European Congress on Computational Methods
in Applied Sciences and Engineering (ECCOMAS 2012)
J. Eberhardsteiner et.al. (eds.)
Vienna, Austria, September 10-14, 2012

# 3D ANISOTROPIC DELAUNAY MESHING FOR IDEAL INTERFACING TO BLOCK-UNSTRUCTURED MIXED MESHES USING A SPARSE OCTREE FOR METRIC SIZE PROPAGATION

**Jochen Wild** [1]

[1] DLR Institute of Aerodynamics and Flow Technology
Lilienthalplatz 7, 38108 Braunschweig, Germany
Jochen.Wild@dlr.de

**Keywords:** grid generation, tetrahedron generation, anisotropic triangulation, mixed meshes

**Abstract.** *Generating smooth mixed block-unstructured meshes is a promising way to reduce mesh sizes for outer aerodynamic flow analysis. Beyond the core part of the mesh generation, the smooth meshing of the boundary layer region by structured hexahedrons and prisms, for the remaining flow field a triangulation method is needed that allows for a smooth transition between the different element types. This paper outlines the details of the final brick of the mixed meshing method, an anisotropic triangulation method to generate bounding element conforming tetrahedral meshes. A special emphasis is put on a new sparse octree method for memory and time efficient propagation of the underlying propagation of the anisotropic metric information in the mesh domain. The contribution presents modifications to a Delaunay-type meshing kernel together with a special sparse octree structure for memory efficient propagation of anisotropic size information. The benefits of the method are outlined for two common aerodynamic configurations, the ONERA M6 wing and the Boeing CRM configuration, known from the 4th AIAA drag prediction workshop.*

## 1 INTRODUCTION

Numerical design optimization is an emerging design tool for real world applications in aerodynamics. The state of the art of 3D Navier-Stokes solvers already show a good predictive capability even for more complex 3D aircraft configurations as shown for example in [1]-[3].

Optimization based on high-level computational fluid dynamics (CFD) tools using Reynolds-averaged Navier-Stokes equations (RANS) in the meantime is widely used for 2D airfoil design. Today the step towards 3D wing optimization is to be done. Although already practical for simple configurations like wing-body, the effort of calculating more complex configurations is still too high for application within an optimization environment. The main limiting factor hereby is the number of grid points necessary for accurate flow simulation using hybrid unstructured grids consisting of tetrahedral and prismatic elements. E.g. for the configuration used in the 1st AIAA Drag Prediction Workshop [1] the medium hybrid unstructured grid used giving promising results contained approximately 12 million grid points. This resulted in a computational effort of 100 hours CPU-time on a NEC SX5 high performance computer for a given flow condition. Even the grid generation for this case took about 1 day turn-around time. This computational effort is clearly unacceptable in terms of an optimization process, where multiple configurations have to be evaluated. As a rough estimate, when assuming 200 evaluations for a converged optimization, this will lead linearly to 200 days grid generation and 800 CPU-days flow calculation.

Regarding applications in high-lift optimization the situation worsens. Today, hybrid unstructured meshes for transport aircraft configurations contain about 20-30 Million grid points. A mesh is usually generated over night when using an advancing front method for high quality meshes. Even with the new computer technology providing massively parallel LINUX cluster systems, 1 to 5 days are needed for a fully converged flow simulation. Using such meshes would require 40 to 1000 days of computing, depending on the optimization algorithm

The big shortcoming of hybrid unstructured grids is the low anisotropy of surface triangles resulting in a large number of grid points, which is agglomerated through the number of prismatic layers for the boundary layer resolution. This low anisotropy leads to an unnecessary high resolution in span direction, especially for high aspect ratio wings. Recovering the experience with the application of structured grids, it is known, that the aspect ratios of the surface quadrilaterals can be much higher, additionally resulting in well aligned body conforming meshes. The shortcoming of an overall use of structured meshes is the increasing complexity of the targeted configurations, where structured meshing reaches its limitations, mainly due to grid topology issues.

A second bottleneck when generating meshes for aerodynamic analysis of aircrafts arise for the memory requirements of most of the methods. For propagating size information to guide the meshing process it is common to use octree type structures. These methods are fast and easy to implement and rely on bi-sectioning the overall domain wherever detailed size information is available. The details of the method can be found in text-books. Unfortunately, when regarding aerodynamic configurations there is a large difference between the requested shortest and largest edge lengths. The shortest edges in such a mesh arise from the first cell distance required for a proper resolution of the boundary layer, which is usually in the order of $1.0x10^{-5}$ and $1.0x10^{-6}$ related to the aerodynamic mean chord. On the other hand the far-field has to be located far away from the configuration, where 100-500 times the aerodynamic mean chord is a common recommendation. Due to the bi-sectioning an octree covering this

domain will have about 25 levels. On smaller desktop computers this can lead to a non-coverable memory requirement and a failure of mesh generation.

In the past the author et al. [4]-[7] reported new strategies to generate mixed meshes to significantly reduce mesh sizes without reduction of accuracy. The method relies on generating smooth structured hexahedral and prismatic elements for the resolution of boundary layers by a parabolic marching routine based on the face-weighted Laplace equation for a proper alignment and anisotropic stretching of cells resolving viscous flow regimes. Details of the method to generate structured hexahedral and prismatic mesh blocks are described there. This paper outlines the details of the final brick of the mixed meshing method, an anisotropic triangulation method to generate bounding element conforming tetrahedral meshes. A special emphasis is put on a new sparse octree method for memory and time efficient propagation of the underlying propagation of the anisotropic metric information in the mesh domain.

## 2 METRIC REPRESENTATIONS OF BOUNDING ELEMENTS

The purpose is to adopt a volume triangulation in a way that the cells at the boundaries match the size and the orientation of the underlying hexahedral and prismatic elements. The usually high stretching of the hexahedral cells requires the use of an anisotropic triangulation method. For anisotropic meshing we need to store the information in terms of tensors. The metrics can be stored either as a tensor describing the ellipsoids of density prescription or as transformation matrix describing the transformation into the metric space. The metric itself is a symmetric positive definite tensor which can be decomposed into the eigenvector matrix **E** and eigenvalues $\lambda_i$ according to

$$\mathbf{M} = \mathbf{E} \cdot diag\left(\lambda_i\right) \cdot \mathbf{E}^T . \tag{1}$$

The duality to the transformation **T** into metric space is given by

$$\mathbf{T} = diag\left(\sqrt{\lambda_i}\right)\mathbf{E}^T . \tag{2}$$

The metrics resulting from the surrounding cells, including the block-structured hexahedral cells must be specified. There are three ways to de-rive space metrics from the surrounding elements.

### 2.1 Hexahedral metrics

For the attachment to hexahedral elements we use the principal axis of the hexahedron for the specification of the metrics. For this purpose we form the non-orthogonal basis $\mathbf{B}_H$ by averaging the edge vectors along the *i*, *j*, and *k*-directions in structured notation.

$$\mathbf{B}_H = \left[\mathbf{e}_i \quad \mathbf{e}_j \quad \mathbf{e}_k\right]; \quad \begin{aligned} \mathbf{e}_i &= \frac{1}{4}\sum_{j=1}^{2}\sum_{k=1}^{2}\left(\mathbf{x}_{i+1,j,k} - \mathbf{x}_{i,j,k}\right) \\ \mathbf{e}_j &= \frac{1}{4}\sum_{i=1}^{2}\sum_{k=1}^{2}\left(\mathbf{x}_{i,j+1,k} - \mathbf{x}_{i,j,k}\right) \\ \mathbf{e}_k &= \frac{1}{4}\sum_{i=1}^{2}\sum_{j=1}^{2}\left(\mathbf{x}_{i,j,k+1} - \mathbf{x}_{i,j,k}\right) \end{aligned} \tag{3}$$

The corresponding symmetric metrics is then simply obtained by

$$\mathbf{M}_H^{-1} = \mathbf{B}_H \mathbf{B}_H^T . \tag{4}$$

### 2.2 Affine invariant

The affine variant metric [8] describes the least square fit of an ellipse in 2D, or an ellipsoid in 3D, through a given set of points, centred at the centre point of the point set. The affine invariant metric is well defined if the number of non-collinear, resp. non-coplanar, points is larger than 2*N*+1.

$$\mathbf{M}_{A_{ij}}^{-1} = \frac{1}{n}\sum_{k=1}^{n}\left(\mathbf{x}_{ik} - \mathbf{x}_{ic}\right)\left(\mathbf{x}_{jk} - \mathbf{x}_{jc}\right)$$
$$\mathbf{x}_{ic} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_{ik} \tag{5}$$

where *N* is the dimension of the space. This method is suitable for prismatic or pyramidal elements.

### 2.3 Riemannian invariants

In case of neighbouring cells being tetrahedrons, the metric is derived by using the Riemannian invariant metric of the neighbouring cell. The Riemannian invariant metric is the metric that transforms an arbitrary triangle in 2D or a tetrahedron 3D in an ideal one with unity edge length (Figure 1). This leads to solving a system of *N*! equations of the form

$$\left(\mathbf{x}_i - \mathbf{x}_j\right)^T \mathbf{M}_R \left(\mathbf{x}_i - \mathbf{x}_j\right) = \mathbf{1}, \quad i < j; \quad i, j = 1 \ldots N+1, \tag{6}$$

where again *N* is the space dimension.

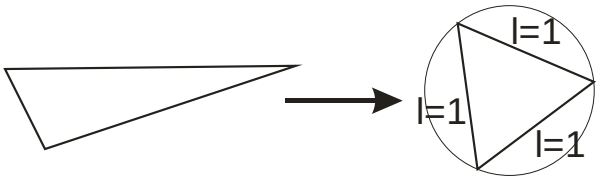



Figure 1: Definition of Riemannian invariant transformation

Figure 2 shows exemplarily the anisotropic metric visualized by ellipsoids derived from the different types of three-dimensional bounding elements

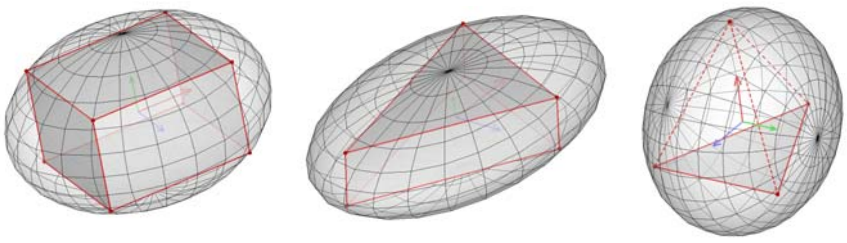


Figure 2: Ellipsoids showing the derived anisotropic metric of different types of cells

## 3 A SPARSE OCTREE FOR SIZE AND ANISOTROPIC METRIC SIZE PROPAGATION

A prerequisite for the anisotropic meshing tetrahedral meshing by Delaunay method is the prescription of a smooth metric space, since the existence of such a triangulation for an arbitrary metric space is not proven. As explained before, for aerodynamic applications a more

efficient way to implement the octree is needed to overcome memory restrictions due to the large number of needed octree levels to resolve the meshing domain properly.

In the following the details of the proposed sparse octree representation for the storage and propagation of size and directionality information for anisotropic meshing are outlined. The sparse octree data structure is based on non-interlaced Morton encoding. Knowing the desired accuracy of the octree a unique key to the desired box containing a desired coordinate location is easily evaluated by integer division. This allows for two major features: a) bottom-up construction of the octree, and b) a very efficient top-down search within the octree.

### 3.1 Non-interlaced Morton encoding

The bottom up generation of the octree was proposed by Dawes et al. 20 using Morton encoding. The shortcoming is that the number of available levels is limited by the length of an integer string divided by the dimension of the domain. We use a non-interlaced Morton encoding (Figure 3), where for each coordinate direction a separate string is used. Using 64bit integers for the encoding allows for 64 octree levels.

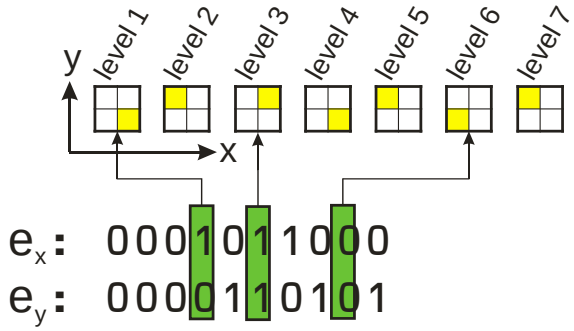



Figure 3: Non-interlaced Morton encoding on a 10bit integer string for a 7-level octree

The encoding of physical coordinates into the Morton encoding integer strings is simply performed using the integer part of the division of the coordinate value by the desired accuracy $\varepsilon$ of the octree

$$e_{x_i} = (\text{int})\left(x_i - \min(x)\right)/\varepsilon \,. \tag{7}$$

The only prerequisite for using this type of encoding is the a priori knowledge or specification of the minimum values in each dimension and the desired accuracy. Nevertheless, the accuracy can be increased by a subdivision algorithm or decreased by removing high-level octree cells afterwards. It should be reminded that the Morton encoding strings change in this case, and therefore should not be stored directly.

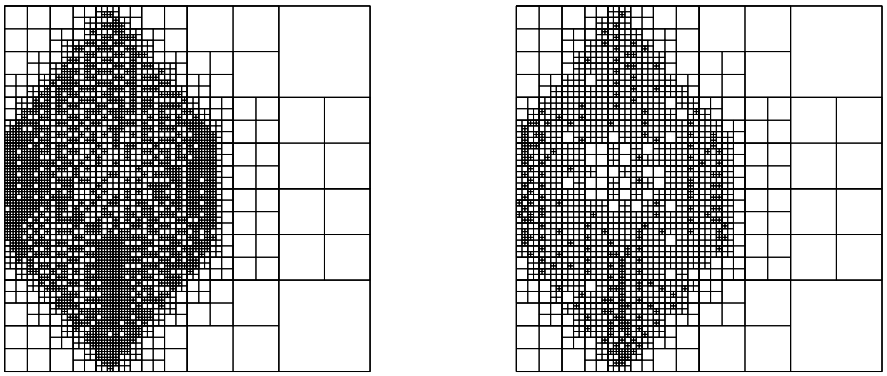
Morton encoding has several advantages. Once the integer strings are defined the movement through the octree data structure only requires bit operations (bitwise AND, OR, and XOR; left bit shifts), which are very fast compared to floating point comparisons. Additionally, to locate a box the octree is only stepped through top-down. For nearest neighbour search the number of steps up and down the octree is immediately identified by an XOR operation on the bit strings.

### 3.2 Octree balancing

We use the usual strategy to balance the octree in a way that every box only has neighbours on the same level or maximum one level higher or lower. The balancing is neces-

sary afterwards for obtaining smooth sizing information throughout the covered domain. To obtain the balancing, after seeding the octree with the prescribed sizing information – and thus generating the octree cells at the highest (finest) level, all cells in the tree are stepped through from highest to lowest level and the octree level of the neighbouring cells are evaluated. In case the level difference is too high the corresponding coarse cell is subsequently subdivided.

### 3.3 Spare box removal



a) full octree b) spare boxes removed

Figure 4: Effect of spare box removal

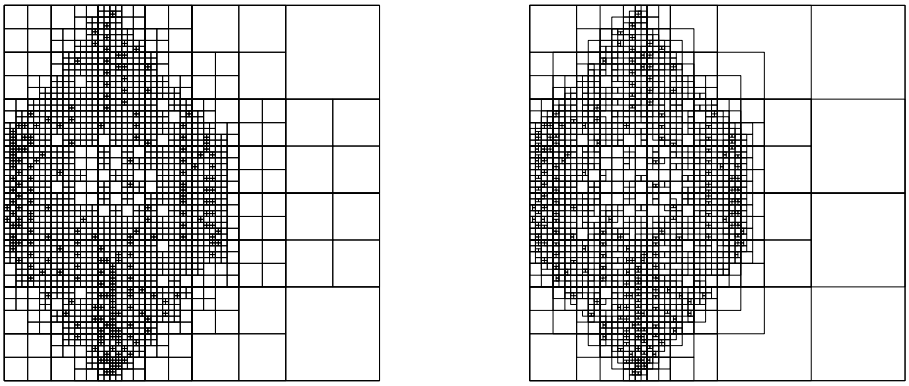
In order to build the octree the information is firstly stored only on the lowest level. If the accuracy is set lower than $\sqrt{n}/2$ times the minimum distance between two input points, every input point is stored in a separate box on the lowest level. As a result a large number of octree divisions exist, where only one child box contains information. These subdivisions can all be eliminated without changing the stored information (Figure 4) by simply propagating the information of a child box to the next higher level and deleting the existing child boxes if it is the only child box of the parent containing information.

### 3.4 Sparse subdivision

In usual octree data structures if an octree box is subdivided it is divided into $2^d$ boxes, regardless if all child boxes will be filled with data. We propose to only subdivide octree boxes into the required children (Figure 5). A test on an input data set of 10000 points shows that the data structure to isolate each point in a separate box in a 12 level octree requires only 10MB of memory, while the full representation would need about 200MB. As shown later, the information stored in the removed nodes of the octree can be reconstructed with only minor numerical effort.

### 3.5 Location search

Searching for a box specifying the information at a desired location is a straight forward approach similar to information insertion. The coordinates of the desired location are encoded into the *n* integer strings of the Morton encoding. Bit arithmetic for each level directly addresses the designated child. The octree is stepped top-down until a box is reached, where the next designated child is not available, which is the desired box containing the information.



a) full subdivision b) sparse subdivision

Figure 5: Sparse subdivision octree structure

### 3.6 Implementation

The octree data structure is implemented as a C++ template, depending on the dimension of the problem and the type of attribute to be propagated (size or metric). The benefit is to have a consistent implementation for any type of size propagation within meshing, either 2D or 3D. Changing the type of attribute only requires overloading of the attribute dependent arithmetic operations.

The meshing process needs all three representations of the anisotropic metric represented by the metric tensor, the transformation matrix, and its inverse, which is the size and orientation information itself. During the setup of the octree, the metric information propagation, and the evaluation of the local prescribed anisotropic metric it is the most often required to evaluate the metric tensor itself, while the transformation matrix is only required in a few cases. Therefore it is beneficial to store the metric itself and not another of the possible representations.

## 4 ANISOTROPIC SIZE PROPAGATION USING SPARSE OCTREE STRUCTURE

### 4.1 Metric propagation

In order to specify a smooth metric field with a specified grading we apply the mixed-space metric propagation method of Alauzet [10] making use of a Euclidian-log size growth. Given a gradation factor γ, two different propagation types are used. The first one, the metric space gradation

$$\mathbf{M}_M\left(\mathbf{x}+d\mathbf{x}\right)=\mathbf{E}\left[\frac{1}{\left(1+\sqrt{d\mathbf{x}^T\mathbf{M}\left(x\right)d\mathbf{x}}\ln\left(\gamma\right)\right)^2}diag\left(\lambda_i\right)\right]\mathbf{E}^T \tag{8}$$

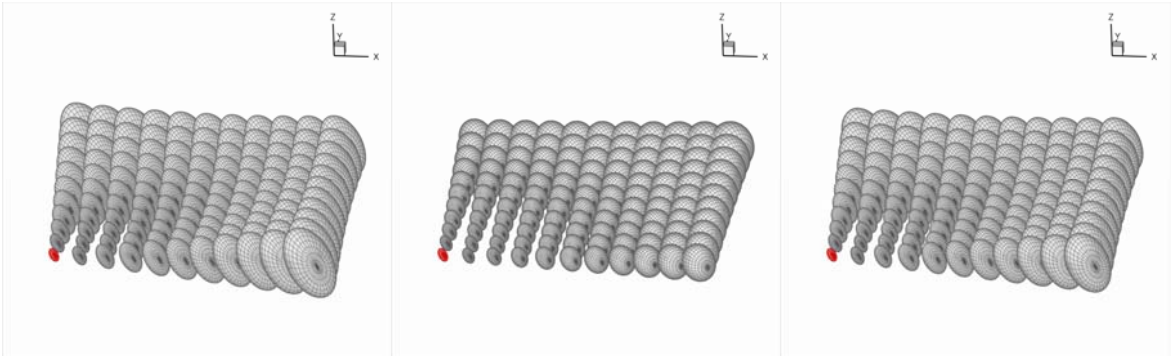
grades the size of the ellipsoids in the field, but is not changing the aspect ratio or the directionality. The second, the physical space gradation

$$\mathbf{M}_P\left(\mathbf{x}+d\mathbf{x}\right)=\mathbf{E}\left[diag\left(\frac{1}{\left(1+\sqrt{\lambda_i}\left\|d\mathbf{x}\right\|_2\ln\left(\gamma\right)\right)^2}\lambda_i\right)\right]\mathbf{E}^T \tag{9}$$

grades the eigenvalues independently and by this getting the corresponding ellipsoid more and more isotropic with increasing distance. Alauzet proposes a combination of both in the form of

$$\mathbf{M}\left(\mathbf{x}+d\mathbf{x}\right)=\mathbf{E}\left[diag\left(\frac{1}{\left[\left(1+\sqrt{\lambda_i}\left\|d\mathbf{x}\right\|_2\ln\left(\gamma\right)\right)^t\left(1+\sqrt{d\mathbf{x}^T\mathbf{M}\left(x\right)d\mathbf{x}}\ln\left(\gamma\right)\right)^{1-t}\right]^2}\lambda_i\right)\right]\mathbf{E}^T, \tag{10}$$

which we use with a value for the mixing exponent $t$ = 3/4. Figure 6 compares the propagation of a metric (red ellipsoid) using metric, physical and a mixed space gradation using $t$ = 1/4 for emphasizing the differences.



a) metric space (t=0) b) physical space (t=1) c) mixed space (t=1/4)

Figure 6: Gradation of metric size propagation (γ=1.05)

### 4.2 Metric intersection

There are two different methods of Metric intersection available. We found, that the metric intersection by simultaneous reduction published by Alauzet [10] and Dobrzynski and Frey [11] has the shortcoming that the self intersection of a metric gives an unexpected result. We use the method published by McKenzie et al. [12], where one metric $\mathbf{M}_1$ is transformed into the space of the second metric $\mathbf{M}_2$.

$$\tilde{\mathbf{M}}_1=\left(\mathbf{T}_2^{-1}\right)^T\mathbf{M}_1\mathbf{T}_2^{-1};\quad \tilde{\mathbf{M}}_1=\tilde{\mathbf{E}}_1 diag\left(\tilde{\lambda}_{1_i}\right)\tilde{\mathbf{E}}_1^T. \tag{11}$$

Intersection is performed in the metric space by limiting the eigenvalue to be smaller than or equal to 1.

$$\mathbf{M}_I=\mathbf{T}_2^T\tilde{\mathbf{E}}_1 diag\left(\max\left(\tilde{\lambda}_{1_i},1\right)\right)\tilde{\mathbf{E}}_1^T\mathbf{T}_2. \tag{12}$$

This intersection is commutative and has no singularity if both metrics are equal. Figure 7 shows an example of the intersection of two metrics in 3D.

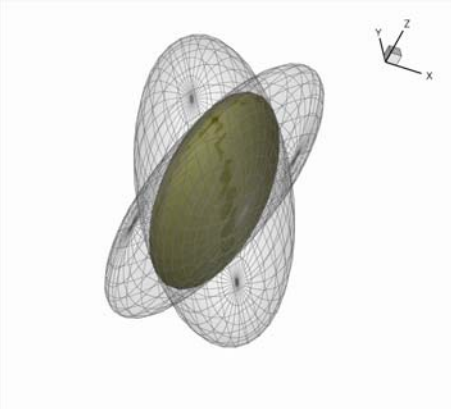


Figure 7: Intersection of two metrics

### 4.3 Metric interpolation

For metric interpolation we use the Euclidian-log-space interpolation proposed by Alauzet [10]

$$\mathbf{M}(x) = e^{\sum_{i=1}^{k} \alpha_i \ln(\mathbf{M}(x_i))}$$
$$\ln(\mathbf{M}) = \mathbf{E}\,diag\left(\ln(\lambda_i)\right)\mathbf{E}^T; \quad e^{\mathbf{M}} = \mathbf{E}\,diag\left(e^{\lambda_i}\right)\mathbf{E}^T; \quad \sum_{i=1}^{k} \alpha_i = 1 \tag{13}$$

where the weightings are the same as for a linear interpolation. Figure 8 shows an example of the planar interpolated metric field based on 4 metrics at the corners (red ellipsoids)

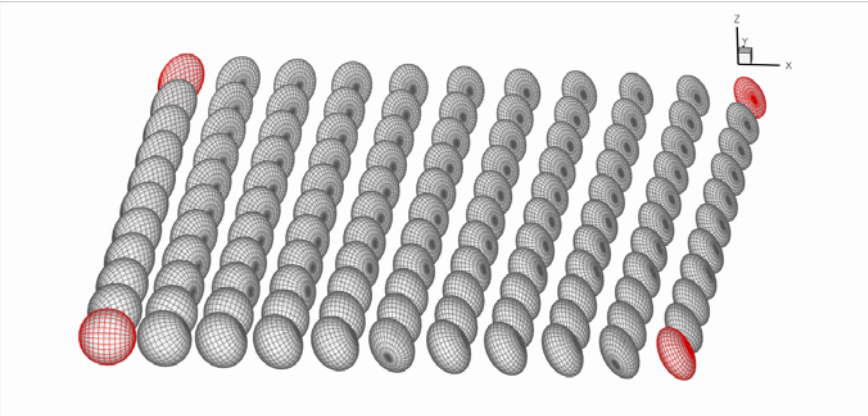


Figure 8: Planar interpolation of metrics (k=4, red ellipsoids)

Special attention has to be paid to the sparse octree representation, since a box, where a desired location for metric interpolation falls in, may contain additional information of child boxes. Since the octree is balanced it is suitable to always perform a sub-level interpolation on a level lower than the box containing the box unless it is already the finest one. First the available corners are selected. Second, the still missing corners of the not-existing child box could be interpolated first on the side faces and edges. But this double interpolation is not evaluated directly. Instead the weightings of the corner points are adjusted. Figure 9 shall help to explain the procedure. The yellow node is the desired interpolation point and the red nodes are the nodes existing in the sparse octree structure. The point **p** falls in a box that contains addi-

tional information on level $m$+1, which would be neglected if the box corner nodes would be used directly.

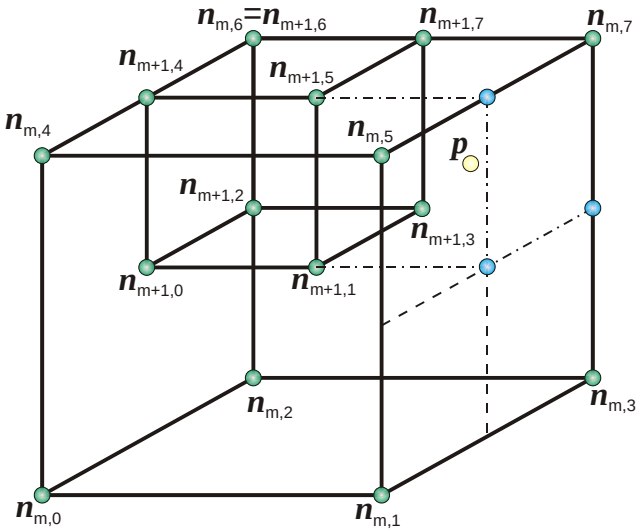



Figure 9: Sub-level interpolation on sparse octree structure

If no sub-level information is available a linear interpolation of **p** based on **n** can be written in vector form

$$\mathbf{p} = \alpha_i \mathbf{n}_i = \begin{pmatrix} (1-\eta)(1-\xi)(1-\zeta) \\ (\eta)(1-\xi)(1-\zeta) \\ (1-\eta)(\xi)(1-\zeta) \\ (\eta)(\xi)(1-\zeta) \\ (1-\eta)(1-\xi)(\zeta) \\ (\eta)(1-\xi)(\zeta) \\ (1-\eta)(\xi)(\zeta) \\ (\eta)(\xi)(\zeta) \end{pmatrix} \bullet \begin{pmatrix} \mathbf{n}_0 \\ \mathbf{n}_1 \\ \mathbf{n}_2 \\ \mathbf{n}_3 \\ \mathbf{n}_4 \\ \mathbf{n}_5 \\ \mathbf{n}_6 \\ \mathbf{n}_7 \end{pmatrix}, \tag{14}$$

where the interpolating coefficients $\eta, \xi, \zeta$ are the remains of the integer division already performed during the decomposition into the encoding strings.

$$\begin{pmatrix} \eta \\ \zeta \\ \xi \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} = \mathbf{x} - \left( (\mathrm{int}) \left( \mathbf{x} - \min(\mathbf{x}) \right) / \varepsilon \right) \tag{15}$$

Further, we use the indices of the nodes starting at 0 so they are identical to the vertical column of the corresponding encoding string. For the sub-level interpolation we would have to interpolate the blue nodes first to get a squared box around the interpolation point. Since it is always a bi-sectioning we can write this in more detail for the example in the figure as

$$\mathbf{p} = \boldsymbol{\alpha}_{m+1} \bullet \begin{pmatrix} \mathbf{n}_{m+1,1} \\ 0.25\mathbf{n}_{m,1} + 0.25\mathbf{n}_{m,3} + 0.25\mathbf{n}_{m,5} + 0.25\mathbf{n}_{m,7} \\ \mathbf{n}_{m+1,3} \\ 0.5\mathbf{n}_{m,3} + 0.5\mathbf{n}_{m,7} \\ \mathbf{n}_{m+1,5} \\ 0.5\mathbf{n}_{m,5} + 0.5\mathbf{n}_{m,7} \\ \mathbf{n}_{m+1,7} \\ \mathbf{n}_{m,7} \end{pmatrix}, \tag{16}$$

which can be also written as vector-matrix-vector product of the form

$$\mathbf{p} = \boldsymbol{\alpha}_{m+1} \mathbf{L} \tilde{\mathbf{n}} \tag{17}$$

For the case depicted in Figure 9 the sparse interpolation modifier **L** writes

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0.25 & 0 & 0.25 & 0 & 0.25 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and the node vector is a mixture of nodes on the box level *m* and the neighbouring nodes on the sub-level *m*+1

$$\tilde{\mathbf{n}}^T = \begin{pmatrix} \mathbf{n}_{m+1,1} & \mathbf{n}_{m,1} & \mathbf{n}_{m+1,3} & \mathbf{n}_{m,3} & \mathbf{n}_{m+1,5} & \mathbf{n}_{m,5} & \mathbf{n}_{m+1,7} & \mathbf{n}_{m,7} \end{pmatrix} \tag{18}$$

Each line in **L** corresponds to the linear interpolation coefficients for the corner points of the virtual sublevel box. To finally perform the metric interpolation in the Euclidean-log-space on a sparse box we replace the vector of weighting coefficients in eq. by the vector-matrix product

$$\alpha_i = \boldsymbol{\alpha}_{m+1} \mathbf{L} \tag{19}$$

and use the metric information at the neighbouring nodes selected as described above.

## 5 ANISOTROPIC FIELD TRIANGULATION

Although Labelle and Shewchuk [13] showed that there is no guarantee for the existence of an anisotropic Delaunay triangulation on the basis of a general non-uniform metric space, we believe that practically it can be obtained in any case where the variation of the metric is smooth enough in space.

As the kernel for the volume field triangulation we make use of the SIMMETRIX code [15]. The desired anisotropic size information stored in the sparse octree structure described above was linked towards the algorithm by providing a user defined sizing function which is called by the kernel throughout the mesh generation process. Initial test to only provide the

anisotropy at the boundary alone failed and underlined the provision of a smooth metric field for successful meshing. For the purpose of this work, the SIMMETRIX code is integrated into our in-house mesh generation software MegaCads [16], which has originally been a pure multi-block structured mesh generator.

## 6 RESULTS

In order to assess the properties of the proposed methods, in the following three different meshing cases are shown, starting from a generic cube and followed by two relevant aerodynamic configurations, the ONERA M6 wing and the Boeing CRM configuration. Finally also the memory efficiency is evaluated for 2 of these cases.

### 6.1 Cube

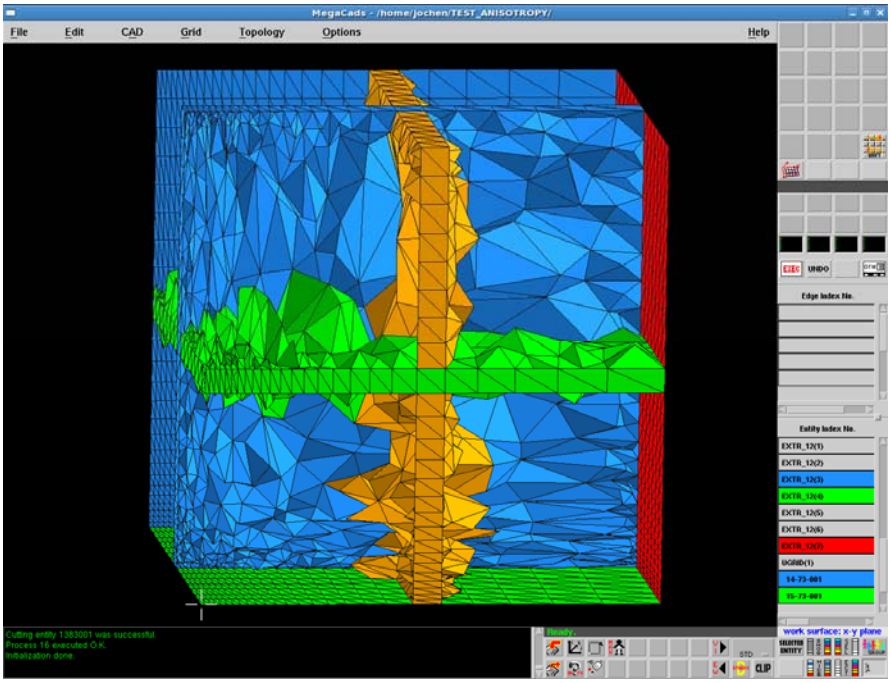



Figure 10: Slices through anisotropic triangulation of unit cube with the anisotropy defined by the boundary elements of a corresponding structured hexahedral mesh of the cube

We defined a unit cube filled with a structured mesh of variable stretching. The structured boundary faces were triangulated and the metric of the original hexahedral elements at the boundaries were used for the metric space. The surface mesh consists of 2752 input points. Figure 10 shows parts of the boundary mesh and slices through the anisotropic volume mesh. It is clearly visible that the tetrahedral triangulation is well adapted to the anisotropy of the bounding surface mesh.

## 6.2 ONERA M6 wing

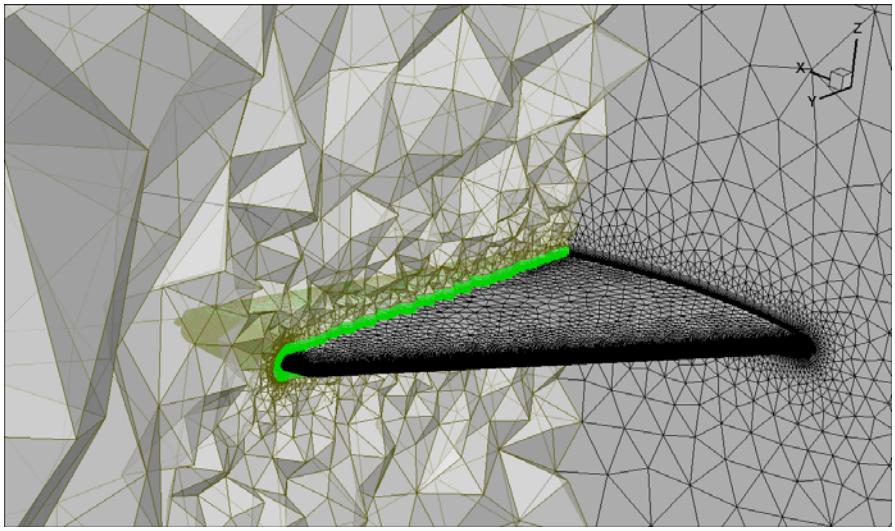



Figure 11: Reference hybrid mesh around the ONERA M6 wing (O-type boundary layer grid); prisms: green; tetrahedrons: red.

The aerodynamic configuration where we demonstrate the capabilities of the chosen mixed mesh approach is the well known ONERA M6 wing where measurements at transonic conditions are available for a range of angles of attack [17]. In order to assess the different ingredients of the meshing procedure three different meshes are generated and used in the calculations. In all cases a special a priori knowledge of shock positions was neglected to pronounce more the differences in the different approaches. First, as a reference, a hybrid mesh consisting only of prismatic and tetrahedral elements was generated using the available software CENTAUR [18] (Figure 11). This represents the state of the art of mesh generation used at our institute and generates high-quality, but isotropic meshes. The second mesh uses the mixed mesh approach without passing anisotropic into the tetrahedral space (Figure 12). Finally, the third grid uses the complete approach including the generation of anisotropic tetrahedrons using the metric information from hexahedral and prismatic elements at the interfaces (Figure 13).

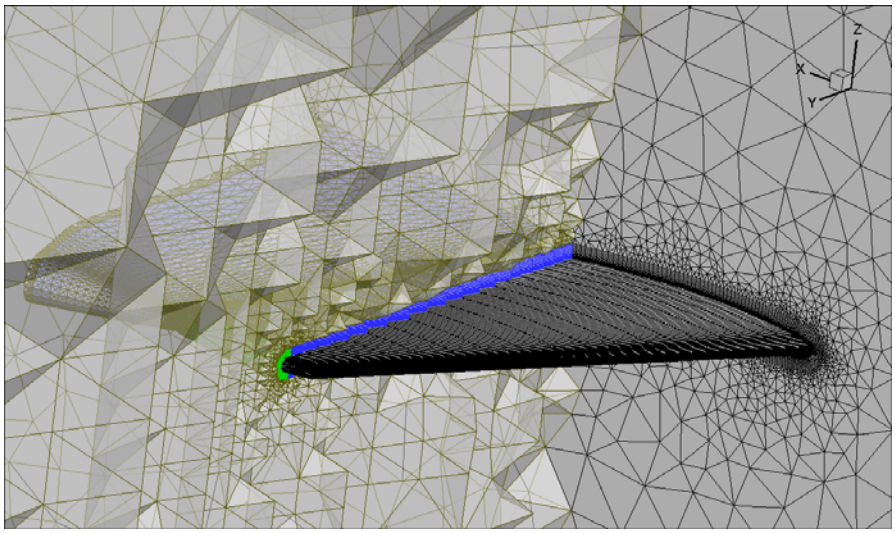



Figure 12: Application of mixed mesh method with isotropic tetrahedrons to the ONERA M6 wing (C-type boundary layer and wake grid); hexahedrons: blue; prisms: green; tetrahedrons: red.

The comparison in this work concentrates on the benefits of using structured highly stretched hexahedral elements for shear layer resolution. The surface resolution has been chosen similar to what has been used in structured or unstructured meshing practice. The wall normal resolution is the same for all meshes with 32 structured/prismatic layers and a first wall spacing for obtaining dimensionless wall distances around $y^+$=1. Due to the different methodology the mixed meshes allow for a resolution of the wing wake with C-type hexahedral grids while the hybrid approach is only capable for resolving this region with an O-type topology. No emphasis has been put on resolving possible shock positions since this would represent an a priori knowledge of the flow. Additionally a possible weak shock capturing can pronounce the differences in the different grid structures.

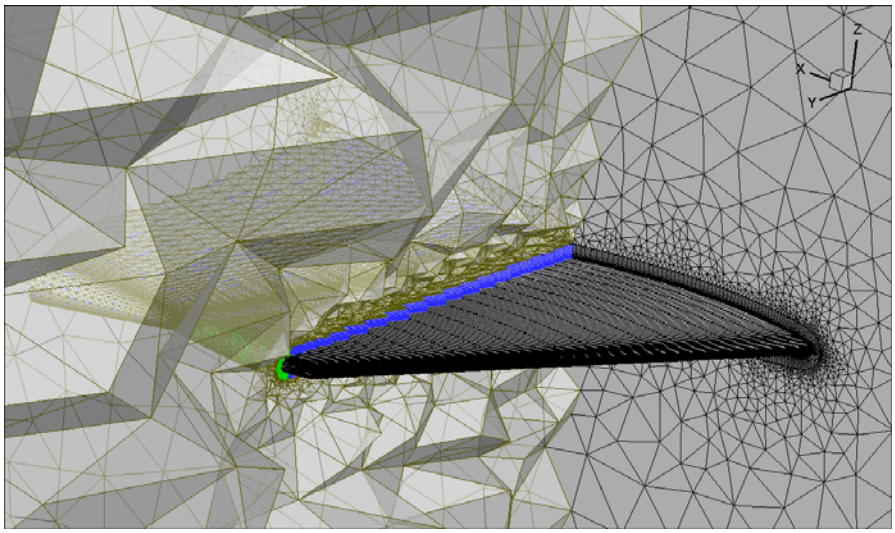



Figure 13: Application of mixed mesh method with anisotropic tetrahedrons to the ONERA M6 wing (C-type boundary layer and wake grid); hexahedrons: blue; prisms: green; tetrahedrons: red.

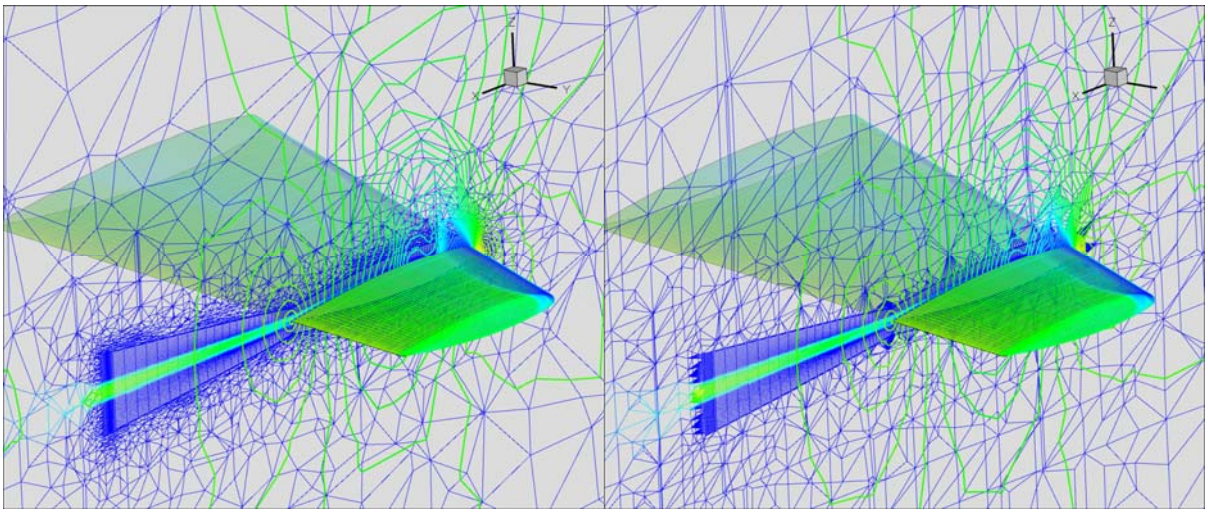
| | points | surface elements | hexahedrons | tetrahedrons | prisms | pyramids |
|---|---|---|---|---|---|---|
| mixed-anisotropic | 456212 | 32826 | 191828 | 467445 | 316352 | 41752 |
| mixed-isotropic | 445310 | 32826 | 191828 | 416421 | 316352 | 41752 |
| hybrid | 1916138 | 113702 | - | 974515 | 3443584 | - |

Table 1: Mesh sizes and characteristics of the meshes used for the ONERA M6 wing calculations

The sizes of the meshes are reported in Table 1. The mixed meshes both only have slightly more than 20% of the points used in the hybrid grid making use of a high stretching in span direction. The adoption of the tetrahedrons to the metrics of the structured elements itself generates an only small overhead of about 2.5% in terms of points and about 12% in terms of tetrahedrons.

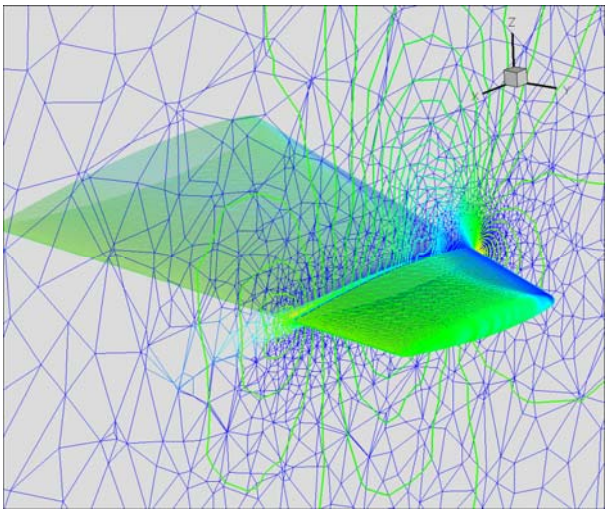
CFD calculations have been performed using the DLR TAU code [19], which is an unstructured finite volume RANS solver second order in time and space. The unsteady equations are integrated in time by an explicit Runge-Kutta time stepping until a steady state is reached. Convergence is accelerated by applying multigrid and local time stepping. For turbulence modeling the one-equation model of Spalart and Allmaras is used. Calculations have been performed for transonic conditions $M$=0.84 at a Reynolds number of $Re$=11.7x10$^6$, at two distinct angles of attack of the experimental data base, $\alpha$=3° and $\alpha$=6°. The results of these com-

putations have been reported in detail in [7], so here for emphasizing the benefit of mixed meshes Figure 14 shows details of the flow fields for the lower angle of attack, where the differences in the surface pressure distributions are less severe. The figure shows the computed flow fields by pressure contours on the wing, and local Mach number and total pressure loss contours in a planar slice through the mesh. The structured elements in the C-type grid much better capture the wake than the O-type structure obtained by hybrid meshing. The O-type grid diffuses the wake in short distance after the wing surface, while for the C-type grids the wake is resolved up to the end of the structured grid area. The deficiency using isotropic tetrahedral meshing in conjunction with stretched hexahedrons is a result of the supersonic area leaving the region of structured cells and by this being immediately diffused or dissipated.



a) mixed anisotropic mesh b) mixed isotropic mesh

Figure 14: Comparison of the computed flow fields around the ONERA M6 wing at an angle of attack $\alpha$=3°; contours on surface: pressure; contour lines on planar slice: local Mach number; mesh in slice plane colored by total pressure loss.



c) hybrid mesh

Figure 14 (cont.): Comparison of the computed flow fields around the ONERA M6 wing at an angle of attack $\alpha$=3°; contours on surface: pressure; contour lines on planar slice: local Mach number; mesh in slice plane colored by total pressure loss.

### 6.3 Boeing CRM model (AIAA DPW4)

The last meshing example is a mesh around the Boeing CRM model [20] used for the 4th AIAA Drag Prediction workshop. The CRM configuration is a wing-body-tail configuration for cruise flight conditions. The boundary layer regions on the wing and the tail are covered by structured C-type grids, the viscous regions at the fuselage and the sing and tail tips are resolved by prismatic layers. The wakes of the wing and the tail plane have been resolved with structured hexahedral meshes up to a half local wing chord downstream the trailing edge. The outer tetrahedral mesh was generated with the SIMMETRIX code applying our sparse octree method for the metric space definition. Figure 15 shows the surface mesh and the slice through the volume mesh, where the different types of elements are highlighted using different colours. Again, the good adaption of the tetrahedrons to the highly stretched hexahedral layer is visible. The mesh contains $3.02 \times 10^6$ points, verifying again the big potential of saving points when compared to the about $10 \times 10^6$ to $15 \times 10^6$ points used in the more classical medium sized hybrid grids used for the meshes at the 4th AIAA Drag Prediction Workshop [21].

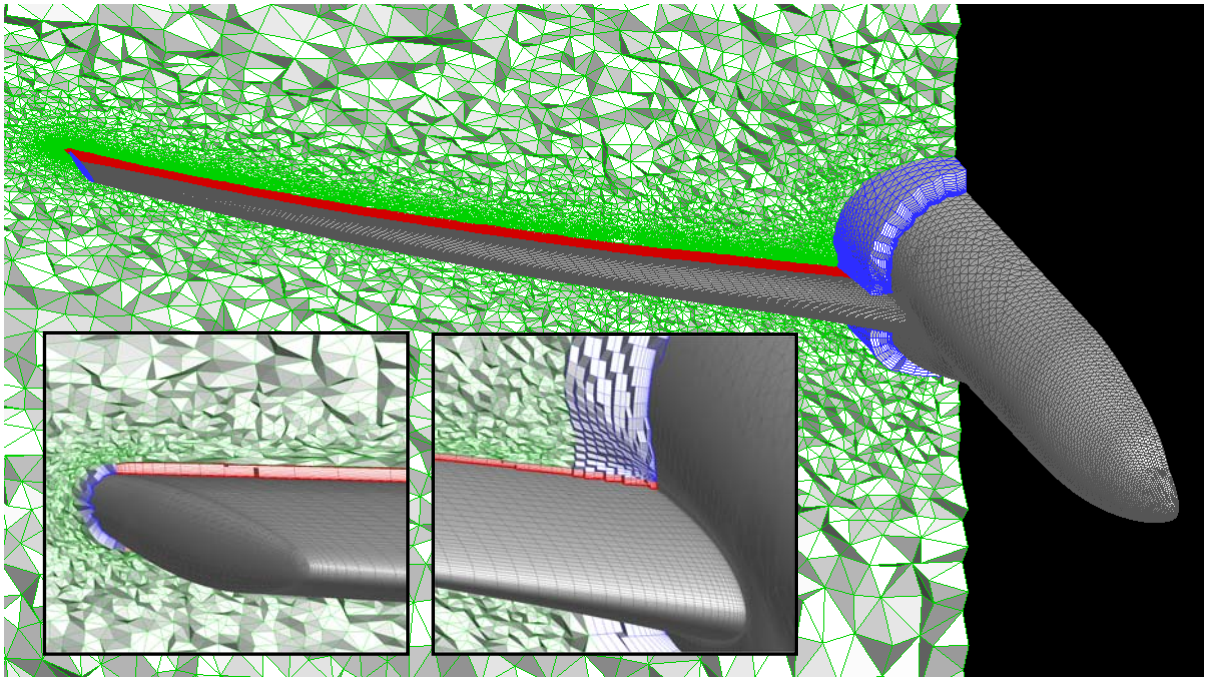


Figure 15: Mixed block-unstructured mesh around the Boeing CRM configuration: surface mesh (grey), hexahedrons (red), prisms (blue), tetrahedrons (green)

CFD simulations have been performed for the common flow condition at $M = 0.85$ and $Re = 5\text{x}106$. The angle of attack was adjusted during the calculations to achieve the target lift coefficient $C_L = 0.5$, resulting in an angle of attack of $\alpha = 2.327°$. Figure 16 shows a view into the computed flow field. The surface is colored by the local pressure, while in three sections through the flow field iso-lines of the local Mach number are displayed. The flow field shows a sharp and smooth orientation of the shock over the wing span. The known double-shock is visible at the outer portion wing. The close-ups show the intersections of the fuselage with the wing (upper-left) and the tail plane (lower), respectively. By the chosen mesh topology enabling structured H-type grids in the intersection, and thereby full resolution of both boundary layers of the adjacent bodies, the well known small separations at the trailing edge of the junctions [21] are resolved. As for the previous calculations of the M6 wing the resolved wakes show the propagation of the velocity deficit up to the end of the structured mesh part.

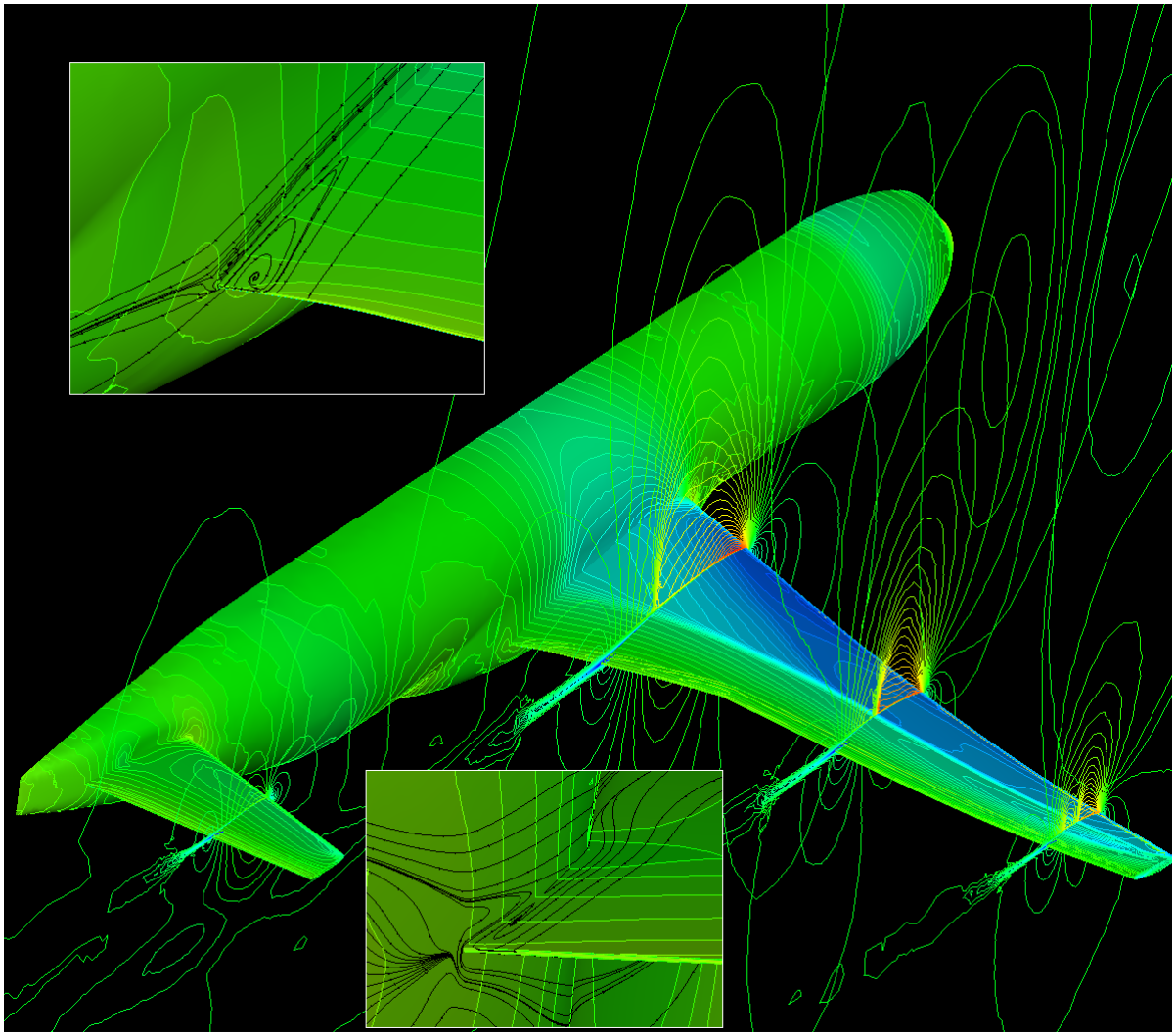
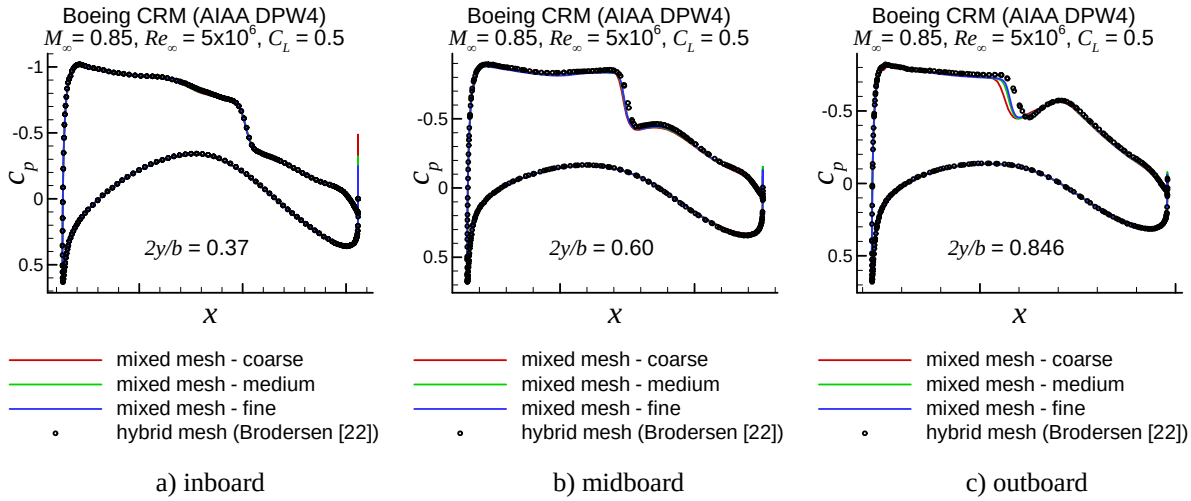


Figure 16: flow field of the CFD simulated flow around the Boeing CRM configuration at $M$ = 0.85, $Re$ = 5x106, $\alpha$ = 2.327° ($C_L$ = 0.5).




a) inboard b) midboard c) outboard

Figure 17: Comparison of pressure distributions of the Boeing CRM configuration at three section cuts between different grid resolutions of the mixed mesh approach and a classical hybrid unstructured mesh

In order to assess the quality of the grids generated by the presented approach a grid refinement study has been performed by generating two additional grids with subsequently refined resolution. The refinement was obtained by adjusting the number of grid points of the structured mesh part by a factor of 1.5 in each direction and a corresponding reduction of the prescribed cell sizes, according to the gridding guidelines for the 4th AIAA Drag Prediction Workshop. The resulting overall point numbers for the two finer meshes are $7.2 \times 10^6$ and $18.8 \times 10^6$. Pressure distributions at three section cuts (Figure 17) show a very high self-similarity of the mesh family and also a good agreement with results obtained on a classical hybrid mesh of $13 \times 10^6$ points [22]. Only the most outboard wing section shows a visible difference of the shock position, which may be improved by an adoption of the surface mesh in this area. Keeping in mind that the targeted point reduction is emphasized for the purpose of design optimization the coarsest level can be seen as highly sufficient to resolve critical flow features in order to get sensitivities that are dominated by physics and not by grid quality.

### 6.4 Memory efficiency of the sparse octree structure

The following table summarizes the sizes of the octree for the first two cases. The memory reported is the size of the octree alone. The memory includes the space needed for storing the data of 3D metric tensors and the corner nodes. Clearly observable is the large potential of memory savings by the sparse octree data structure without loosing any information.

| case | Cube (2752 input points) | | | ONERA M6 (10031 input points) | | |
|---|---|---|---|---|---|---|
| no. of levels | 7 | | | 12 | | |
| octree structure | full | spare box removal | sparse | full | spare box removal | sparse |
| no. of nodes | 49,115 | 10,772 | 7,118 | 2,481,532 | 111,139 | 69,302 |
| no. of boxes | 33,252 | 7,264 | 2,904 | 1,656,096 | 77,536 | 27,950 |
| memory | 6,020KB | 4,392KB | 2,236KB | 210MB | 47MB | 10MB |

Table 2: Differences in the sizes of 3D octree structures for two different cases depending on the completeness of the data structure

Table 3 summarizes how the size of the octree scales with a subsequent grid refinement as it was performed on the CRM example. The octree in this case is only given for one of the four tetrahedral blocks generated, while the grid sizes are given for the overall grid. Beside the effect, that the number of tetrahedrons scale less with the refinement ratio than other element types (theoretically: $1.5^3 = 3.375$), it is observed that the size of the octree scales less than linearly with number of bounding points and comparably to the number of finally generated tetrahedrons.

| CRM mesh size | coarse | scaling ↔ | medium | scaling ↔ | fine |
|---|---|---|---|---|---|
| no. of grid points | 3,029,465 | 2.38 | 7,217,962 | 2.61 | 18,852,841 |
| no. of tetrahedrons | 12,860,499 | 2.01 | 25,876,384 | 2.13 | 55,049,698 |
| no. of octree levels | 18 | 1.00 | 18 | 1.11 | 20 |
| no. of octree nodes | 163,902 | 2.20 | 359,853 | 2.26 | 812,066 |
| no. of octree boxes | 65,729 | 2.21 | 145,257 | 2.26 | 328,301 |
| memory (MB) | 19.8 | 2.20 | 43.5 | 2.26 | 98.2 |

Table 3: Scaling of the size of the sparse octree structure for a subsequent mesh refinement of the mesh of the Boeing CRM configuration

## 7 CONCLUSION

This paper presented a method for anisotropic tetrahedron generation conforming to a metric space derived from surrounding elements for a smooth block-unstructured mixed meshing. It was shown that the block-unstructured mixed meshing is a valuable approach to significantly reduce grid sizes for the aerodynamic analysis of aircraft configurations, with a potential of saving 80-90% of the points used in state-of-the-art hybrid grind generation. Even the introduction of anisotropic tetrahedrons does not compromise significantly this achievement. Additionally, the ability to generate C-type grids achieves an additional gain in accuracy for viscous flow computations by resolving the velocity deficit in wake regions. An additional memory and time efficiency was obtained by implementing a new sparse octree for anisotropic metric size propagation throughout the meshing domain. The memory requirement was reduced up to 95% compared to a full octree structure and still an 80% reduction was obtained compared to an already memory optimized variant. Time efficiency was obtained by using a modified non-interlaced Morton encoding scheme, allowing the conversion of most of the octree operations from floating point to bit arithmetic.

## ACKNOWLEDGEMENTS

The Author would like to thank Carl Olivier-Gooch and Cord Rossow, Cord for enabling the cooperation with the University of British Columbia, Vancouver, Canada, and the corresponding guest stay during which most of the presented work was performed, and Carl for his guidance during the work on volume triangulation methods and their extension to anisotropic meshing. A last thank is towards Joe Walsh and the support group of SIMMETRIX in order to adapt the MeshSim code towards the very hard requirements for our type of meshes.

## REFERENCES

[1] D. Levy, T. Zickuhr, J. Vassberg, S. Agrawal, R. Wahls, S. Pirzadeh, M. Hemsch: Data summary from the first AIAA computational fluid dynamics drag prediction workshop. *Journal of Aircraft* 40(5) (2003), 875-882.

[2] O. Brodersen, M. Rakowitz, S. Amant, P. Larrieu, D. Destarac, M. Sutcliffe: Airbus, ONERA and DLR results from the 2nd AIAA drag pre-diction workshop. *Journal of Aircraft* 42(4) (2005), 932-940.

[3] R. Rudnik, R. Heinrich, B. Eisfeld, T. Schwarz: DLR contributions to code validation activities within the European high lift project EUROLIFT. In Breitsamer et al. (eds.): New Results in Numerical and Experimental Fluid Mechanics IV, *Notes on Numerical Fluid Mechanics* 87, Springer, Berlin Heidelberg, 2004, 42-49.

[4] J. Wild: Acceleration of Aerodynamic Optimization Based on RANS-Equations by Using Semi-Structured Grids. In K.C. Giannakoglou, W. Haase (eds.): *ERCOFTAC Design Optimization: Methods & Applications,* conference proceedings, CD-ROM (2004).

[5] J. Wild , P. Niederdrenk, T. Gerhold: Marching Generation of Smooth Structured and Hybrid Meshes Based on Metric Identity. In B.W. Hanks (ed.): *Proceedings of the 14th Int. Meshing Roundtable,* Springer, Berlin, 2005, 109-128.

[6] J. Wild: Application Of Smooth Mixed Meshes Based On Metric Identity In Aerospace Analysis And Design. In R.V. Grimella (ed.): *Proceedings of the 17th Int. Meshing Roundtable,* Springer, Berlin Heidelberg, 2008, 387-398.

[7] J. Wild: Smooth Mixed Meshes for Acceleration of RANS CFD in Aircraft Analysis and Design. *48th AIAA Aerospace Science Meeting and Exhibit*, AIAA-2011-1267 (2011).

[8] J. Hoschek, D. Lasser: *Grundlagen der geometrischen Datenverarbeitung*. 2nd Edition, Teubner, Stuttgart, 1992.

[9] W.N. Dawes, S.A. Harvey, S. Fellows, N. Eccles, D. Jaeggi, W.P. Kellar: A practical demonstration of scalable, parallel mesh generation. *47th AIAA Aerospace Meeting and Exhibit*, AIAA 2009-981 (2009).

[10] F. Alauzet: Size gradation and control of anisotropic meshes. *Finite Elements in Analysis and Design* 46 (2010), 181-202.

[11] C. Dobrzinsky, P. Frey: Anisotropic Delaunay mesh adaptation for unsteady simulations. In R.V. Grimella (ed.) *Proceedings of the 17th International Meshing Round Table*, Springer, Berlin Heidelberg, 2008, 177-194.

[12] S. McKenzie, J. Dompierre, A. Turcotte, E. Meng: On metric tensor representation, intersection and union. *Proceedings of the 11th ISSG conference*, Montreal, (2009).

[13] F. Labelle, J.R. Shewchuk: Anisotropic Voronoi Diagrams and Guaranteed-Quality Anisotropic Mesh Generation. *Proceedings of the Nineteenth Annual Symposium on Computational geometry SCG '03*, ACM, New York, NY, ,2003, 191-200.

[14] J.R. Shewchuk: Robust Adaptive Floating-Point Geometric Predicates. *Proceedings of the Twelfth Annual Symposium on Computational Geometry, SCG '96*. ACM, New York, NY, 1996, 141-150.

[15] Simmetrix Inc., http://www.simmetric.com (accessed 2010)

[16] O. Brodersen, M. Hepperle, A. Ronzheimer, C.-C. Rossow, B. Schöning: The Parametric Grid Generation System MegaCads. Proc. *5th Intern. Conf. On Numerical Grid Generation in Comp. Field Simulation*, National Science Foundation (NSF), 1996, 353-362, http://www.megacads.dlr.de.

[17] V. Schmidt, F. Charpin: Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers. *Experimental data base for computer program assessment*, AGARD AR-138, B1 (1979).

[18] CentaurSoft, http://www.centaursoft.com (accessed 2010).

[19] D. Schwamborn, T. Gerhold, R. Heinrich: The DLR TAU-Code: Recent Applications in Research and Industry. *ECCOMAS CFD 2006 CONFERENCE*, Netherlands (2006).

[20] J.V. Vassberg, M.A. DeHaan, S.M. Rivers, R.A. Wahls: Development of a Common Research Model for Applied CFD Validation Studies. *26th AIAA Applied Aerodynamics Conference*, AIAA-2008-6919 (2008).

[21] J. Vassberg, E. Tinoco, M. Mani, B. Rider, Z. Zickuhr, D. Levy, O. Brodersen, B. Eisfeld, S. Crippa, R. Wahls, J. Morrison, D. Mavriplis, M. Murayama: Summary of the Fourth AIAA CFD Drag Prediction Workshop. *28th AIAA Applied Aerodynamics Conference*, AIAA-2010-4547 (2010).

[22] O. Brodersen, S. Crippa, B. Eisfeld, S. Keye, S. Geisbauer: DLR Results from the Fourth AIAA CFD Drag Prediction Workshop. *28th AIAA Applied Aerodynamics Conferenc*e, AIAA 2010-4223 (2010).