Computational Fluid Dynamics 101

Module 01 - Meshing Tips & Techniques

Formule ETS ; Aerodynamics Department

Version:	v0.1
Date:	September 21, 2025
Authors:	Hugues Perrin
Confidentiality:	Public Teaching Resource

Summary

1 Introduction

This guide aims to teach, understand and apply conventionally good meshing practices for both general aerodynamics CFD and motorsport-oriented CFD. This guide is part of the onboarding program for the aspiring aerodynamicists of Formule ETS.

Caution: There is no such thing as absolute truth. Double-check every unclear information of this guide

1.1 What is meshing ?

Meshing is the process of turning a continuous fluid domain (your geometry + the surrounding flow region) into a finite set of small sub-regions (cells or elements) over which the Navier–Stokes equations are approximated and solved numerically. Meshing converts PDEs into a very large system of algebraic equations by integrating fluxes across faces and storing unknowns at nodes or cell centers. The mesh's shape, size, and quality directly control numerical accuracy, stability, and cost. Meshing is a **discretization process**.

Mesh quality is crucial to achieve good convergence and stability of the solution. There are a few metrics to help determine the quality of a mesh that will be detailled further down the guide.

Note: A clean geometry is key for a good mesh. A good mesh is key for a reliable solution.

1.2 How to mesh ?

For most commercial softwares, meshing is a built-in user friendly function. It is generally tied to a sense of automation ("Automated Mesh" in StarCCM+) that eases the painful process of meshing, enabling quicker workflows. Mesh functions generally have a set of parameters to play with to achieve a good mesh quality and a more precise solution.

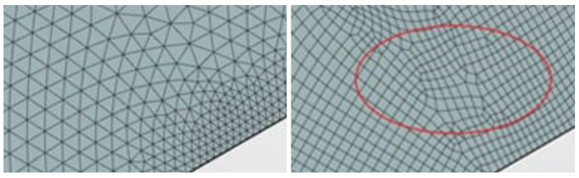

Figure 1: Good/Bad mesh comparison

Generally, it is possible to modify mesh parameters locally. This allows for the use of a different mesh strategies for harder-to-mesh geometries without exponentially increasing the rest of the mesh. This is called **refinement**.

Proper refinement is crucial to achieve precise results for high shear flows and aggressive field gradients. Usual refinement locations are around rotating objects (wheels, blades, fans), thin objects (wing trailing edges, thin plates) and wake generated by the object. Refining the wake of the objects allow for a more precise field calculation and overall better results.

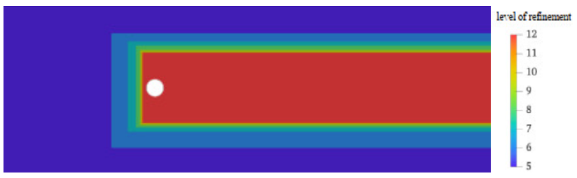

Figure 2: Wake Refinement Example

1.3 What is a good mesh ?

There is no perfect recipe to create a mesh. Each mesh has to be tailored to the input geometry and adapted to the type of study that is happening. Cell count of a mesh is dependent on refinement but also available time and computing power. A perfect mesh is a mesh that lays the foundation for CFD that correlates to experimental data with the lowest cell count possible.

Note: A general, symmetry based, straight line simulation of a FSAE should count 15-30M cells.

2 Types of Mesh

Meshes come in different flavor depending on how cells interact with each other, and depending on how they look like. This section will provide information on the different types of meshes, their advantages, and their drawbacks.

2.1 About Mesh structure

A mesh is basically two things packed up in a multi GigaOctet file:
1. A list of all the vertices(points) of the mesh, and their x,y(,z) coordinates
2. A description of which connection to other vertices a vertex has

We can then distinguish 2 distinct type of mesh structure depending on how mesh is generated, and how information are stored within the mesh file: structured and unstructured meshes.

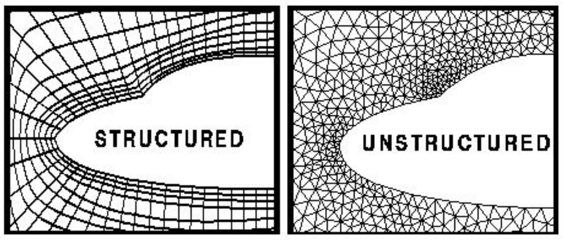

Figure 3: Structured vs Unstructured meshes

2.1.1 Structured Meshes

In the case of structured meshes, the mesh is fully following the body which can sometimes lead to a sub-optimal or degenerate mesh for complex bodies. The mesh becomes it's own referencing grid where a cell's neighbors are simply $(i \pm 1, j, k), (i, j \pm 1, k), (i, j, k \pm 1)$: the connectivity between each cell is **implicit**. The mesh geometry may be cartesian (perfect squares/triangles) or curvilinear for a better body mapping. Complex shapes might need higher-level strategies such as overset meshes, or multi-block meshes. Structured mesh finds its use in computation time optimization and simple geometries meshing. It is also very practical for low-level computation such as Euler equation models, or inviscid/laminar RANS.

2.1.2 Untructured Meshes

As for unstructured meshes, the connectivity is **explicit** and requires more space to store the graphs that detail which cells are sharing a face and what face/nodes compose a unique cell. Although presenting a higher volume, it allows for exponentially more versatile meshes, with very straightforward refinement possibilities. The versatility of unstructured meshes allow for very various element types such as tetrahedrons, pyramids, cubes, hexahedrons etc. It is also possible to use multiple element types to mesh different parts of the domain, a common example is using a curvilinear cube approach for boundary-layers and switching to tetrahedrons outside of the BL.

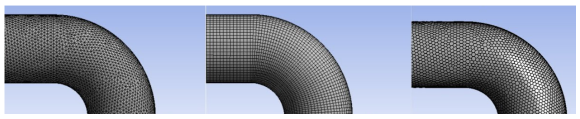

Figure 4: Different cell types for unstr. meshes

2.2 Advantages and Drawbacks

Each of these structures has its advantages and drawbacks. It's an engineer's job to determine which structure should be applied to his problem. Below is a table that compares both structures on different aspects.

Aspect	Structured	Unstructured
Geometry Fit	-	++
Near-Wall	-	++
Accuracy	=	+
Quality	=	+
Performance	++	=
Adaptivity	-	+
Complexity	++	+
Storage	++	−

2.3 Typical use cases

Structured meshes are mostly used for power-greedy models (DNS, LES) on simple internal geometry flows, or smooth external flows with curvilinear blocks, lattice-Boltzmann and finite-difference solvers.

Unstructured meshes are preferred for complex geometries and faster simulation preparation. Its adaptability permits very fast growing meshes from small refined cells, creating more optimized meshes at the cost of an increased file size which can be an industry concern. Unstructured meshes are king for most of the industrial applications due to its ease of use.

Note: For FSAE CFD, unstructured meshes are preferred due to very complex geometry and refinement

2.4 Surface Meshes

When exporting CAD parts from your software and importing them in your CFD software, it is generally already considered as meshed. Unfortunately, this mesh is generally not adapted to generate a volume mesh from due to very odd element shapes.

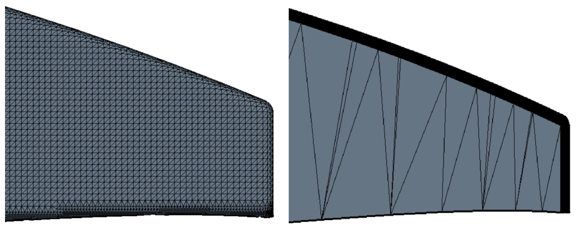

Figure 5: Remeshed Surface vs Intial CAD Mesh

Most commercial software will use this surface mesh to create a volume mesh that is perfectly adapted to the part. It is then **critical** to have a good surface mesh to obtain a good volume mesh.

In StarCCM+, the Automated Mesh proposes a ”Surface Remesher” option. It is also possible to use a ”Surface Wrapper” to repair and render your cad air-tight.

Caution: Do NOT remesh surface to infinitesimal sizes, it will exponentially increase your sim/mesh file's size.

3 Mesh Generation

There are different ways of generating a mesh for CFD applications. This section aims to provide guidelines on how a mesh is generated and how to control it.

3.1 Mesh Generation Algorithms

There are two general main ways of generating a mesh :

1. Extruding a surace mesh into a volume mesh, this ensure a surgical respect of the initial surface mesh but is more computationally expensive because of the increased number of surface mesh elements.

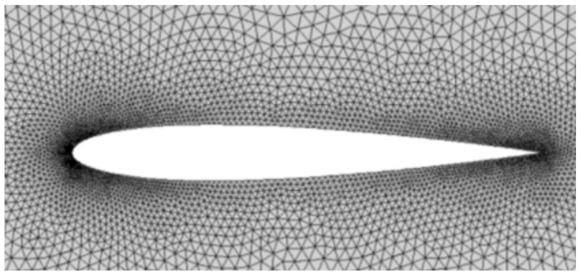

Figure 6: Extruded Tetrahedral Mesh

2. Creating a volume mesh before intersecting the mesh with the studied object. This is faster, but might oversimplify the geometry if refinements aren't correctly set or if the parameters are too soft. A variant is a trimmed cell mesher that will reduce its elements' sizes each time it intersects the surface up to a defined threshold.

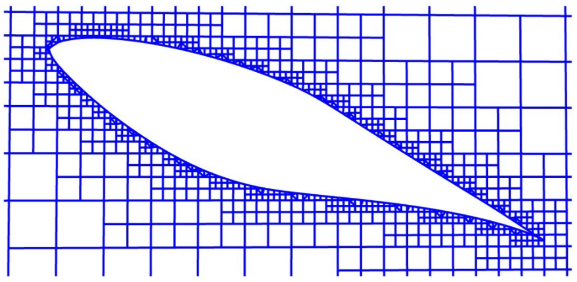

Figure 7: Trimmed Cell Mesh

Both of these methods are reliably used across industries with different mesh geometries (tet, cube, poly, etc.). The author recommends trimmed-cell meshers if available, and polyhedral surface extrusion for a more organic rendering of the shapes, lighter mesh files and increased quality of wake flows.

3.2 General Mesh Parameters

This section will present basic use and parameters for the meshing operations, whether they're surface meshing or volume meshing. Particular cases (boundary layer, wake) are subject to dedicated sections. All parameters aren't covered.

Warning: This section is about StarCCM+ in particular. Parameters and appellations may differ for other softwares.

3.2.1 Surface Wrappers

Following are the existing Surface Wrapper options. Each of them enables different parameters that can be changed.

Mesher Execution Mode (Serial, Parallel, Concurrent) : determines how the meshing operation is distributed or not on multiple CPUs. Impacts wall time (total real meshing time) and determinism (repeatability) of the mesh.
Curvature Refinement : enables curvature-based surface sizing that improves feature capture on curved surfaces. Locally increases the cell count to match curvature deviation.
Proximity Refinement : adds refinement in narrow gaps/contacts to prevent for leakages in small clearance zones. Locally increases the cell count to refine the features.
Gap Closure : Seals leaks below a user-defined threshold to stabilize wrap on imperfect CAD. May remove small features in the process.
Mesh Alignment : Aligns wrapper mesh to a reference location to help matching across parts. Has minor impact on cell-count but can help meshing periodicity.

The following lines are the parameters that control how the mesh is generated. Some of them are activated by enabling different controls of the above list.

Base Size : Sets global size scale. Other sizes can be absolute or a defined percentage of the base size.
Target/Minimum Surface Size : drives overall element density and how tight the refinement can be to match initial part. Too high will create poor geometry, too low risks high wall time.
Surface Growth Rate : Defines how smoothly sizes change across the surface.
Surface Curvature : Refines until chords resolve capture. More points or smaller deviation increases precision and cell count.
Surface Proximity : Refines to maintain a defined number of cells within a gap to prevent bridging or poor mesh between elements.
Edge Proximity : Safeguard for sharp edges. Lower feature angle preserves more edges. Adds near-edge refinement.
Gap Closure : Defines what gets sealed before wrapping. Can intentionally remove pores/hole that would impact mesh quality.
Smallest Disconnected Surface : Deletes tiny islands/tiny internal volumes below thresholds to prevent junk cells.

3.2.2 Automated Mesh

Following are the existing Automated Mesh operation-level properties. Each of them enables different parameters that can be changed.

Per-Part Meshing : mesh each part separately (non-conformal) vs all parts together (conformal). Helps when interfaces are problematic; changes interface treatment and parallelization.
Mesher Execution Mode (Serial, Concurrent, Parallel) : determines how the meshing operation is split between CPUs and how each CPU meshes.

Caution: Parameters (aside of PRISM layers, see "Boundary Layer Meshing" section, have the same name as Surface Wrapper parameters. See the Surface Wrapper parameter list for reference.

3.2.3 Recommended Sizes

Below is a list of recommended sizes for different FSAE-tied use cases. They are NOT mandatory and do NOT represent a promise of good results. Those size have to be tailored to each user geometry, they only provide a starting base. They're also tied to a note that gives information on how the mesh quality can be increased locally.

Feature	Size[mm]	Notes
Body panels	6-20	Curv. Ref.
Floor	5-10	Prox. Ref.
Undertray	3-10	–
Diffuser	3-10	Vol. Growth
Wings LE	1-5	Curv. Ref.
Wing TE	0.5-3	Edge Ref.
Gurneys	0.1-0.5	Edge Ref.
Endplates	3-20	–
Flap Gaps	0.5-3	Gap Clos.
Suspensions	0.5-3	Curv. Prox.
Tyres	2-8	Contact Patch

3.2.4 Quick summary

The parameters can be separated into 4 groups depending on what they do and/or impact:

The **Size Parameters** mechanically control the size of the element. They help refine and repair the geometry. They also allow for easy feature management for geometry simplification.

The **Growth Parameters** control how the size increases or decreases from a size threshold. Smoother transitions help with quality to the detriment of cell count.

The **Feature Parameters** allows for a precise spending of cell count to preserve or delete features that could impact mesh quality.

The **Execution Parameters** offers options to optimize run time, repeatability, robustness and determinism.

4 Boundary Layer Refinement

In viscous aerodynamics, the fluid/wall interaction is a key aspect that may single-handledly ruin calculations if poorly managed. When a fluid interacts with a solid, a **boundary layer** forms as the presence of the solid reduces the speed of the neighboring flow. The boundary layer is the part of the fluid that is directly affected by the presence of a wall.

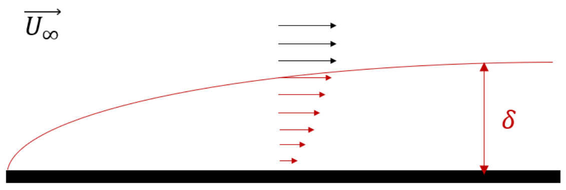

Figure 8: Boundary Layer Representation

The boundary layer region is a very high gradient region, meaning it controls a disproportionately large part of the bounding aerodynamics (drag, lift, flow attachment, stall prediction). It is essential to model the boundary layer correctly as to not drastically and negatively impact the solution.

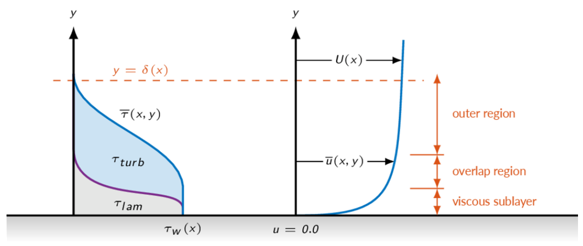

Figure 9: Gradients within the BL

It is possible to either directly calculate the boundary layer, or model it with the use of wall functions. This choice is driven by the calculation of dimensionless coefficients, like U^+ or y^+.

$$y^+ = \frac{y_P \cdot u_\tau}{\nu} \qquad U^+ = \frac{u}{u_\tau}$$

Those two coefficients indicate where you're positioned within the boundary layer. For example, for $y^+ = U^+ = 1$, you are calculating in the viscous sub-layer. Those coefficients are generally used to prove mesh-solution quality, and to dimension the mesh in the first place.

When it comes to mesh generation, y^+ is the fundamental metric. It is used to calculate how far from the surface the first layer of the mesh should be depending on what you're trying to do, namely calculate or model the flows within the boundary layer.

The phenomena happening inside a boundary-layer are very complex, and involve multiple sub-layers with their respective approximation equations. The 0 to U_{infty} gradient implies the existence of a laminar, viscous, sub-layer where y^+ is roughly equivalent to U^+ for a very low Reynolds, before transitioning to a turbulent flow.

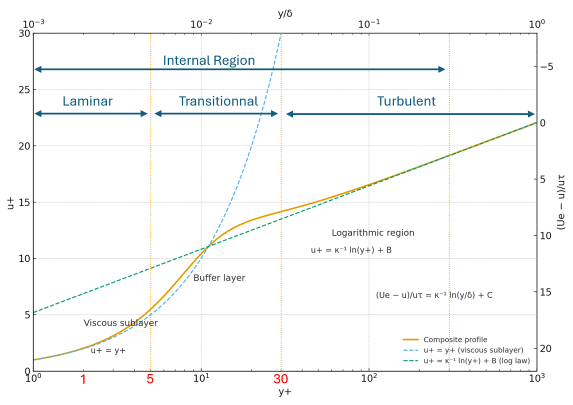

Figure 10: Sub-regions of a boundary layer

To precisely capture all those effects is to give a sufficient amounts of points to reduce imprecision errors. This is generally done through a very strong refinement in the area neighboring walls of the studied object.

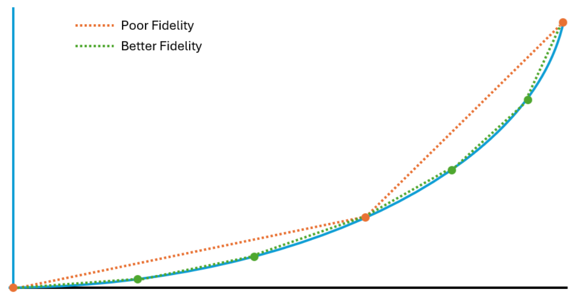

Figure 11: High vs Low Fidelity Modeling

The strength of this refinement is dictated by the user choice of calculating or modeling the boundary layer, which also depends on available time and computing power. This refinement is generally achieved by the use of a PRISM mesher.

4.1 PRISM Layers Refinement

A PRISM-Layers mesher is an algorithm that aims to increase the number of cells close to walls. It does so to increase the fidelity of the modeling of the boundary layer and improve the reliability of the simulation and its results. It is based on the use of the aforementioned dimensionless coefficients which we can input in a Y+ calculator (widely available on the internet) to export the necessary values for the PRISM mesher. Meshes including PRISM layers are considered hybrid meshes.

In the case of StarCCM+ (and strangely), the height of the first layer Δy is not an input, but rather an implicit calculation:

$$\Delta y \cdot G^N = \delta \qquad <=> \qquad \Delta y = \frac{\delta}{G^N}$$

where G is a user-defined growth factor, N is a user-defined number of layers and δ is the total boundary layer height.

Note: It is widely recognized that N should be above 16 for boundary layer calculation, and G under 1.3. Finding a good combination is key.

We can then merge this equation with the y^+ equation to obtain the equation used for first layer height calculation:

$$\Delta y = \frac{y^+ \cdot \nu}{u_\tau} = \frac{\delta}{G^N}$$

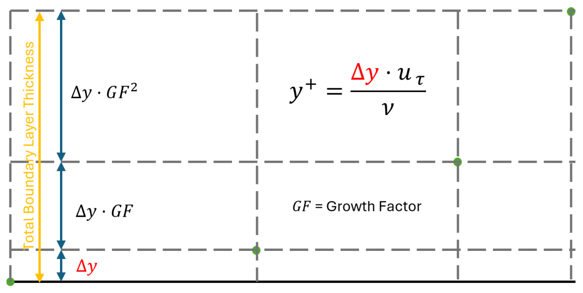

Figure 12: PRISM Layers definition

4.2 Calculating the sub-layers ($y^+ < 1$)

Very close to the surface, the velocity gradient (the slope of the velocity form) is very steep and needs a lot of mesh layers to properly model the boundary layer. This means a drastic increase in cell count along the necessity of using an adapted model and set of closure equations.

Warning: Using unadapted models will produce poor results, even with $y^+ << 0$

The most common models used for boundary layer resolution are RANS $k-\omega$ (with SST transition or not), RANS RSM, RANS $\gamma-Re_\theta-SST$ and some a few $k-\epsilon$ variants. It is also possible to mix LES and Unsteady RANS for high-engineering cases.

In the FSAE context, Δy can drop as low as 1E-6 m to keep y^+ expectations. This causes a cell count crisis that will impact drastically computation time and, in some case, simulation robustness. It is crucial to study where you need precise details on the sub-layers behavior and where you can use simpler modeling techniques (see 4.3).

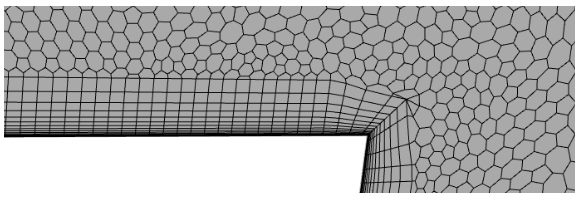

Figure 13: PRISM Layer Refinement

4.3 Modeling the sub-layers ($y^+ > 30$)

Above $y^+ = 30$ the calculations take place in the logarithmic turbulent part of the boundary layer. Integrating the logarithmic functions of this sub-layer up to the wall can produce unreliable and downright false results.

In an effort to cut computation time, Wall Functions (often called Wall Treatments) were created to approximate the effect of the lower sub-layers on the flow. Those approximations are efficient to an extent, but start losing accuracy for wall-separating flows, high shear stresses and high flux heat transfer.

However, the wall treatment approach is cost effective, and can lay a strong aerodynamic foundation. It is also as efficient as low y^+ methods for simple flows.

Note: CFD is about optimizing cost vs accuracy. Simpler method shouldn't be forgotten, even less in high-pressure, fast development cycles.

4.4 Coffin corner ($1 < y^+ < 30$)

The coffin corner zone of the boundary layer, actually called the buffer sub-layer in technical terms, is the zone of transition between a laminar dominant sub-layer and a turbulent sub-layer.

As such, it is difficult to quantify the turbulence of the flow in this zone, although some equations exist like the Van Driest equation. However, experimental data as shown a rather high deviation from existing models. It is highly discouraged to place the first layer of the mesh in this zone. The computation of the effects within this zone gain in accuracy with the number of points in the two other sub-layers.

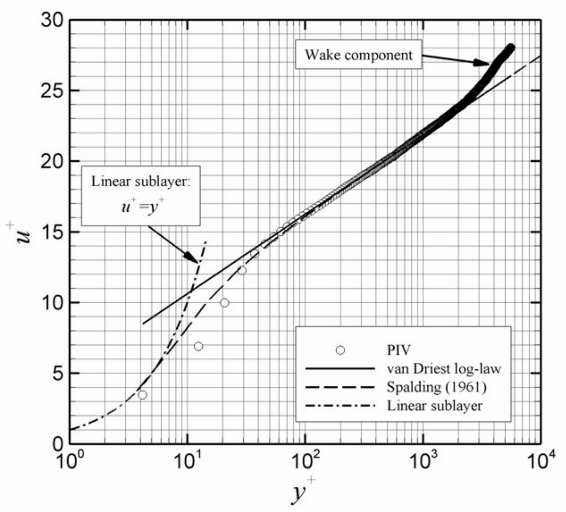

Figure 14: Experimental Data - Humble

4.5 General Recommendations

Define $y^+ < 1$ for CFD parts that are expected to present high shear, strong gradients and boundary layer separation. For FSAE, this could include wings, wheels, suspension, chassis tubes and driver. There should be enough layers between any floor/venturi tunnels and moving ground.

Keep $y^+ > 30$ for simple shapes that aren't too long and for which you do not expect high force generation. This may include endplates, supports or body panels.

5 Mesh Quality Metrics

To measure the quality of a mesh, there's a few metrics that deserve close monitoring. Poor meshes are the principal cause for unstable solutions, high residuals, low accuracy to experimental validation data and ultimately sky-rocketing residuals leading to divergence.

5.1 Non-orthogonality

Non-orthogonality measures the angle between a face's normal and the vector joining the two adjacent cell centers; large angles make fluxes and gradients inaccurate and force heavy non-orthogonal corrections.

It should be kept below about 70 degrees for most cells, and becomes critical when over 85 degrees which will lead to degraded convergence and noisy solutions.

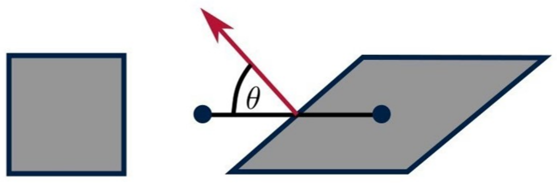

Figure 15: Cell Non-orthogonality

If non-orthogonality is too high, realigning the cell with extra transition layers and reducing growth rates. If available, run post-mesh optimization to repair bad elements.

5.2 Skewness

Skewness quantifies how far a cell/face deviates from an ideal, orthogonal/equiangular shape. High skewness creates negative interpolation weights, excess numerical diffusion, and stability issues leading to local divergence.

Skewness should stay below a 0.5 threshold for the majority of cells. Physically flawed oscillations start appearing along residuals stall when a noticeable fraction gets above 0.85.

Increasing curvature and proximity meshing parameters can help reduce high skewness. Increasing the number of cells and aligning them on bounding walls can also drastically reduce skewness.

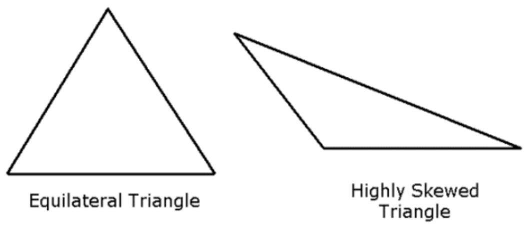

Figure 16: Element Skewness

5.3 Aspect Ratio

Aspect Ratio is the longest over shortest cell dimension. High AR is acceptable only when aligned with the dominant gradient (e.g., prism layers that are normal to a wall). Otherwise, it amplifies cross-diffusion and conditioning problems leading to poor solution stability or divergence.

Core (non-PRISM) mesh shouldn't exceed an AR of 10 to 20. PRISM-layer elements can legitimately reach AR upwards of 100-1000 before starting to cause issues. Widespread AR of over 50 becomes critical to the mesh's integrity.

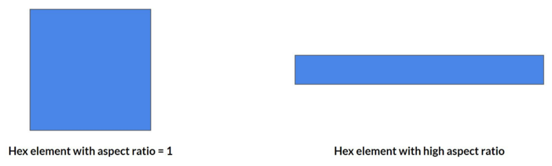
Hex element with aspect ratio = 1
Hex element with high aspect ratio

Figure 17: Element Aspect Ratio

Lowering growth rates and adding intermediate transition layers can help correct critical AR cells. Refining in the cross-gradient direction is also strongly recommended for near wall core mesh.

5.4 Jacobian

The Jacobian determinant measures element inversion and distortion. The scaled Jacobian (-1 to 1) folds size out to leave pure shape quality. Jacobians should always have a positive value.

Caution: Negative Jacobian values will likely abort the mesh operation and display a message error.

Jacobian metrics become critical to the mesh when falling under 0.2 and will likely force solver errors, divergence and invalid mesh errors.

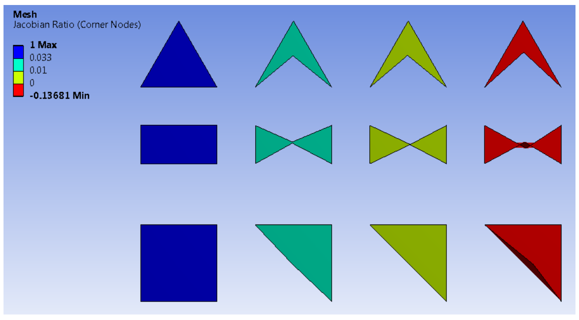

Figure 18: Element Jacobian Examples

A good way of solving Jacobian issues is to manually repair invalid cells that didn't spark an error message. To avoid crashes, reducing curvature deviation distance and lowering growth rates generally help create a valid mesh.

5.5 Metrics Analysis

For every generated mesh there should be a complete diagnostics to determine mesh quality before running a CFD computation. These diagnostics should include:

- Generate global stats : maximum, minimum, average, percentiles for each of the aforementioned metrics.

- Map the lowest percentiles elements for each metrics to localize and fix problematic elements.

- Map y^+ surface values for boundary layer meshing verification and buffer layer avoidance.

Note: All targets are met, no poor quality clusters: mesh is ready for CFD.

Caution: Small poor quality clusters: fix errors, or accept and monitor them.

Warning: Lots of clusters, thresholds not met for multiple metrics on a non-negligible number of cells: NO-GO, fix the mesh

5.6 Mesh Independence

When discretizing the domain in finite volumes, an infinitely precise solution can be achieved with infinitely small elements. However, as computation cost grows exponentially and elements get smaller, there comes a point where the solution doesn't change beyond computer precision and oscillation reduction. The goal, like mentioned in the intro, is to find the smallest mesh that achieves mesh independence.

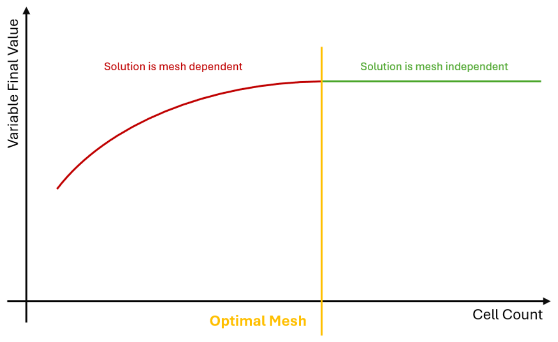

Figure 19: Mesh Independence Example

Performing a mesh independence study is crucial to ensure repeatability and determinism of the solution.

Note: It is common practice to use a slightly finer than optimized mesh for production.