutting interface is probably not able to answer all needs. Indeed, structured meshes are generally designed to be aligned with the flow direction and the cell anisotropy ratio can be very high, typically up to $10^5$ in a turbulent boundary layer. On top of that, an unstructured CFD code can generally not handle such anisotropic elements efficiently except with appropriate expensive numerical schemes. As a consequence, a mismatched abutting formalism seems compulsory for RANS applications depending on the location of the multizone interfaces. Of course, the matching interface strategy between structured and unstructured zones remains the preferred option in isotropic region and especially for Large Eddy Simulation. Secondly, special flux treatment at this interface can ensure the finite volume property of flux conservation.

The numerical implementation relies both on ghost cells (also called halo cells or dummy cells) and on flux interpolation. Fluxes on block interface are computed by an adaption of the numerical schemes used for interior faces. Figure 3 shows a sketch of a mismatched abutting interface that is used to explain the numerical implementation. The main principle consists in cutting the right face of the cell $(i_1 = 1, j_1)$ which abuts the right block at the interface $i_1 = 1/2$ (solid bold line) into many faces on which fluxes will be computed. The sum of all the fluxes of these so-called intersection faces will provide the flux on the whole face. On Fig. 3 the ghost cell at $i_1 = 0$ (dash-dotted line) adjacent to the cell $(i_1 = 1, j_1)$ is composed by the vertices $A$, $B$, $B'$ and $A'$. The surface (segment here) $AB$ is divided in two parts $AM$ and $MB$. For the left domain the numerical flux on $AB$ can be written:

$$f_{1/2,j_1} = f_{AM} + f_{MB} = \alpha f(W_{i_1,j_1}, W_{i_2,j_2}) + (1 - \alpha) f(W_{i_1,j_1}, W_{i_2,j_2+1})$$  (11)
Figure 2. Sketch of a mismatched abutting interface (solid blue line in the center) between structured blocks (solid black lines) and unstructured zones (red, green and black dashed lines).

Figure 3. Mismatched abutting interface between two 2D zones (solid line on the left with numbering (i1,j1), dashed line on the right with numbering (i2,j2)). Cell (B,B',A',A) or (i1=0,j1) represents a ghost cell belonging to left grid.
with $\alpha = \frac{|AM|}{|AB|}$ and $W$ the conservative variables.

Then, the intersection face between two quadrilaterals (or triangles) has to be computed. The intersection weighting coefficient $\left(\frac{|AM|}{|AB|}\right)$ is deduced from the polygon clipping algorithm of Sutherland-Hodgman that provides the intersection polygon of the two faces (A,B) and (M,N). This algorithm uses a divide-and-conquer strategy to cut successively the subject polygon (A,B) with the edges of the clipping polygon (M,N).

This numerical treatment has to be implemented for each numerical spatial scheme. Two schemes have been considered up to now: the first-order Roe scheme and a second-order Roe scheme.

The first-order Roe scheme only requires the left and right cell values for each intersection face and the normal vector to this face. It is then straightforward to apply Eq. 11.

The extension to second-order space accuracy is made through a MUSCL approach and piecewise constant approximations of primitive variables are replaced by slope-limited piecewise linear approximations for each cell (see Section III.B). For structured blocks, the MUSCL approach takes into account mesh directions and the extrapolated states for the interface $(i+1/2)$ are given by Eqs. 1 with the slope limiter $\psi$ that ensures the TVD principle. The stencil contains therefore four cells for each extrapolation and each mesh direction. At the boundary between two structured blocks, two rows of ghost cells are used to keep the same mathematical formulation at the block interface and thus the flux conservation.

For unstructured grids, the left and right extrapolated states are given by Eqs. 2. Therefore, only one row of ghost cells is needed at the boundary between two unstructured zones. As for structured grids, the implementation of schemes does not account for cell stretching and in particular, it is assumed that the interface is located at the middle of $C_L C_R$.

For hybrid mesh, the numerical treatment at the interface between two unstructured grids utilizes Eqs. 2. For a boundary between two structured grids, Eqs. 1 are used. The choice was made to apply the same numerical schemes at the mismatched abutting interface working on the same data in both zones. Thus, Eqs. 2 are applied to compute fluxes at this multizone interface. To compute fluxes at interfaces next to the border interfaces on the structured side, the first row of ghost cell is filled with an interpolated value (see Fig. 3):

$$W_{i1=0,j1} = \frac{|AM|}{|AB|}W_{i2,j2} + \frac{|BM|}{|AB|}W_{i2,j2+1}$$

An example of computation over a hybrid grid for a 2D NACA0012 airfoil is given in Fig. 4. The mesh is composed of hexahedra for structured blocks and prisms for the surrounding unstructured zone. The abutting mesh interface can be clearly seen in the wake region. The transonic computation shows some limited discontinuity at the abutting interface for the density on Fig. 5. This is justified by the large discrepancy between the cell dimensions around involved abutting interfaces.

### VI. Solver efficiency and parallel scalability

A large-scale effort is in progress to obtain a highly efficient solver, tuned to the current generation of multicore massively parallel computing platforms.

#### A. Single core CPU optimization

Well-designed CFD codes are known to be very demanding in terms of memory bandwidth between the computational cores and the memory system. This is specially a problem for modern processors, such as Intel Wesmere or AMD Magny-Cours, where the increase in the number of cores is not associated with an equivalent increase in memory bandwidth. It was found that to use efficiently the cache system of these processors, a significant number of existing elsA routines had to be modified. The key ideas are:

1. improve temporal locality: when data is fetched from memory into the processor registers, use it "as much as possible", by performing several computational tasks in the same computational loop. This coding style is somewhat contradictory with the requirement of modularity ("one task, one dedicated method"), but this is the price to be paid.

2. improve spatial locality: organize data structure so that computations involve data located in a "small" region of memory, thus minimizing potential failure (cache misses...) of the memory system to feed the computing kernel.
Figure 4. Example of a hybrid grid composed of 6 structured blocks (red and black) and 1 unstructured zone (blue). The computation is done on 3 CPUs: one for the unstructured zone, one for the red blocks and one for the black blocks.

Figure 5. Density contours obtained for the computation on NACA0012 profile. The discontinuity of density contours across the abutting interface is due to strong and inappropriate cell size variations in the mesh.
Typically the single core acceleration for the unstructured solver can reach 50 to 100%, depending on applications.

**B. Unstructured parallel computations**

For parallel computations, one has to provide a buffer for data exchange between blocks (and processors). The buffer for structured blocks is based on the definition of two rows of ghost cells on each matching block boundaries. This approach enables the use of the same schemes inside the block and at matching block interfaces. For unstructured grids, the number of rows has been fixed to 1. This choice is motivated by:

- One row of ghost cell enables a closed face-based data structure: every face has two volumes and one can be (or not) a ghost cell.
- Defining properly a second row of ghost cells is not achieved easily for unstructured grids. In practice, which cells must be considered at a 1-to-1 block boundary? Those sharing a face? Plus those sharing a node with a boundary face? And what about volumes that share a mesh vertex? Finally, it seems to be dependent of the solver stencil, which is not required for a buffer.

For unstructured computations, a splitter tool for multi-element unstructured grids, based on the METIS library, has been developed.

**C. Parallel scalability**

The number of computing cores available in industry or research computational centers has increased markedly in the last few years, and will continue to grow in the foreseeable future. This means that the challenge of maintaining a good efficiency with an increasing number of computational cores has to be addressed, which is not a trivial task. Hybrid mesh parallel computations have two specific features:

- Load balancing: Contrary to structured meshes, where topological constraints complicate the task of block splitting, unstructured meshes are relatively easy to partition and excellent tools are available, such as METIS or SCOTCH. However, partition of a truly hybrid mesh, in which structured and unstructured zones coexist, is a very difficult task, which is apparently not addressed in the literature.
- The numerical algorithms used to couple the zones must be carefully tuned in order to maintain good efficiency, particularly if strong scalability is required.

Figure 6 shows the measured parallel strong\(^1\) scalability for the ”medium” configuration (unstructured mesh) from the 1st AIAA CFD High Lift Prediction Workshop.\(^5\) The mesh is composed of 21.75 million tetrahedra.

Figure 7 shows the parallel scalability of the structured solver measured on a INTEL-Nehalem based platform, up to 2048 cores, with a mesh size of 1.7 \(10^9\) nodes, for the 4th AIAA Drag Prediction Workshop configuration.\(^4\)

**VII. Validation examples**

The hybrid structured-unstructured RANS solver has been validated by a variety of test cases. In the following four validation examples are presented for steady external flows.

**A. RAE2822 2D profile – hexahedra grid**

Fig. 8 shows the density distribution computed for a RAE2822 profile; inflow data are chosen according to Hellström.\(^43\) The mesh is entirely composed of hexahedra. The red lines show the solution on a structured grid and the black lines show the solution on the same grid treated as unstructured. The agreement between structured and structured solutions is satisfactory.

\(^{1}\)strong scalability measures the (wall clock) computing time reduction when the number of processors is increased,\(^{40–42}\) keeping the same configuration
Figure 6. Strong scalability: unstructured configuration (HiLiftPW-1)

Figure 7. Strong scalability: structured configuration (DPW4)
Figure 8. Comparison of the density computed on a structured grid (red lines) and on the same grid treated as an unstructured one (black lines) for the RAE2822, case 9, Spalart-Allmaras.

B. High Lift inviscid unstructured computation

Fig. 9 gives an example of Euler unstructured computation for a high-lift configuration representative of current civil aircrafts. The mesh is composed of 40 million tetrahedra and the computation has been performed on 64 computing cores. However, the convergence rate is not optimum due to the lack of a multigrid algorithm\(^k\).

Figure 9. Example of computation with elsA for a high lift configuration representative of current civil aircrafts. The mesh is composed of 40 million tetrahedra. The wall skin pressure and a partial view of the surface triangular mesh are given. The white filets represent the streamlines around the wing.

C. High Lift turbulent unstructured computation (HiliftPW1)

Two different meshes available from the High Lift Prediction Workshop\(^5\) have been used to validate our unstructured solver. Spalart-Allmaras turbulence model was used in both cases.

\(^k\)The multigrid algorithm is under development by ONERA
1. **Coarse mesh**

Fig. 10 presents the wall pressure computed on a relatively coarse mesh:

- 32 computational cores;
- unstructured hexahedra, built from 1-to-1 structured grid;
- configuration 1, serie A, coarse grid;
- Slat 30 Flap 25;
- number of grid points: 20,356,741
- number of grid cells: 20,107,008 (half-configuration);
- $Mach = 0.2, Re = 4.3 \times 10^6, Temp = 520R$

![Figure 10. Trap Wing configuration – wall pressure.](image)

2. **Medium mesh**

Fig. 11 illustrates the medium mesh, with 21,743,354 tetrahedra only. The computations have been performed with 480 Intel Wesmere cores. Computed wall density and wall pressure are given in Fig. 12 and 13.

**D. Drag Prediction Workshop DPW4**

We are currently validating our code using several meshes available from the DPW4 web site. Fig. 14 shows a zoom on one of the CENTAUR meshes provided by DLR, with $35 \times 10^6$ computational cells. The mesh has been splitted in 480 unstructured zones. Fig. 15 shows the computed wall density (computation is not fully converged).

**VIII. Conclusion**

The hybrid mesh paradigm described in this paper combines the flexibility and convenience of unstructured grids with the efficiency and accuracy of structured grids to make optimal use of both. The unstructured grid solver has been developed with consistent numerical algorithms, data structure, user interface and parallelization to those of the structured one.
Figure 11. Trap Wing configuration – Medium mesh

Figure 12. Trap Wing configuration – Medium mesh – wall density
Figure 13. Trap Wing configuration – Medium mesh – wall pressure

Figure 14. DPW4 CENTAUR medium mesh
Many coding elements are shared between the two solvers, insuring consistency and avoiding duplication, thus leading hopefully to reduce software maintenance and validation costs.

The numerical algorithm used at the interface between structured and unstructured zones is a key point of our approach, and the advantages associated with abutting interface treatment have been demonstrated.

The fully integrated software architecture should facilitate maintenance tasks, as well as deployment on massively parallel computing platforms.

Promising results have already been obtained.

Several improvements are currently in development, such as a multigrid smoother to improve the robustness and convergence of the unstructured solver, a matrix-free GMRES implicit stage, a new scheme for diffusive fluxes with better accuracy, optimized data memory layout for fast memory access, integration into Airbus FlowSimulator\textsuperscript{44} and a load balancing tool tuned to parallel hybrid mesh computations to achieve good scalability.

References

3. CFD General Notation System,” \textit{http://www.cgns.org}.

\textsuperscript{1}a Python-controlled computing environment to unify CFD simulation workflows, see \textit{http://www.euroscipy.org/talk/896}
