Mesh Adaptation that Works!

Paul Galpin (ISimQ) & Nick Wyman (Pointwise)

1 CFD Simulation Quality

You may already be using CFD in your product design process, or you may be interested in enhancing your product or process by CFD simulation technology. Either way, you require high-quality CFD simulations to make reliable decisions that benefit your product. While it is easy to obtain a CFD simulation result these days using commercial CFD software, the quality of a given CFD simulation remains squarely on the shoulders of the CFD analyst.

ISimQ measures CFD simulation quality in terms of numerical error, model error and systematic error. Model errors originate from physical models such as turbulence, heat transfer, phase change, and chemical reactions. Systematic errors relate to differences between the true and the computer representation of a device, such as the level of geometric detail and boundary conditions. Numerical errors refer to the discrete CFD solution of the governing conservation equations. They consist of discretisation errors, iteration errors and precision errors. In contrast to model and systematic errors, CFD analysts are in direct control of numerical errors.

Discretisation errors are mesh-related. In real estate sales, they say that there are three essential considerations: location, location and location. Likewise, in CFD simulations, the three most important factors impacting solution quality are meshing, meshing and meshing. A mesh spacing that does not adequately resolve local variations in the flow variables introduces discretisation error, i.e. the flow equations are not accurately solved. On the other hand, if the mesh is overly refined, the computational time and effort is needlessly increased, as calculation times are proportional to N^a, where N is the number of grid nodes, and the exponent a is usually larger than unity. Mesh element types and data structures also impact the person-effort required to generate a mesh, and the cost per unit of accuracy. Mesh types range from structured hexahedra to unstructured polyhedra and from manual to fully automated meshing procedures. A well-made hexahedral mesh is generally more accurate per unit of computational effort compared to an automatically generated unstructured mesh but requires more human time and meshing skills to attain. If that is not enough, the mesh quality also has a massive impact on the robustness of CFD simulations. In a perfect world, a CFD mesh has elements with 90° between adjacent edges, a volume expansion ratio approaching unity, and a low mesh aspect ratio. In practice, all meshes fall short of "perfection". Therefore, all you who generate CFD meshes, the responsibility for computational robustness, accuracy and efficiency depends on you. Or does it not?

2 Mesh Adaptation Goals and Challenges

For the last three decades, mesh adaptation procedures have been available. With mesh adaptation, a CFD simulation begins on an initial mesh, and the CFD simulation procedure itself improves the mesh to reduce the discretisation error for the flow at hand. In a first step, the adaptation algorithms estimate truncation errors, for example, by examining local gradients of the flow variables. Then, they enrich the mesh in the areas of the highest gradients, hoping to reduce discretisation errors and to determine the "ideal" mesh for the simulation problem. Adaptation sounds impressive. It is available in most commercial CFD packages. So why are we not all using adaptation in our CFD simulations?

As usual, the devil is in the details. The problem is that, strangely, most mesh adaptation procedures negate the key benefits that they are trying to address:

- Adaptation does not resolve the correct geometry. Most adaptation procedures are integral to the CFD solver. Hence, they adapt only to a faceted approximation of the actual geometry. Commonly, the geometry for adaptation is the mesh face representation in the original (initial) mesh. After adaptation, one has the ideal mesh for precisely the wrong geometry.
- Adaptation decreases the mesh quality when locally refining the mesh. Many adaptation procedures use a divide-and-conquer approach to enrich the mesh, whereby an existing mesh element is locally divided into additional elements. While convenient to program, this approach can lead to a steady decrease in mesh quality with refinement, causing a decrease in robustness, longer run times and perhaps even *increased* discretisation error.
- Adaptation in near-wall shear layers where the gradients of the flow variables are extensive has many challenges. Brute force approaches typically use isotropic refinement near walls, causing an explosion in the mesh size. A common strategy to avoid the mesh size explosion employs stretched tetrahedra to resolve the large gradients normal to walls without over-refining parallel to the wall. However, this approach leads to a massive decrease in mesh quality. Researchers are working towards intelligent anisotropic adaptation, but solver robustness on an anisotropic adapted mesh remains a formidable challenge. Frustratingly, do we need a CFD simulation plus an automatic mesh adaptation procedure to tell us what we already know, that the gradients of flow variables are large in boundary layer regions?
- Adaptation procedures often lead to excessive run-times, either because the mesh was over-refined in some direction or location, or the mesh quality decreased during adaptation causing the CFD solver to struggle, or even the simple issue of when to stop the refinement procedure. If the computational effort grows far beyond a "standard" CFD simulation, analysts give up on adaptation and make the best mesh possible given the expected flow field and the human time, computing time and resource constraints.

3 A New Mesh Adaptation Procedure

In a joint effort, Pointwise and ISimQ have developed a new mesh adaptation procedure that addresses the above challenges.

The adaptation procedure separates the meshing and solving steps in a coordinated and automated way managed by an overall adaptation program. It accomplishes several benefits. In the first step, the analyst creates an initial mesh to start the adaptation procedure. This initial mesh should resolve the near-wall boundary layer regions adequately. For example, Pointwise's T-Rex approach automatically creates regions of hexahedral mesh layers with high aspect ratio for computational efficiency, accuracy and robustness reasons in this area. The

initial mesh generation step controls the target near-wall distance, y^+, thereby removing this task from the mesh adaptation procedure.

Discretisation errors, e_h, are the difference between the solution on an infinitely fine grid and a finite-width grid:

$$e_h = \phi - \phi_h$$

They originate from truncation errors, τ_h, defined as:

$$\tau_h = \sum_m a_m \phi_m - \sum_m a_m (\phi_h)_m$$

a_m are entries in the finite-volume coefficient matrix. The index m runs over all the connections of a grid point. For linear problems, discretisation and truncation errors are related by:

$$\tau_h = \sum_m a_m (e_h)_m$$

The equation shows that truncation errors act as source terms for discretisation errors. Hence, to address the problem at its source, truncation errors form the basis for the local mesh adaptation.

The present adaptation algorithm estimates truncation errors along each edge of the mesh as the difference of a linear and cubic interpolation of relevant flow variables. Figure 1 shows a hexahedral mesh element with vertex-based variable storage.

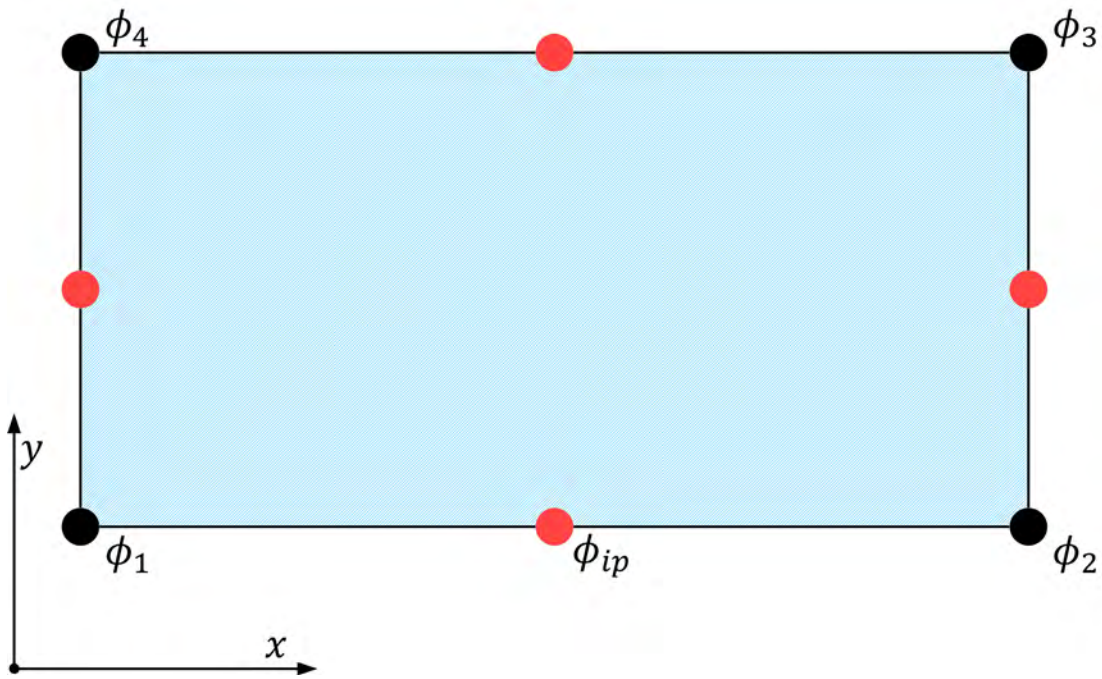

Figure 1: Mesh element & mesh edges

Linear and cubic interpolations to the integration point, "ip", located at the middle of the edge between points "1" and "2" yield:

$$\phi_{ip}^{l} = \frac{1}{2}(\phi_1 + \phi_2)$$

$$\phi_{ip}^{c} = \frac{1}{2}(\phi_1 + \phi_2) + \frac{h}{8}\left[\left(\frac{\partial \phi}{\partial x}\right)_1 - \left(\frac{\partial \phi}{\partial x}\right)_2\right]$$

The truncation error estimate for the edge is, therefore:

$$\tau_{ip} = \frac{h}{8}\left[\left(\frac{\partial \phi}{\partial x}\right)_1 - \left(\frac{\partial \phi}{\partial x}\right)_2\right]$$

In three dimensions, the truncation error estimate is an anisotropic tensor. In the current work, we use a scalar adaptation sensor, S, which is proportional to the tensor truncation error estimate:

$$S \propto h\left|\tilde{h}_j\left(\frac{\partial \phi}{\partial x_j}\right)_1 - \tilde{h}_j\left(\frac{\partial \phi}{\partial x_j}\right)_2\right|$$

Here:

$$h = \sqrt{h_j h_j} \quad \tilde{h}_j = \frac{h_j}{h}$$

It is desirable to avoid overly refining already short mesh edges, e.g. those in the near-wall region or at other discontinuities, during the adaptation procedure. Raising the edge length in the adaptation sensor to an exponent, p, decreases the sensor quickly with a reduction in edge length. The final form of the sensor becomes:

$$S = h^p\left|\tilde{h}_j\left(\frac{\partial \phi}{\partial x_j}\right)_1 - \tilde{h}_j\left(\frac{\partial \phi}{\partial x_j}\right)_2\right|$$

Practical values of the exponent p range between 2.5 and 3.0 for three-dimensional meshes. The smaller the value for p the more uniform and rapid is the refinement procedure, with a trade-off between the quality of the resultant adaptation relative to the rate of adaptation.

After a first CFD simulation on the initial mesh, the adaptation algorithm extracts gradients of critical flow variables such as velocities, Mach numbers or temperatures, and calculates the sensor field. Each vertex of the mesh records the maximum sensor value for all edges common to the vertex. The resulting S-field values are then compared to pre-determined thresholds:

$$S_T = h_T^p\left|\tilde{h}_j\left(\frac{\partial \phi}{\partial x_j}\right)_1 - \tilde{h}_j\left(\frac{\partial \phi}{\partial x_j}\right)_2\right|$$

The goal is to define a target adapted edge scale length, h_T, such that all edges are at or below the threshold value of the adaptation sensor, S_T. The resulting target edge length becomes:

$$h_T = h\sqrt[p]{S_T/S}$$

An iterative algorithm computes S_T such that for the given adaptation cycle n the ratio of new mesh complexity, C_m^{new} to old (previous) mesh complexity, C_m^{old}, is fixed, typically around a value of 1.3. The most straightforward mesh complexity measure is the sum over all mesh nodes N of the ratio of the local mesh volume to the cube of the local edge length h_n identified as having the maximum sensor value S within the volume n:

$$C_m = \sum_{n=1}^{N}\left(\frac{1}{h_n^3}\right)V_n$$

For a uniform hexahedral mesh and constant ϕ, the local complexity is unity hence the mesh complexity C_m equals the total number of nodes, N. As the mesh becomes more "complex" in terms of the range of incident edge lengths per node, C_m grows for a given mesh size N.

The adaptation software calculates and forwards a point cloud of h_T target edge lengths to the Pointwise meshing software, augmenting the user-defined initial mesh generation state. The meshing software then generates an improved mesh to achieve the desired distribution of local target edge length, h_T. The adapted mesh preserves the initial user-defined mesh settings, and most importantly, the boundary layer meshing strategy. The adapted mesh inherently conforms to the underlying geometry known by the mesh generator. The mesh quality consistently improves with each mesh adaptation cycle, as the point cloud data consistently refine the mesh and no a priori choice of a "local subdivision" is necessary. As a bonus, the adaptation process naturally identifies and corrects areas of large mesh expansion ratio.

A controlling program manages the adaptation cycle, consisting of the following steps:

1. Generate the initial mesh
2. Compute the CFD simulation on the current mesh
3. Compute ϕ and the gradients of ϕ at the mesh vertices
4. Compute S_T to achieve a specified ratio of C_m^{new}/C_m^{old}
5. Compute the point cloud for the target mesh length h_T subject to a threshold S_T
6. Automatically generate a new mesh augmented with the point cloud data
7. Repeat steps 2) to 6) until convergence

The percentage of edges marked for adaptation controls convergence. Initially, the process identifies a small percentage of nodes for adaptation, as the truncation errors are most prominent at a few locations. As the adaptation proceeds, the truncation errors become more and more uniform, and the percentage of edges marked for adaptation increases. Typically, the adaptation is "converged" when the adaptation algorithm has marked most edges for adaptation, e.g. more than 90 %. Such percentages indicate that the truncation error is essentially uniform everywhere. It is possible to use more than one point cloud. For example, the current and prior point clouds can combine to smooth out any cycle-to-cycle edge length chatter within the adaptation process.

The entire process is remarkably computationally efficient. Because the mesh is refined only in local areas, the updated CFD simulation on the freshly adapted mesh does not require many iterations to resolve the local mesh changes. The restart process, however, does rely on a high-quality and automated interpolation procedure to map the previous solution onto the adapted mesh. This functionality is built-in to many CFD solvers. For example, ANSYS CFX requires no user input when the "initial conditions" are on a different mesh than the current CFD simulation mesh. The CPU time during the adaptation steps can be managed by converging sufficiently but not overly tightly during each step and saving tight convergence for the final adapted mesh. In a three-dimensional simulation of the "Aachen turbine"[1], the total simulation was close to the simulation time required to run from an initial guess starting on the final adapted mesh. There is a "multigrid-like" effect whereby the main flow features and the "hard work" to adjust the flow from the start occur with little computational effort with the coarser meshes. The adapted finer meshes require fewer CFD simulation iterations as the mesh changes are small at that the end of the adaptation cycle.

4 Results

A flow simulation of the Aachen turbine, consisting of 41 rotor blades rotating at 35,000 rpm at a nominal design point, illustrates the procedure. Adaptation rates p of 2.5 and 3.0 were evaluated, with a complexity ratio C_m of 1.3 during each adaptation step. The initial mesh,

[1] Walraevens, R. E., Gallus, H. E., 1995, "Stator-Rotor-Stator Interaction in an Axial Flow Turbine and its Influence on Loss Mechanisms", AGARD CP 571

generated with the Pointwise T-Rex meshing software, consisted of 400,000 nodes. The non-dimensional distance of the first node away from the wall, y^+, was chosen to be approximately 4. Figure 2 shows the initial mesh with hexahedral elements in the boundary layer region.

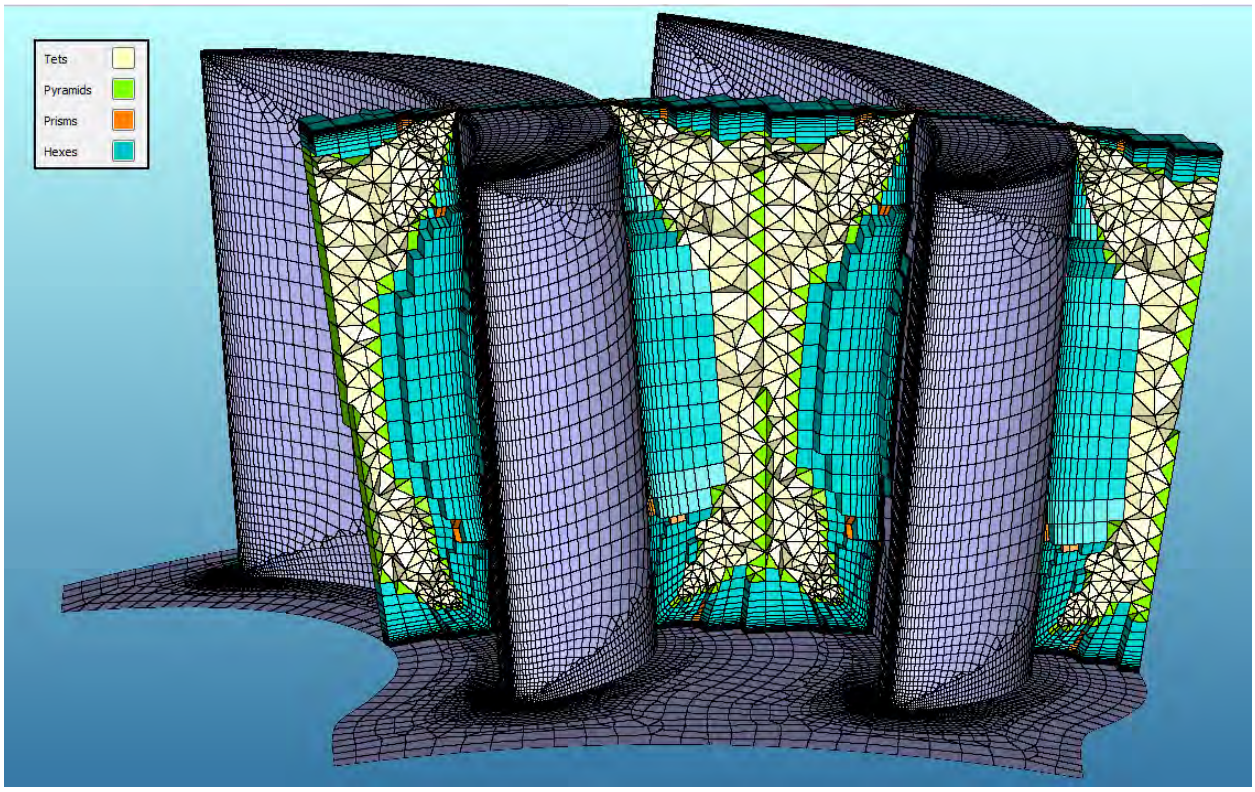

Figure 2: Aachen turbine - Initial mesh

Figure 3 illustrates the iterative convergence of the adaptation process. It consists of nine adaptation steps with the rate controlled by $p = 3$. The adaptation cycle terminated when more than 90 % of the edges were marked for adaptation.

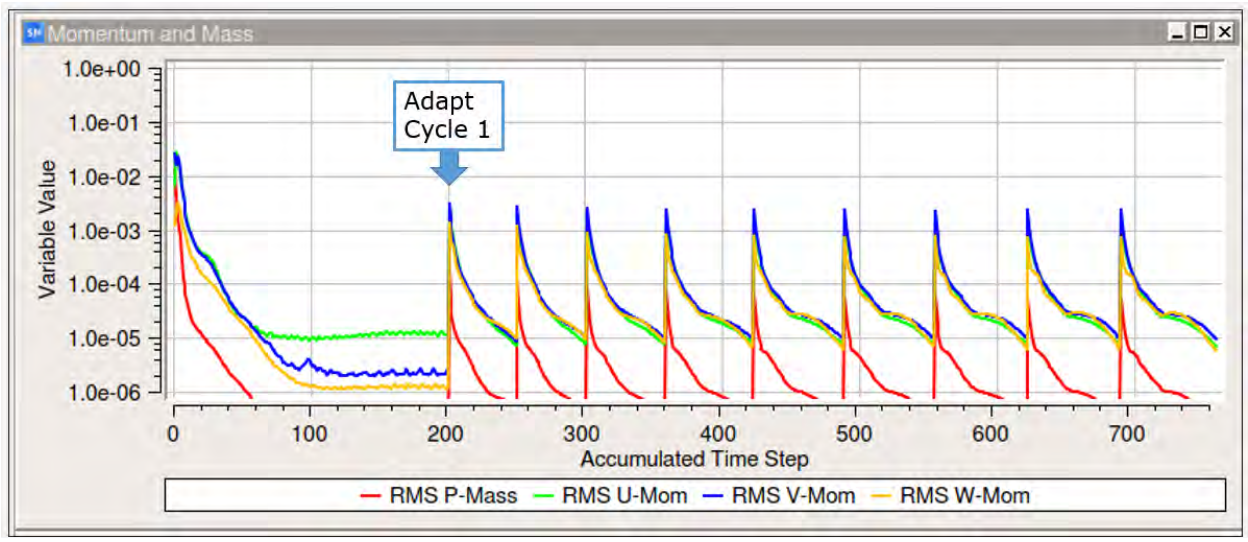

Figure 3: Aachen turbine – convergence rates of the r.m.s.-residual norms

Figure 4 shows the adapted mesh at mid-span between hub and shroud. The mesh has adapted in the wake flow and at the leading edge while preserving the hexahedral boundary layer mesh region.

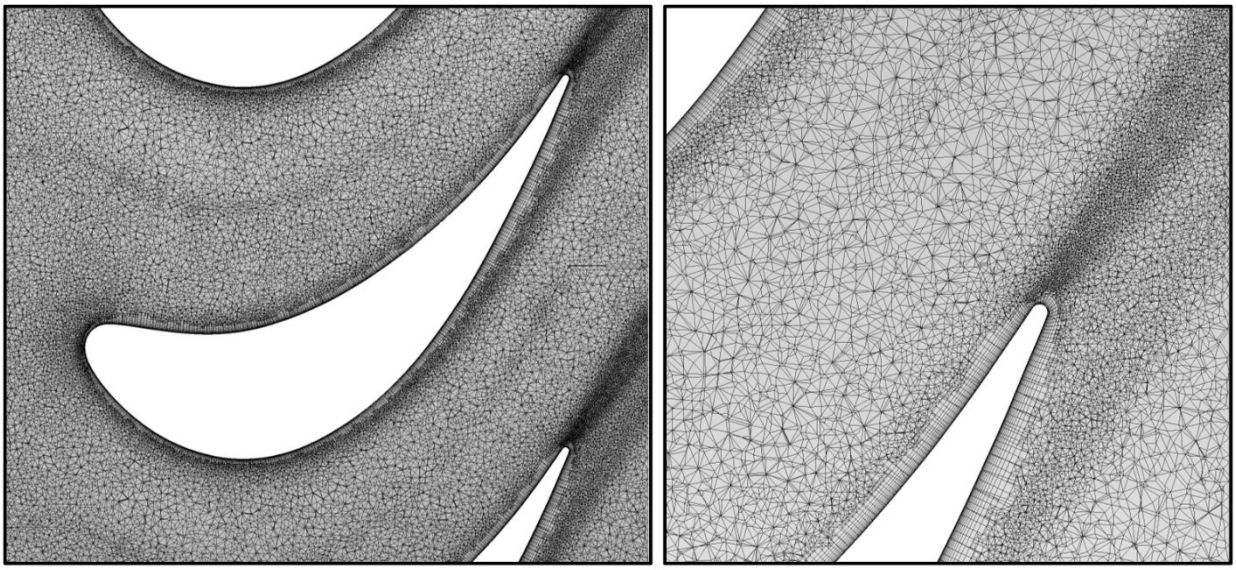

Figure 4: Aachen turbine – adapted mesh at mid-span

The hub-side horseshoe vortex around the leading edge, shown in Figure 5, impacts the end-wall shear stress in Figure 6. The adapted mesh in Figure 7 resolves these subtle flow features.

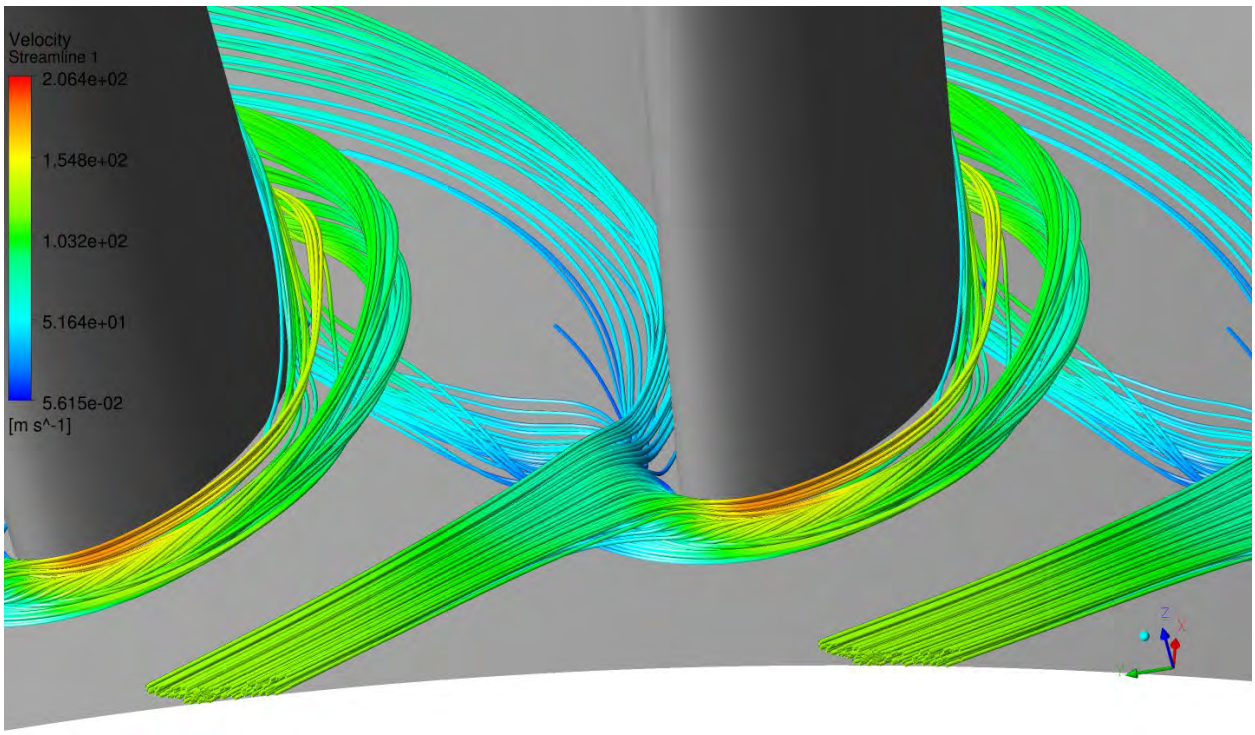

Figure 5: Aachen turbine – horseshoe vortex at the leading edge

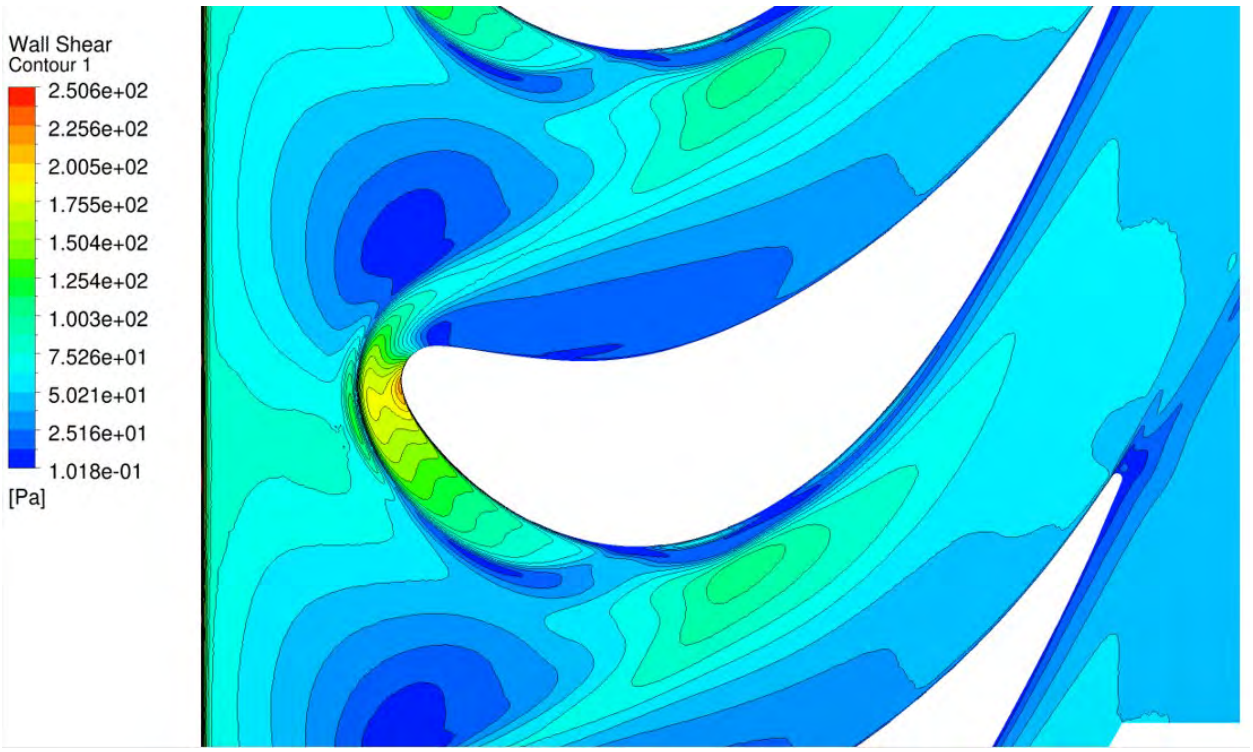

Figure 6: Aachen turbine – wall shear stress contours on the hub surface

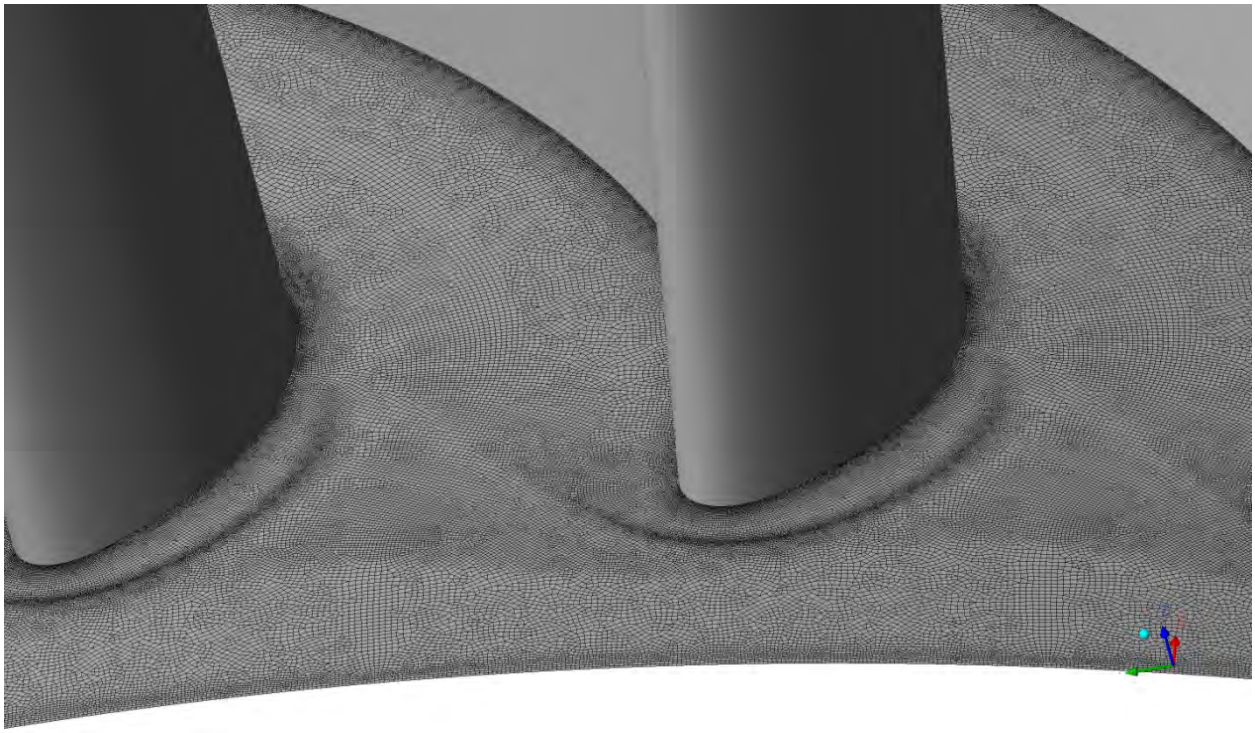

Figure 7: Aachen turbine – adapted mesh on the hub surface

Figure 8 shows the $p = 2.5$ simulation times per adaptation cycle for the cumulative simulation times up to the end of each cycle, and the time to compute on a given adapted mesh from initial conditions. At the end of the adaptation cycles, the cumulative time for all adaptation cycles is less than or equal to the computational time to perform one simulation from start to finish on each adapted mesh. The curves show that there is little to no computational penalty by employing mesh adaptation relative to a single solution on any given mesh. Adaptation can be efficiently performed across a speedline, ensuring an optimal mesh for each operating

point, for example, by resolving the specific secondary flows and wake departure angles with adapted meshes unique to each operating point.

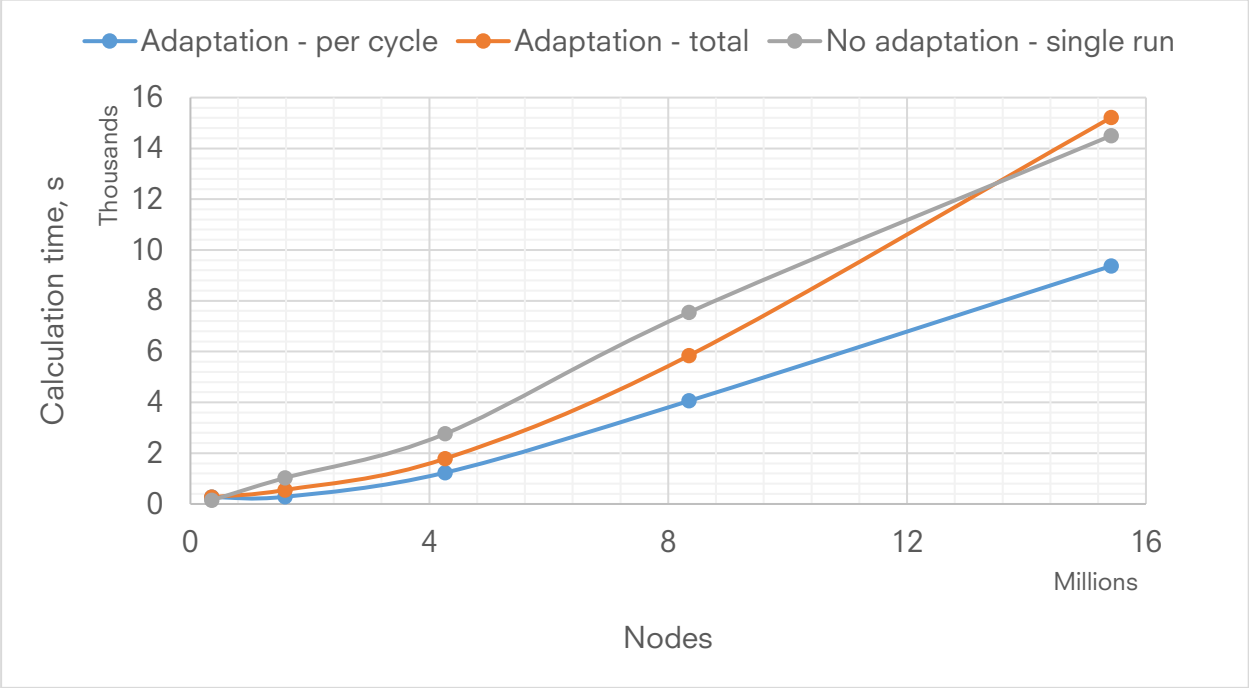

Figure 8: Aachen turbine – simulation times

5 Summary & Outlook

A new adaptation process exists that achieves the hopes and promises of adaptation from long ago. Pointwise and ISimQ have developed an efficient adaptation procedure that:

- adapts to the underlying geometry,
- efficiently resolves the mesh with high aspect ratios within boundary layer regions,
- provides effective control over the rate of adaptation, focusing on regions where the mesh is coarse, thereby
- successively improving the mesh quality, leading to a
- highly robust and efficient automated mesh adaptation procedure for the effective control of discretisation error on real-world cases.

6 Contact

If you are interested in the mesh adaptation procedure or if you require additional information, please send an email to info@isimt.com or info@pointwise.com