FROM GEOMETRY TO CFD-BASED AERODYNAMIC DERIVATIVES - AN AUTOMATED APPROACH

Maximilian Tomac , David Eller
Aeronautical and Vehicle Engineering
Royal Institute of Technology
SE - 100 44 Stockholm, Sweden

Keywords: *Mesh Generation, Automated Analysis, CFD*

1 Introduction

The aim of the EU-funded collaborative research project SimSAC is to generate stability and control data for preliminary aircraft design using methods of varying fidelity. This paper is concerned with the automatic determination of aerodynamic data, namely the computation of aerodynamic derivatives for the rigid aircraft with control surfaces. In order to obtain this data, the aircraft geometry must be defined, computational meshes for different aerodynamic solvers need to be created and finally, the flow solver parameter settings must be adapted to reliably perform multiple solutions from which the derivatives are computed. To illustrate the process, the modelling, mesh generation and automated computation setup procedures are applied to the EADS Ranger 2000 jet trainer.

Most of the geometry modelling process could be performed inside one of the industry standard general purpose CAD systems, and the mesh generation left to a commercial mesh generator. This approach, although theoretically highly attractive, has significant disadvantages. Due to the complex nature of both CAD systems and high-fidelity flow solvers, combined with the associated substantial licensing and training cost, the number of aerospace CFD experts who also are experienced users of both CAD and mesh generation software is rather limited. Furthermore, the geometry description exported by standard CAD systems to the mesh generation software tends to be unnecessarily complex and usually requires extensive manual simplification efforts, thereby severely encumbering automation.

2 Variable-fidelity framework CEASIOM

Figure 1 shows an overview of the CEASIOM framework. The framework integrates discipline-specific tools with main focus on aircraft conceptual design. As mentioned above, the focus of this paper is on reducing the engineer's workload by means of automatic determination of aerodynamic data. This brings us to `sumo` and the interaction between the *CFD* and *AMB* modules in figure 1. Methods spanning from simple empirical procedures and vortex-lattice (VLM) methods to inviscid Euler and viscous RANS simulations are used within the *CFD* module. For further details see references [1, 2].

3 Geometry

The simplest geometry representation is a collection of global shape parameters which can be used to describe a set of typical aircraft configurations with a level of accuracy suitable for use with VLM aerodynamics. The particular set of shape parameters adopted for the SimSAC project is here referred to as the CEASIOM [2] geometry description. An example for a typical CEASIOM geometry dataset is shown in Figure 2. The VLM model only includes lifting surfaces such as wing and vertical tail and stabilizer, while

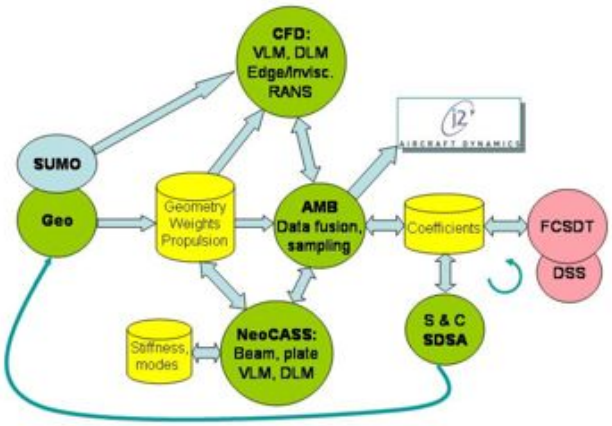


Fig. 1 Illustration of CEASIOM framework

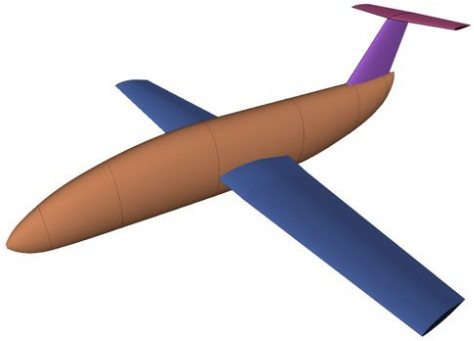


Fig. 2 CEASIOM geometry of baseline model

the CEASIOM geometry representation also includes airfoil thickness and non-lifting surfaces such as the fuselage. At the next higher level of detail, the graphical surface modelling tool `sumo` can be used to define a more detailed geometry based on a moderate number (often less than 30) spline surfaces. This description is used to generate input for CFD solutions based on the Euler equations. At a later stage when details of interest have been added in `sumo` , the geometry representation can be exported to CAD systems or commercial mesh generation software by means of IGES [3] files.

4 Mesh generation

From the CEASIOM geometry parameters, the VLM mesh can be generated directly within the Matlab-based CEASIOM environment. For higher-fidelity geometry descriptions and the generation of meshes for CFD solutions based on the Euler equations, `sumo` and TetGen [4] are employed, as described in the following sections.

Generation of volume meshes for Navier-Stokes computations can exploit the comparatively simple geometry representation of available in `sumo` . Continuous geometry data can be exported in IGES format, while volume meshes are exported using CGNS [5]. The commercial mesh generation software ICEM-CFD [6] is then controlled by small scripts in order to automatically generate a hybrid prismatic-tetrahedral mesh suitable for Reynolds-averaged Navier-Stokes (RANS) simulations, as detailed in Section 4.3. Figure 3 illustrates the typical level of detail which can be achieved with this approach.

4.1 Surface mesh generation

The accuracy of the numerical solution of the flow problem is obviously rather strongly affected by the quality of the surface discretization. As the solver used in this study exploits only piecewise linear geometry and represents the solution as nodal values, the surface mesh must

- approximate the underlying geometric surface with sufficient accuray;
- allow resolution of important flow features such as pressure distributions.

While the first requirement is, in most cases, a necessary condition for the second, it is not always sufficient. As an example, consider the pressure recovery near the trailing edge of a lifting surface. Although the surface geometry usually is relatively flat in this region, fairly small triangles are needed in order to resolve the steep pressure gradient accurately. For a single flight condition, this could be achieved by means of solution-adaptive mesh refinement, but for the multitude of cases considered here, that is not necessarily a practical option.

The algorithm employed in `sumo` to automatically generate surface meshes ensures that the first requirement above is fulfilled to given

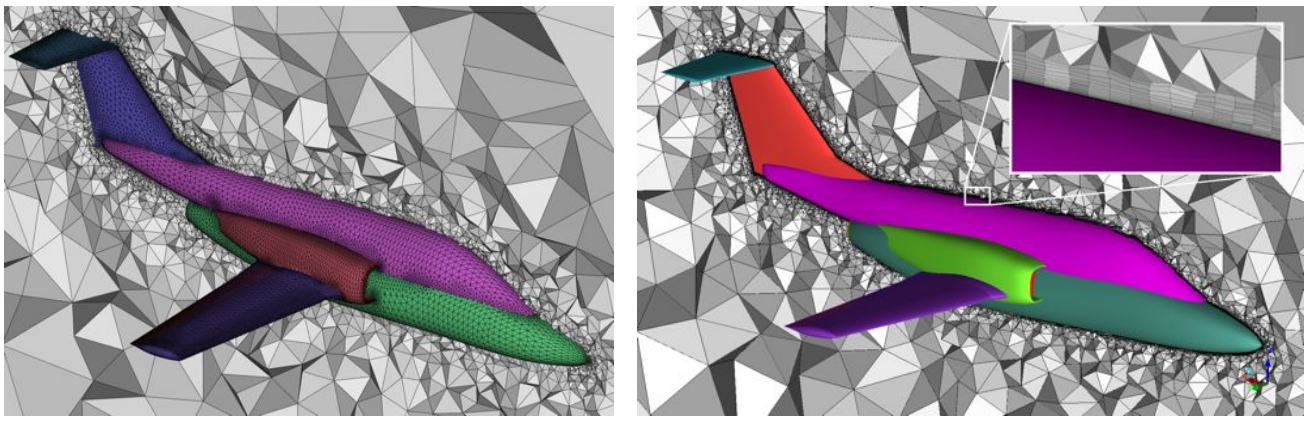


Fig. 3 Left: Euler mesh; Right: RANS mesh, both based on `sumo` geometry

tolerances. A set of geometric heuristics described below are used to refine particular regions where large variations in pressure are typically observed.

All geometric surfaces which make up the aircraft configuration are bi-parametric surfaces of the form $(x,y,z) = S(u,v)$, where at least the first derivatives with respect to the parameters u and v must be continuous. The mesh generation is performed in the parameter space (u,v) by means of a modified Delaunay algorithm similar to that presented by Chew [7]. A *plane* triangular mesh is said to be Delaunay if the *circumcircle* of any plane triangle does not contain any vertex of another triangle. Chew extended this definition to triangles on a parametric surface, where the mesh is defined to be Delaunay if the *circumsphere* of any three-dimensional triangle does not enclose vertices of any other triangle. A mesh which does not have this property can, in principle, be made conforming by a sequence of edge flips. Here, however, lies an important difference to the two-dimensional case, since edge flips in three dimensions can lead to geometric irregularities which do not exist in 2D.

As an example, consider a long triangle edge lying on the leading edge of a thin wing surface. If this edge needs to be flipped in order to re-establish the Delaunay property, it will connect a point on the upper surface of the wing surface with a point on the lower surface, which most likely leads to very poor mesh quality. Even Chew states that the process of improving mesh quality (constituting the dominating part of the whole mesh generation) must start from a mesh which is Delaunay in the three-dimensional sense. For general parametric surfaces, this initialization is not trivial.

4.1.1 Mesh initialization

To initialize the process with an (at least approximately) Delaunay surface mesh, a structured quadrilateral mesh is first created on each surface. Since the parameter domain $(u,v) \in [0,1]^2$ is quadratic, this step is comparatively simple. In the common case of surfaces with very large changes in local curvature, such as wings, highly stretched quadrilateral elements result. Splitting these quadrilaterals into triangles would obviously not yield a surface mesh conforming to the Delaunay property. Therefore, another pass is applied in which the initial, highly stretched quadrilateral mesh is converted into a geometrically adapted triangular mesh. Figure 4 shows the hybrid mesh just after conversion to triangular elements and the final mesh which fulfills the Delaunay property. In some cases, the mesh quality requirements of the flow solver may allow to use surface discretizations of the type depicted to the left in Figure 4. Usually, however, considerably better quality must be achieved.

4.1.2 Refinement

Once initialized, the triangular mesh is refined until a set of criteria are fulfilled. These crite-

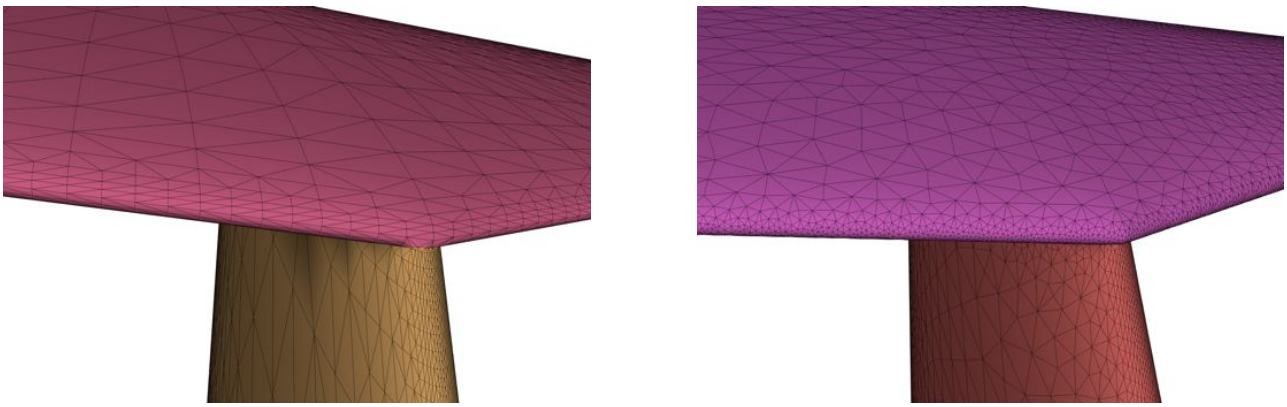


Fig. 4 Left: Initial triangular mesh. Right: Delaunay mesh.

ria can be set by the user, but the mesh generator attempts to determine sensible default values which in most cases yield a usable discretization. The following mesh generation parameters can be used to control the mesh generation process:

Dihedral angle is the angle between the normal vectors of two triangles sharing an edge. This parameter mainly affects the resolution of strongly curved regions such as wing leading edges.

Edge length can be controlled by enforcing both a minimum and maximum length. The minimum edge length serves to avoid resolution of irrelevant geometric details.

The triangle stretch ratio is defined as the length of longest divided by shortest edge. Here, the magnitude of the maximum acceptable ratio depends entirely on the properties of the flow solver used. A high stretch ratio allows to discretize large surfaces with small regions with strong curvature, such as thin wings, with fewer elements.

Leading- and trailing edge refinement factors are used to control resolution of regions which typically feature large pressure gradients. Since the geometry modelling component has information on which surfaces are wings, this abstract information can be exploited to enforce a gradual decrease of the maximum edge length from the value specified using the edge length parameter to a smaller value along the leading/trailing edge line. Figure 5 displays the effect on the root leading edge region of the EADS Ranger test case described below.

Mesh refinement is performed in an iterative process. In each pass, all triangles which violate one of the quality criteria are collected as candidates for refinement. The element refinement itself is performed by splitting the longest edge of the candidate triangle. Since most edges are shared by two triangles, this split operation normally affects both of them. Therefore, the quality criteria are evaluated anew each time an affected triangle is processed in order to determine whether a split is still necessary. Following each edge split, the Delaunay property is reestablished by a series of edge flips. This part of the algorithm is entirely identical to the two-dimensional Delaunay procedure with the only exception that no edge flips are performed which would lead to geometric degeneracies. Here, an edge is deemed degenerate if the angle between the surface normals at its vertices exceeds 90 degree. Once all candidate triangles are processed, the quality criteria is evaluated again for all triangles, resulting in a new set of triangles which violate the imposed criteria. This iterative refinement procedure is repeated as long as non-conforming elements are found, which usually requires between four and eight passes.

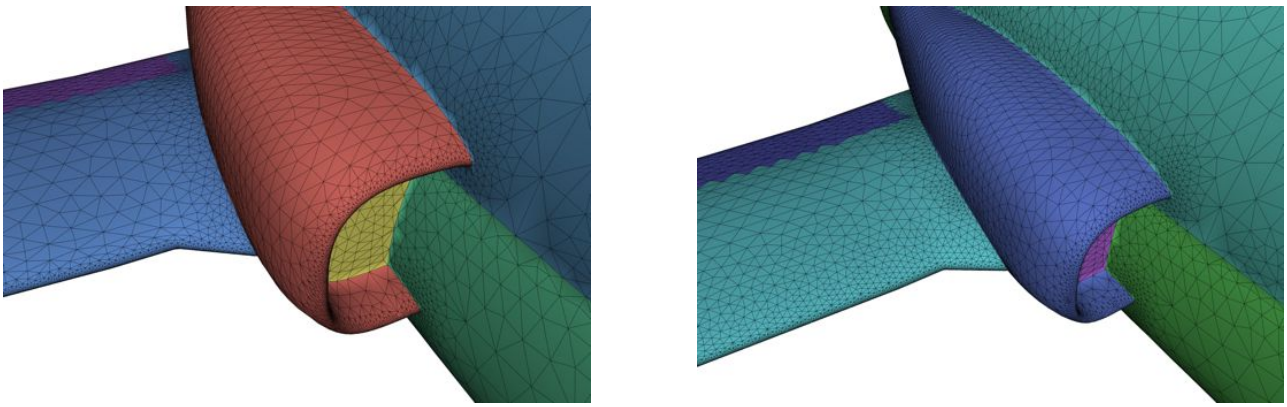


Fig. 5 Left: Leading edge refinement factor 2.0; Right: Factor 4.0

4.1.3 Surface intersections

The surface geometry module in sumo does not compute surface intersections based on the continuous geometry, as that would require additional user interaction in order to determine which part of a surface sliced by an intersection curve is to be kept and which discarded. Eliminating this interaction is a necessary condition for automatic mesh generation. Hence, surface intersections are computed based on the discrete geometry.

First, a triangular mesh is created on each surface using the procedure described above. Then, a bounding volume hierarchy (BVH) is established in order to allow for the efficient computation of element intersections. In this case, an unbalanced binary tree with axis-aligned bounding boxes is used, as such a BVH can be constructed at low computational cost. The triangle meshes generated by the algorithm above are characterized by a very high degree of spatial coherence, so that such a simple BVH is already very effective in eliding the vast majority of element intersection tests. Finally, the intersections of individual triangles are determined using the triangle-triangle intersection test by Möller [8].

As each mesh element carries both three-dimensional (x, y, z) geometry as well as parameter values (u, v), the intersection line can be constructed both in 3D space and in the parameter space of the intersecting surfaces. A limitation of the current implementation is that no more than three surfaces may intersect in a single point. Usually, discrete intersection lines determined in this way are very irregular in the sense that they often contain extremely small segments where one triangle intersected another at a small distance from one of its vertices. A postprocessing stage removes such excessively short segments where possible.

Intersection lines are then imposed as constrained edges for a subsequent mesh refinement pass. In order to recover these lines in the final mesh, such edges cannot be flipped or split by the mesh refinement process. For this reason, mesh quality criteria may not be fulfilled exactly near surface intersection lines.

Once every individual surface is meshed and all intersection constraints satisfied, the discrete meshes are merged and duplicate vertices (along intersections) eliminated. From the geometry modelling topology, a small set of elements on the external (wetted) surface of the configuration is identified. Starting from these triangles, the merged mesh is processed by a topological walk over element edges. For each edge which is connected to exactly two triangles and touched by a triangle identified as external, the opposed triangle (across the edge) is added to the set of external elements. Edges on intersection lines are connected to four triangles and always reached by the walk over an external element. From the three remaining triangles, the one whose plane includes the smallest angle with the plane of the external triangle is selected as the next external element.

This procedure is reliable as long as the connectivity of the discrete mesh near intersection lines is identical to the (mathematical, not necessarily numerical) connectivity of the continuous surface representation. In order to resolve typical degenerate cases which can occur whenever that is not the case, information about the originating surface of each element is exploited to assist in the decision.

4.2 Euler mesh generation

Once a high quality surface mesh is created, satisfying the requirements mentioned earlier, an unstructured tetrahedral volume mesh for use in the solution of the Euler equations is generated using TetGen [4], developed by Hang Si at the Weierstrass Institute in Berlin. TetGen is a very efficient quality-constrained tetrahedral Delaunay mesh generator. Starting from an initial constrained Delaunay tetrahedralization of the domain, nodes are dynamically inserted until a given set of quality criteria is met. The domain is delimited by a surface mesh of the aircraft configuration and the farfield boundary, created by `sumo` . Tetrahedral quality criteria available in TetGen, version 1.4.3, include

- maximum element volume;
- maximum ratio of circumsphere radius to tetrahedron edge length;
- minimum dihedral angle between faces.

Furthermore, a field of maximum permitted element volumes can be defined once a volume mesh exists. This feature is not yet exploited by the procedure implemented in `sumo` .

In order to comply with imposed element quality requirements, TetGen will subdivide even boundary triangles (if explicitly allowed to do so). Since no higher-order geometry description is available at this stage, this subdivision is performed in the triangle plane.

For reasonably complex configuration such as the Ranger example displayed in Figure 3, surface mesh generation requires about 40 seconds on a quad-core 2.8 GHz Intel Core i7 series processor. TetGen, which is a purely serial program, generates a tetrahedral mesh with about one million cells in less than one minute on the same computer.

4.3 RANS mesh generation

Towards the end of the conceptual design stage it might become of interest to perform higher fidelity analyses, e.g. based on RANS simulations. At this stage it may be presumed that considerable changes to the external geometry, especially the global topology, are less likely. For RANS solutions, the mesh must not only resolve large solution gradients created by shocks or trailing vortices, but should likewise be sufficiently fine in viscous boundary layers. This is not always easily accomplished and can often become a rather time consuming process.

A program written in the python scripting language has therefore been implemented to automate this process to some degree by means of templates for the ICEM-CFD mesh generation package. At the time of writing, this approach allows the handling of a wide class of topologically roughly similar configurations with limited user intervention, thereby drastically reducing the amount of engineering man-hours needed for mesh generation.

The required user intervention is confined to the establishment of a file with parameters (described in Section 4.3.1) controlling the prismatic mesh generation procedure in ICEM-CFD. Based upon these settings, the python program will then assign suitable parameters to different surface entities in the mesh and call the mesh generation program, using the tetrahedral mesh generated by `sumo` and TetGen as input. After typical mesh generation execution times of about one hour per 3 million mesh nodes (on a quad-core Intel Xeon 5520), the quality of the resulting hybrid mesh needs to be reviewed in order to identify outright failures of the template-based approach. Currently, this step has not been automated. Should the mesh quality be substantially acceptable, further global mesh smoothing operations can be ap-

plied to improve overall quality.

Compared to a traditional manual bottom-up mesh generation procedure, this approach saves a lot of man-hours, but requires a surface mesh of sufficiently high quality, as the surface discretization is not changed during the automated hybrid mesh generation procedure. Even the tetrahedral volume mesh used as input needs to be of acceptable resolution, but this requirements is not too difficult to achieve. Note that the quality of the tetrahedral mesh domain of the final hybrid mesh will be improved by the global smoothing passes in ICEM-CFD.

4.3.1 Prism parameter settings

The build-up of the prismatic boundary-layer mesh is controlled by a set of parameters which are shortly described in the following list. Most of these parameters can be selected once on a typically sized configuration and will then yield acceptable mesh quality for roughly similar geometries.

Initial height is the height of the first prismatic cell. Depending on turbulence model and boundary conditions used, the laminar sub-layer of turbulent boundary layers must be resolved. Typically, this requirement is expressed in terms of the dimensionless wall-normal coordinate y^+, defined with the wall shear stress τ_w, local fluid density ρ and kinematic viscosity ν according to

$$y^+ = \frac{\tau_w}{\rho}\frac{y}{\nu}.$$

As the laminar sub-layer extends to approximately $y^+ \approx 4$, (see e.g. Cebeci [9]), a common criterion is to resolve this part of the boundary layer with at least four prismatic cells, so that the first cell height should not exceed $y^+ \approx 1$. The python program can estimate the resulting absolute cell height from a user-supplied reference length and Reynolds number.

Prism height ratio is the ratio of the height of the next prismatic layer over the previous one.

Number of prismatic layers affects the total height and resolution of the boundary layer, it also affects the size of the step in cell volume between last (largest) prismatic and first tetrahedral cell.

Fillet ratio is a parameter which controls the permitted height-to-width ratio of prismatic cells. A larger ratio allows the prismatic layers to progressively smooth out sharp ridges and corners of the surface mesh, which tends to improve the quality of the transition to the tetrahedral mesh.

Orthogonal weight determines the emphasis laid on orthogonality of wall-normal cell edges with respect to the original surface. A weight of one essentially constrains these edges to perfect orthogonality, while a lower value such as 0.1 substantially improves robustness.

Auto reduce allows, when switched on, that colliding prismatic layers are reduced in height in order to avoid interference. Figure 6 shows an example where the layers between the fuselage and nacelle of a rear-mounted engine are squeezed to avoid collision. Furthermore, the image exemplifies the effect of a low orthogonal weight and high fillet ratio, leading to a rounded interface surface to the tetrahedral mesh domain.

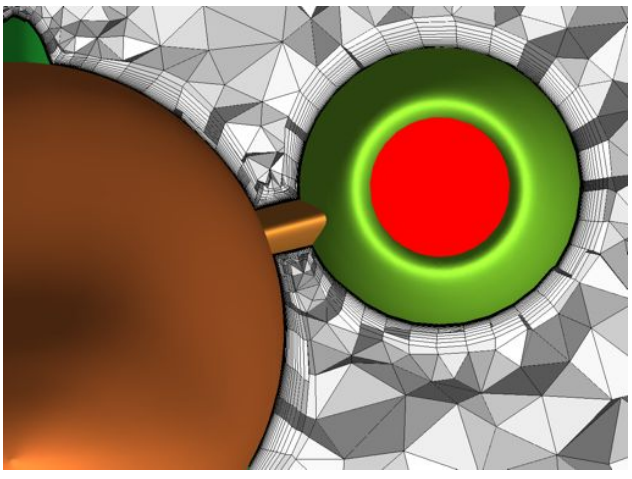


Fig. 6 Slice of a fuselage-nacelle mesh with auto-reduce on, low orthogonal weight, high fillet ratio

Smoothing steps during extrusion controls the number of iterations performed on each prismatic layer before proceeding to the next layer. Smoothing improves mesh quality considerably, but it is also fairly expensive in terms of computational effort.

Allowed prismatic angle is the maximum allowed angle within a prismatic cell. Since the surface mesh generated by `sumo` features sharp wing trailing edges, this value must be set to at least 160°, so that the prismatic cell layer wraps around the trailing edge.

Minimum allowed prismatic quality is a limiting value which controls at which point low-quality prismatic cells are replaced by pyramidal cells which then transition to the tetrahedral domain.

4.3.2 Robustness

This approach has been tested on a number of different aircraft configurations. For demonstration purposes the Ranger 2000 was fitted with a V-Tail instead of the actual T-Tail configuration Fig. 7 (top-left). The program was also applied to a business jet configuration (top-right) as well as a flying wing (bottom-left) and a wind tunnel model of the transonic cruiser (bottom-right), a common test case in the SimSAC project. In all of these cases, the resulting mesh quality is at least acceptable, but certainly not optimal. It should be noted, however, that the flow solver Edge features solution-based mesh adaptation [10], so that an acceptable initial mesh may very well be sufficient to obtain a first solution which is then used to refine specific regions of the mesh.

Since the mesh quality parameters described in Section 4.3.1 are applied globally, local geometric features such as narrow passages between bodies or surface edges with small included angle may force the use of overly conservative (robust) settings. This will then lead to reduced global mesh quality, resulting from excessively large volume ratio of prismatic to tetrahedral neighbour cells or poorly shaped (stretched) tetrahedra filling narrow convex corners.

A particular difficulty is introduced by the sharp wing trailing edges present in the geometry representation used by `sumo` . Low mesh quality in this region is particularly detrimental to solution accuracy. With the currently employed approach, significant enhancements in this area are unlikely, so that changes to the `sumo` geometry representation are probably necessary.

Figure 8 shows the result of applying the script-based approach to a very complex geometry, namely the X-31 experimental aircraft. The hybrid mesh around the canard trailing edge is generated successfully, although the volume ratio between coincident cells is rather large in this area. However, near the main wing trailing edge, the mesh generation algorithm fails to generate prismatic layers, so that pyramidal cells are created there instead. Unfortunately, this does not only impact mesh quality, but results in a substantial number of cells with negative volume, so that a flow solution is not even possible.

5 Automated CFD Solution

Like many other industry-strength CFD systems, the Euler/Navier-Stokes solver Edge [11] used in the SimSAC project requires a considerable number of parameters to be defined for each solution case. Depending on the flight condition of interest, a python program automatically

1. chooses appropriate boundary conditions and solver parameters;
2. generates solver input files and calls the preprocessor;
3. handles communication and data transfer to/from a managed-queue compute cluster;
4. optionally runs mesh adaption schemes;
5. outputs basic quality diagnostics on the solution;
6. alerts the user to insufficiently converged results;

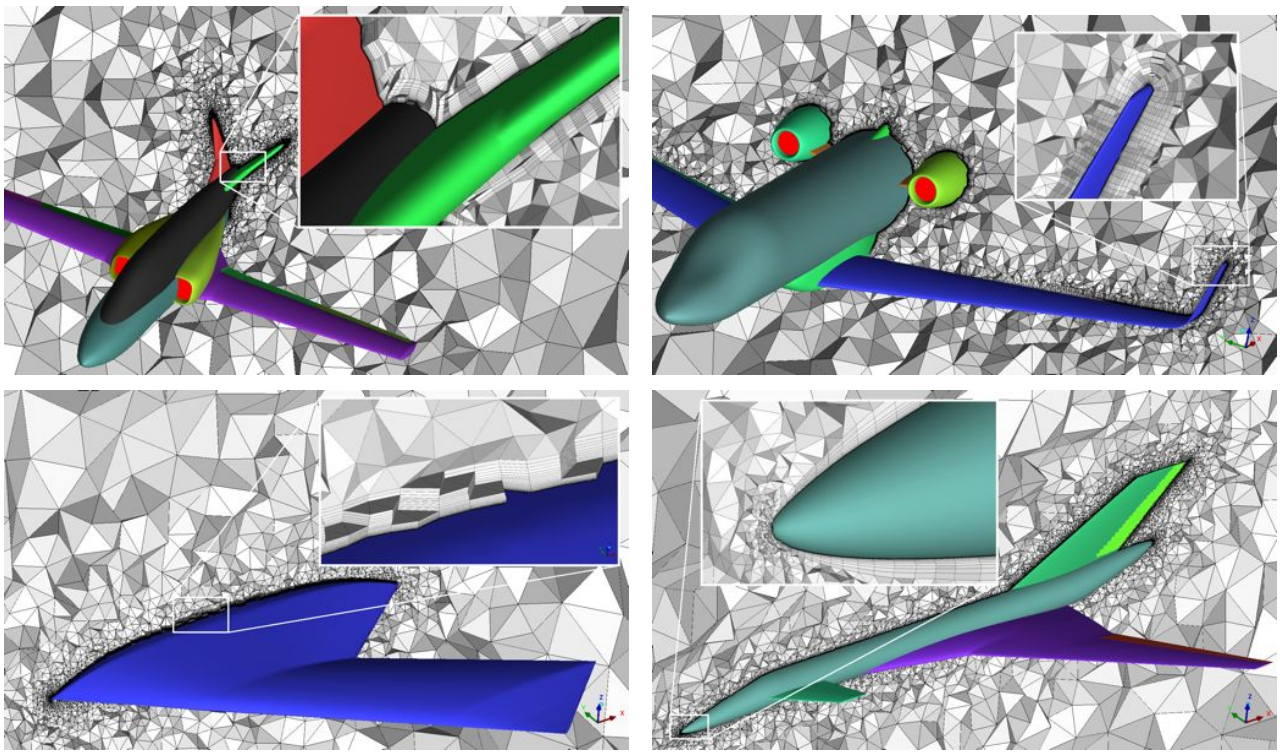


Fig. 7 Example configurations for automatic RANS mesh generation.

7. extracts force coefficients and prepares data for flow visualization from solver output files.

The python program is controlled by a table of flight conditions of interest, which can also contain specific solver parameters should these deviate from default values. As output, the script will generate aerodynamic force and moment coefficients for the flight conditions listed.

5.1 Preprocessing and solver execution

The first step performed by the python script is the settings of proper boundary conditions according to the flight case matrix. This is followed by execution of the solver program (optionally on a remote cluster) and postprocessing operations.

5.1.1 Boundary conditions and parameters

Before the flow solver can be started, a solver parameter file must be written. For this, another python script is used which, starting from a template file containing defaults, inserts the appropriate values for each case to be analysed. Aside from straightforward specification of the flight condition in terms of Mach number, angle of attack, sideslip angle and roll rates, each flight case might also require the setting of control surface angles.

Control surface deflection is simulated by imposing transpiration boundary conditions [12]. With the current version of the Edge solver, this implies performing an pseudo-aeroelastic solution since control surface motion is handled in the same manner as structural deformation. Consequently, it is necessary to generate boundary condition and further supporting files for this prescribed-motion aeroelastic run, which is transparently performed by the python script.

Furthermore, the flight case table may contain numerical solution parameters, such as the CFL number for (pseudo-) time-stepping, iteration numbers, convergence criteria, multigrid parameter or low-speed preconditioning options, which the user may want to change from their default settings for demanding cases. Optionally, automatic solution-based mesh adaptation can be activated, which leads to local refinement in re-

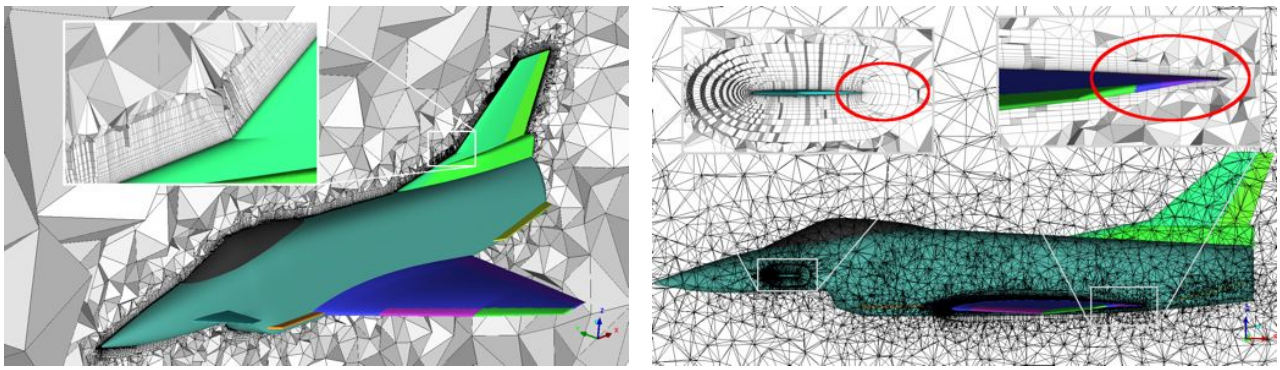


Fig. 8 Difficulties encountered in automated hybrid mesh generation.

gions of strong solution gradients, as shown for a transonic Euler solution in Figure 9. The resulting improvement in solution accuracy is, however, accompanied by a significant increase in computational cost (see Table 1).

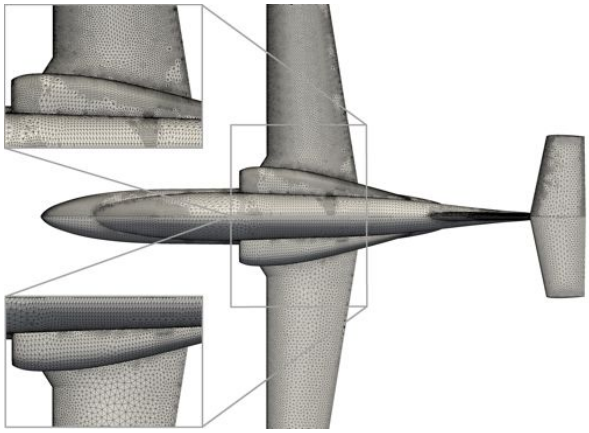


Fig. 9 Adapted mesh for transonic flight case

5.1.2 Work load distribution

Table 1 shows the expected cost from using the different methods. While running a rather dense set of 10,000 cases takes less than 12 hours using the VLM, an automatically generated Euler baseline setup of 500 cases requires around three days of runtime on one eight-core computer. For the higher-resolution meshes, the computing time exceeds one week, which is clearly not feasible in a conceptual design context. To reduce the wallclock time to 12 hours (an overnight run), a compute cluster with at least 144 processor cores would be needed.

The python script used to execute the solver will either submit compute jobs to a shared managed-queue cluster[1] or execute the solver on the local workstation, depending on the number of cases and mesh size. Due to the complexity and differences in the remote communication and file transfer schemes of different computing centres, the job submission code would need to be adapted to exploit different computing resources.

5.2 Postprocessing

Once the solver reports convergence for a certain flight condition, the python management script extracts force and moment coefficients and computes derivatives where applicable. These values are then automatically assembled into a global aerodynamic data table, which, due to the large number of cases, can also be both time-consuming and error-prone if done manually.

For visualization, the open-source system ParaView [13] is employed. ParaView allows extensive scripting of its functionality using python, which will be exploited to automatically generate images such as the contour plot in Figure 10. This module is, however, not completely functional at the time of writing.

6 Example case: Ranger 2000 Jet Trainer

The procedure outlined above is applied to the Ranger 2000 jet trainer for which wind tunnel-

[1] In this case, the high-performance computing center (PDC) at KTH was used.

Method	Elements	nr of cases	Computational effort
VLM	≈ 1000	10,000	$<$ 12h using 1 core
Euler baseline	≈ 1 mio.	500	$\approx$ 72h using 8 cores
Euler refined	≈ 2 mio.	500	$\approx$ 168h using 8 cores
Euler adapted	$\approx 6-12$ mio.	500	$\gg$ 168h using 8 cores
RANS baseline	$\approx 6-12$ mio.	50	$\gg$ 168h using 8 cores

Table 1 Computational effort for different CFD methods

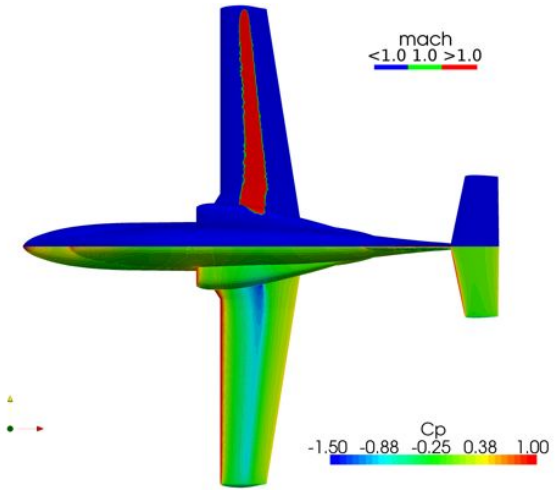


Fig. 10 Ranger at Mach 0.75; Top: local Mach number; Bottom: C_p contours

and flight test data has been made available by the manufacturer [14]. The Ranger is a training aircraft for primary and aerobatic flight training of military jet pilots. It is a mid-wing tandem seat configuration equipped with a Pratt & Whitney JT15 turbofan engine.

6.1 Forces and moments compared to flight test data

The reference area used to normalize aerodynamic coefficients is $S = 15.5m^2$, while reference chord and span are $c = 1.53m$ and $b = 10.5m$, respectively. Moments are computed for a reference point at $(0.44m, 0.0m, -0.11m)$ in the CAD model coordinate system.

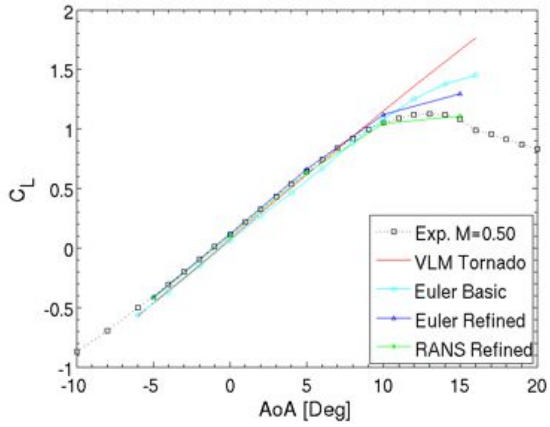
Figure 11 shows a comparison of lift, drag and pitch moment coefficient between experiment and a set of computations at moderate airspeed $(M = 0.5)$ and at near the dive Mach number $(M = 0.75)$. Results for the vortex-lattice and baseline Euler solutions are based on the parametrized CEASIOM geometry shown in Figure 2, while the results for Navier-Stokes solutions and those labeled 'Euler refined' correspond to a somewhat more accurate geometry model created in `sumo` (see Figure 3).

As expected, the linear region lift slope is captured well by all computational methods used. At the higher Mach number, a shock is present on the upper wing surface, which substantially changes the lift curve for $\alpha > 2^\circ$, possibly by inducing boundary layer separation just downstream of the shock. The simulation based on Navier-Stokes equations predicts this behaviour fairly well, while none of the inviscid solutions yields acceptable accuracy outside a narrow band of $-2^\circ < \alpha < 2^\circ$.

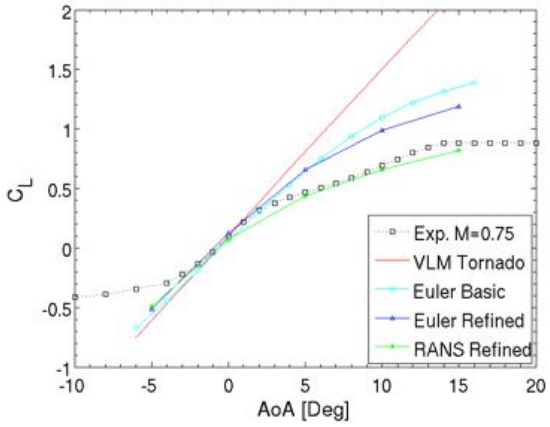
For both drag and moment curves, there is a substantial difference between results for the CEASIOM geometry, which does not include the engine, and the refined, engine-equipped `sumo` model. At the time of writing, it is not clear how the air intake and engine exhaust boundary conditions should be defined in order to better match the experimental setup.

7 Concluding remarks

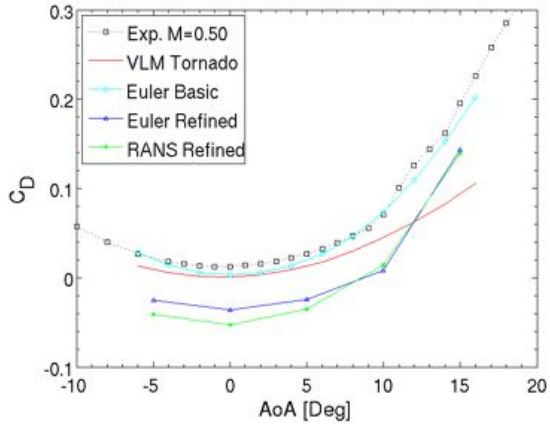
The automated mesh generation and solution process described above has been shown to work fairly well for a wide range of aircraft configurations when targeting inviscid CFD simulations based on the Euler equations. Extension of the process to automated Navier-Stokes solutions has been attempted and found to yield acceptable meshes for many different configurations; however, due to the much more stringent mesh qual-



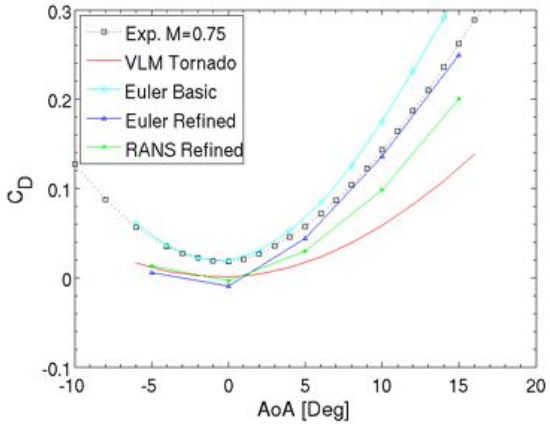
(a) C_L vs Angle of Attack



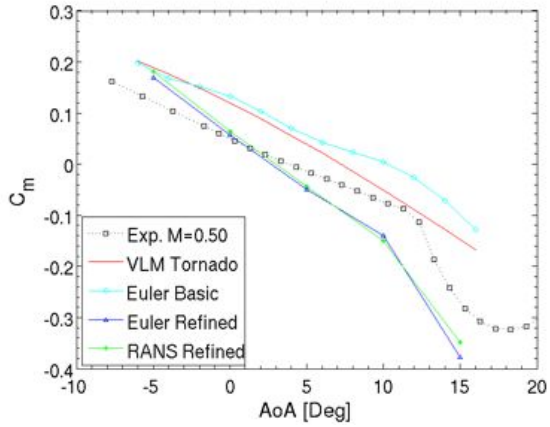
(b) C_L vs Angle of Attack



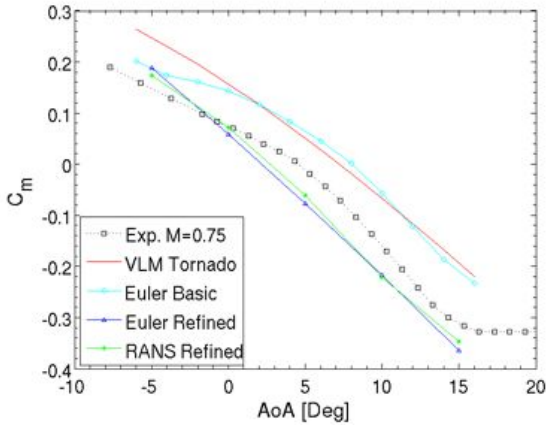
(c) C_D vs Angle of Attack



(d) C_D vs Angle of Attack



(e) C_m vs Angle of Attack



(f) C_m vs Angle of Attack

Fig. 11 Left: Mach 0.5; Right: Mach 0.75

ity requirements and much less robust solution procedures, manual intervention or adaptation of mesh generation and solution parameters is still sometimes necessary. Concerning the solution process itself, it has become obvious that an engine model needs to be used in order to accurately determine the intake and outflow boundary conditions for each flight case.

For the automatic generation of hybrid meshes for Navier-Stokes analyses, the presence of a sharp trailing edge has been identified as a considerable complication. A possible remedy to this problem might be the inclusion of an infinitely thin wake surface extending some distance past the trailing edge. The prismatic mesh could then continue downstream without a large change in the direction of the surface normal, allowing the wake mesh to coarsen progressively. In addition to solving the trailing edge discretization problem, this approach would also lead to a much better resolution of the boundary layer wake (shear layer) in the immediate vicinity of the wing.

8 Acknowledgments

The authors would like to thank Dr. Stefan Hitzel of EADS-MAS who kindly provided the Ranger 2000 data in accessible form.

9 Copyright Statement

References

[1] The SimSAC Project Website. SimSAC Project. http://simsacdesign.org/, July 2009.

[2] A. Rizzi et al. CEASIOM Validation and Its Use in Design - Status of SimSAC Project. In *SAAB Flygtekniksemina rium*, Kolmården, November 2008.

[3] US Product Data Association, Charleston, SC. *Initial Graphics Interchange Standard 5.3*, September 1996. Available from www.uspro.org.

[4] H. Si and K. Gaertner. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, pages 147–163, San Diego, September 2005. Software available from tetgen.berlios.de.

[5] AIAA CFD Committee on Standards. The CFD General Notation System – Standard Interface Data Structures. Technical Report R-101A-2005, AIAA, 2005.

[6] ANSYS, Inc. *ANSYS ICEM CFD 12.1 USER MANUAL*, 2009. http://www.ansys.com.

[7] L. P. Chew. Guaranteed-Quality Mesh Generation for Curved Surfaces. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, San Diego, May 1993. Available from the ACM Digital Library, portal.acm.org/citation.cfm?id=161150.

[8] Tomas Möller. A fast triangle-triangle intersection test. *journal of graphics tools*, 2(2):25–30, 1997.

[9] T. Cebeci and J. Cousteix. *Modeling and Computation of Boundary Layer Flows.* Springer, 2nd edition, 2005.

[10] L. Tysell, T. Berglind, and P. Eneroth. Adaptive Grid Generation for 3D Unstructured Grids. In *6th International Conference on Numerical Grid Generation in Computational Field Simulations*, June 1998.

[11] P. Eliasson. Edge, a Navier-Stokes solver for unstructured grids. In *Finite Volumes for Complex Applications III*, pages 527–534, June 2002.

[12] C. C. Fisher and A. S. Arena Jr. On the Transpiration Method for Efficient Aeroelastic Analysis Using an Euler Solver. In *AIAA Atmospheric Flight Mechanics Conference*, San Diego, CA, July 1996. AIAA-1996-3436.

[13] A. Cedilnik, B. Geveci, K. Moreland, J. Ahrens, and J. Favre. Remote Large Data Visualization in the ParaView Framework. In A. Heirich, B. Raffin, and L. P. Santos, editors, *Eurographics Parallel Graphics and Visualization*, pages 162–170, May 2006.

[14] RANGER 2000 FR06/RP01 Aerodynamic Dataset, Release 1.1. Technical Report TN-R-R-002-M-0011, DASA, 1994.