Computation of Steady Flow Around Circular Cylinder in a Channel at Re = 200

For context, we note that Section 9.12.2 presented 2D laminar flow around a circular cylinder, while Section 9.12.3 described calculations of 3D laminar flow around a circular cylinder mounted between two walls in a duct with a square cross-section.

In this report we extend the 3D presentation. Three grid types are used for flow analysis: trimmed Cartesian, tetrahedral and polyhedral. Prism layers along cylinder and channel walls were created in an analogous way in all three cases. The duct axis points in the *x*-direction and the cylinder axis points in the *y* direction (horizontal).

Figure 1 shows the geometry of the solution domain and the coarse polyhedral grid on the boundaries; Fig. 2 shows a longitudinal section at $y = 0$ through all three kinds of grid. The duct extensions are (in meters): $-1.75 \le x \le 3.25$, $-0.5 \le y \le 0.5$ and $-0.5 \le z \le 0.5$; the cylinder is positioned at the coordinate system origin and has a diameter of 0.4 m. The cylinder thus blocks 40 % of the duct cross-section area. The hypothetic incompressible fluid has a (constant) density of 1 kg/m^3 and a viscosity of 0.005 Pa·s. At the inlet (x = -1.75 m), a uniform velocity field with u_x = 1 m/s was specified, while at the outlet (x = 3.25 m), a constant pressure was prescribed. The Reynolds number based on duct height is Re = 200. The velocity field was initialized (for the coarsest grids from each family) with a constant velocity in the *x*-direction, equal to the inlet velocity. The flow around a cylinder in an infinite environment would be unsteady under the same conditions, but due to the confinement in the duct, the laminar flow is still steady in this case.

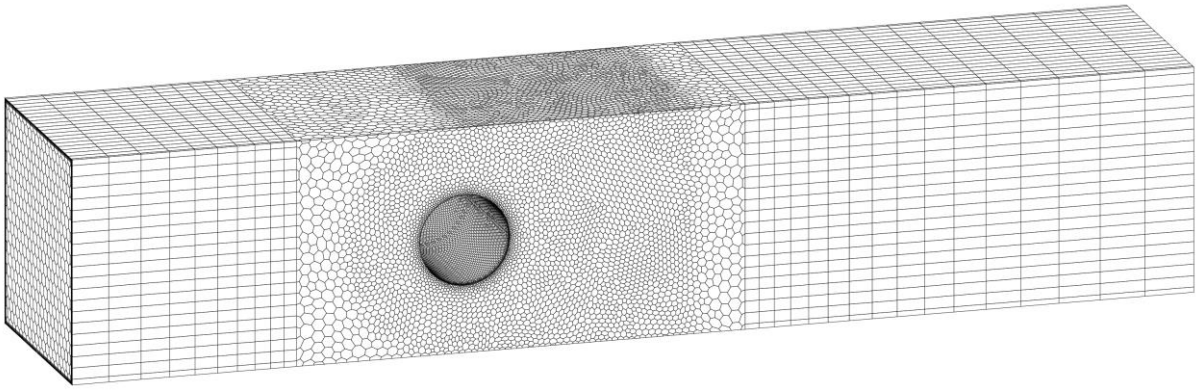

Fig. 1: Cylinder in a duct: geometry and a coarse polyhedral grid on solution domain boundaries.

We generated three types of grid in a shorter duct segment around the cylinder and performed the so-called *grid extrusion* along the duct axis, both from the upstream and from the downstream duct cross-section. This leads to prismatic cells in the portions of the duct downstream of the inlet and upstream of the outlet, in which a gradual expansion or contraction of cells in *x*-direction is achieved, as can be seen in Fig. 1, where prisms with a polygonal base are created. In the case of tetrahedral grid, the extruded sections are made of elongated prisms with a triangular base, while in the case of trimmed hexahedral grid, the extruded regions contain elongated hexahedra.

Local grid refinement around the cylinder, and to some extent downstream of it, was invoked by specifying two volume shapes for which a smaller cell size was required: a larger cylindrical

shape around cylinder and a block on the downstream side. The grid generator (region-based version was used, to enable the above-mentioned extrusion) generates regular-shaped polyhedra (dodecahedra) and tetrahedra within refinement zones specified by regular body shapes, as can be seen in Fig. 2.

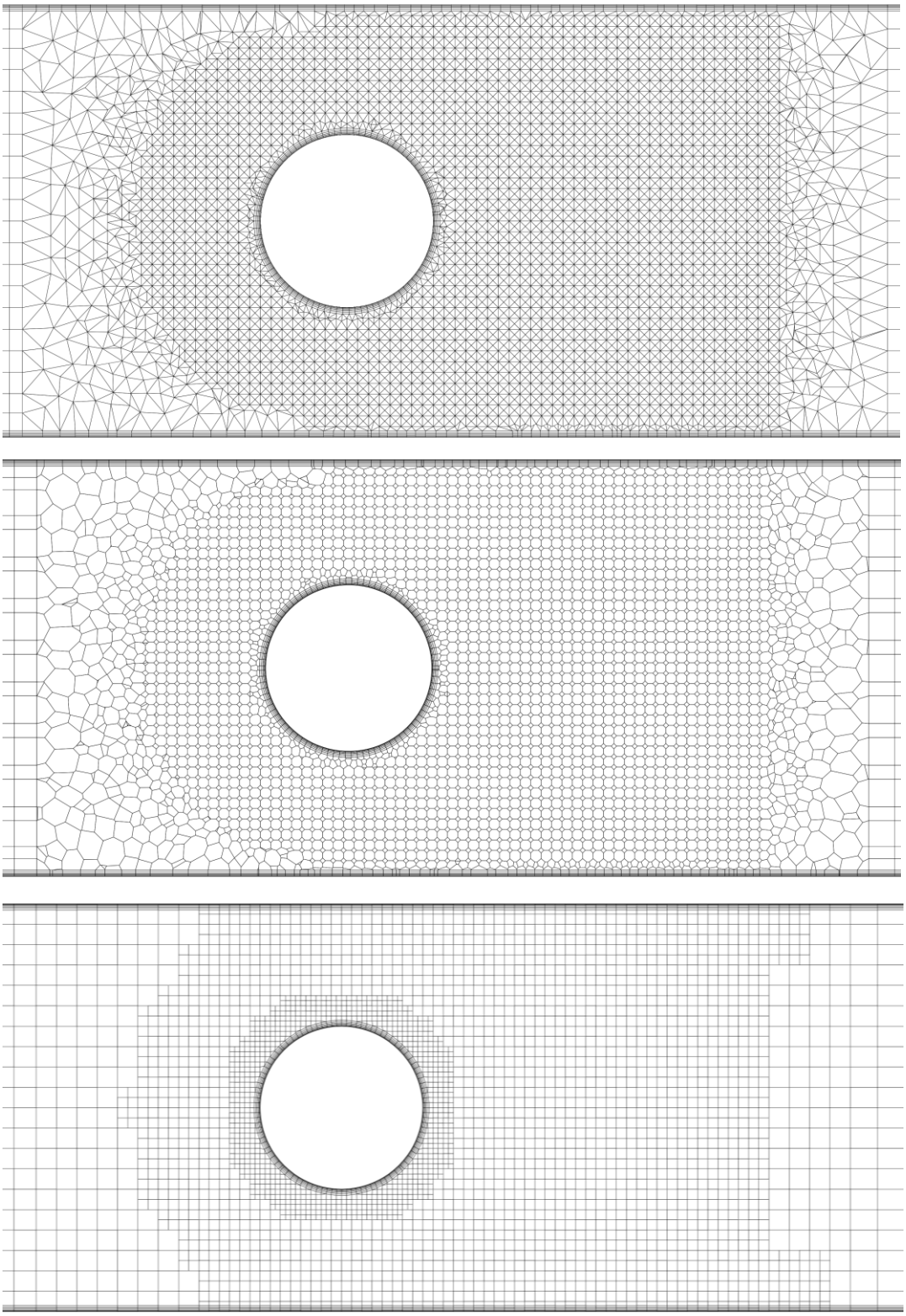

Fig. 2: Computational grid in the longitudinal symmetry plane $y = 0$: tetrahedral grid (upper), polyhedral grid (middle) and trimmed hexahedral grid (lower).

Computations were performed using Simcenter STAR-CCM+ V 14.04. The second-order upwind scheme is used for convection (linear extrapolation to the cell-face center using the variable value and the gradient at the upstream cell center), while linear shape functions are used for approximation of gradients (leading to central differences on Cartesian grids). The SIMPLE algorithm is used with under-relaxation factors of 0.9 for velocities and 0.1 for pressure.

Table 1: The "base-size" parameters for grid generation

Grid no.	Polyhedral	Trimmed	Tetrahedral
1	0.056000	0.045000	0.056000
2	0.037333	0.030000	0.037333
3	0.024888	0.020000	0.024888
4	0.016593	0.013333	-

The "base-size" parameter, which controls the cell size in the grid, was adjusted such that a similar number of polyhedral and trimmed cells are generated (see Table 1). For tetrahedral and polyhedral grids, the base-size parameter was kept the same. Because with this set-up many more tetrahedral than polyhedral cells are generated (cf. Table 2), only three tetrahedral grids were used. Note that the cell size is reduced by a factor 1.5 with each refinement step (this is strictly so everywhere only in Cartesian cells of the trimmed grid; for other grid types, this is true only on average). Numbers of cells for each grid type and refinement level are shown in Table 2.

Grid extrusion is especially helpful when simulating internal flows in geometries with duct- or pipe-like inlet and outlet passages. Prismatic cells obtained by extrusion can be made to grow gradually towards a boundary, which avoids many kinds of numerical problems and typically leads to a better convergence than if polyhedral or tetrahedral grid extends all the way to the boundary. The meshing tools in more recent versions of Simcenter STAR-CCM+ may not create the same grids as those used here, due to the changes in meshing algorithms. However, the results presented here are representative enough to give an estimate of accuracy and efficiency.

Table 2: The numbers of cells generated for each grid type

Grid no.	Polyhedral	Trimmed	Tetrahedral
1	366,223	442,740	1,562,707
2	1,118,317	1,232,001	5,183,125
3	3,410,981	3,633,246	16,947,396
4	10,658,452	11,250,463	-

Simulation always starts on the coarsest grid; once the solution is obtained, it is interpolated (automatically as part of the grid refinement process) to the next finer grid and serves there as the initial guess for the iteration process. The reported computing times for any grid thus include computing times for all coarser grid levels.

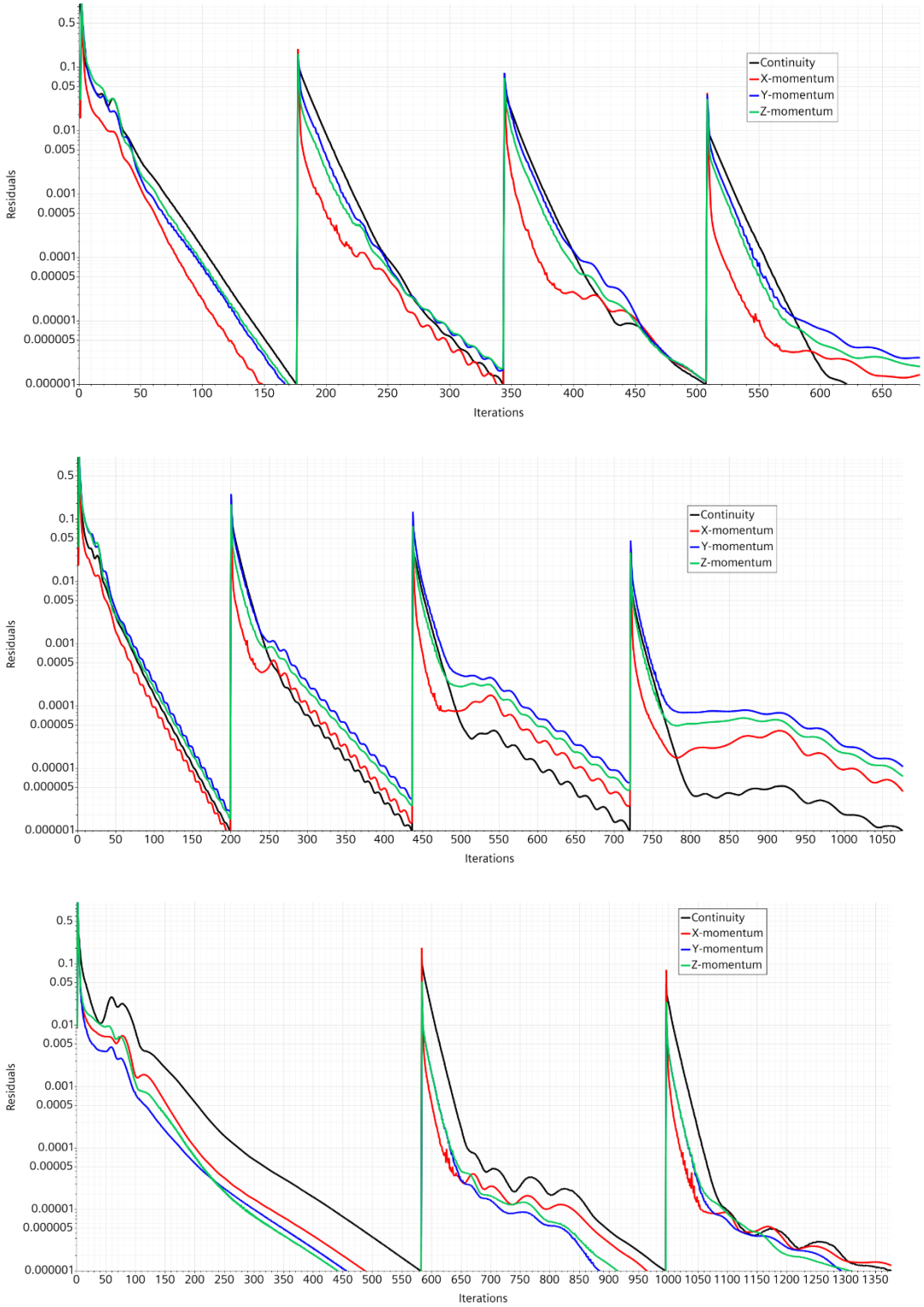

Fig. 3: Variation of the L_1-norm residuals with increasing number of outer iterations in the SIMPLE algorithm for the 3D laminar flow around a circular cylinder in a duct (each jump in residual level indicates the change to a higher grid level): polyhedral (top), trimmed (middle) and tetrahedral grids (bottom).

Figure 3 shows the variation of the L_1 (absolute value sum)-norm residuals with outer iterations for all grids. The convergence criterion for SIMPLE-iterations was set lower than necessary,

because we want to ensure that the obtained results are free from iteration errors, so that only discretization errors remain. The residuals fall one order of magnitude very quickly, but iteration errors usually do not follow residuals from the very beginning. A safe way to estimate iteration errors from a residual plot is to extrapolate backward from the final state. For the coarsest grids, projecting backward along the mean slope we reach iteration 0 at the level of around 0.1, thus indicating that the true reduction of iteration errors is around 5 orders of magnitude. Indeed, a closer look at monitoring values of velocities and pressure shows no variation in the first 5 significant digits, which is another useful check to make sure that the solution has really converged. For the finer grids, we see that the initial level of residuals is always lower than on the preceding grid level. This is due to the fact that we start with a better initial solution on each finer grid level. Starting iterations on finer grids with initialization of variables as used on the coarsest grid (velocity in x-direction equal to the mean channel velocity, all other variables set to zero) would lead to substantially higher numbers of required iterations to reach the same level of residuals. This is even more pronounced when turbulent flows are computed, where the computing time may be cut down by a factor of 3 to 4 by using this initialization approach.

Figure 4 shows the variation of residuals with outer iterations for the level-3 polyhedral grid when the simple initialization is used. In this case, 414 iterations are required to reach the specified convergence criterion. When iterations are initialized with interpolated solution from level-2 grid, only 164 iterations are required, cf., Fig. 3. This means that, without good initialization, 2.83 times more iterations (and computing time) is needed. This ratio becomes larger as the grid is further refined.

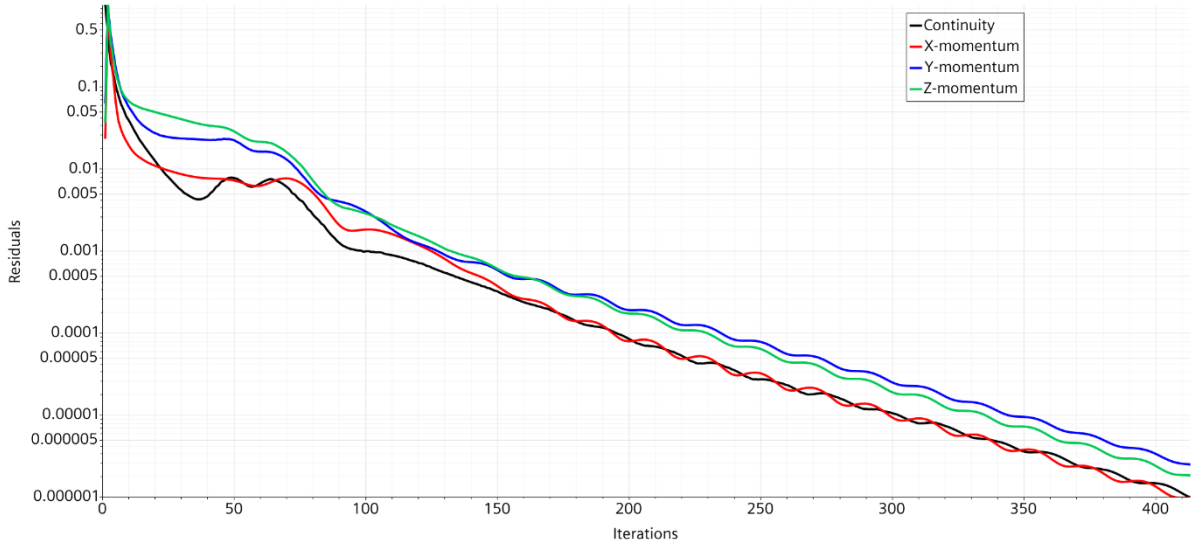

Fig. 4: Variation of the L_1-norm residuals with increasing number of outer iterations in the SIMPLE algorithm for the 3D laminar flow around a circular cylinder in a duct, computed on level-3 polyhedral grid starting with simple initialization: $u_x = 1$ m/s, $u_y = u_z = 0$, $p = 0$.

Figure 5 shows velocity vectors in the two longitudinal symmetry planes, computed using the level-4 trimmed hexahedral grid. One can see in the horizontal plane ($z = 0$) how velocity vectors turn backward at both cylinder ends; that indicates the formation of horse-shoe-vortexes where cylinder is in contact with channel walls. The vertical cross-section shows a strong

acceleration of the flow as it passes around cylinder; velocity vectors double in length around cylinder compared to their size upstream of it. Behind cylinder a recirculation zone is formed, which is almost two diameters long. Because the flow is steady, the velocity field is symmetric in both section planes, due to geometric symmetry. However, note that the length of the recirculation zone behind cylinder varies in the lateral direction. From the $z = 0$ plot of Fig. 5, we can see that it is longest at the center, becomes smaller as one moves towards side walls but then increases again close to walls. Thus, three-dimensionality effects are not limited to cylinder ends where they meet side walls – the effect of side walls is visible across the whole cylinder span.

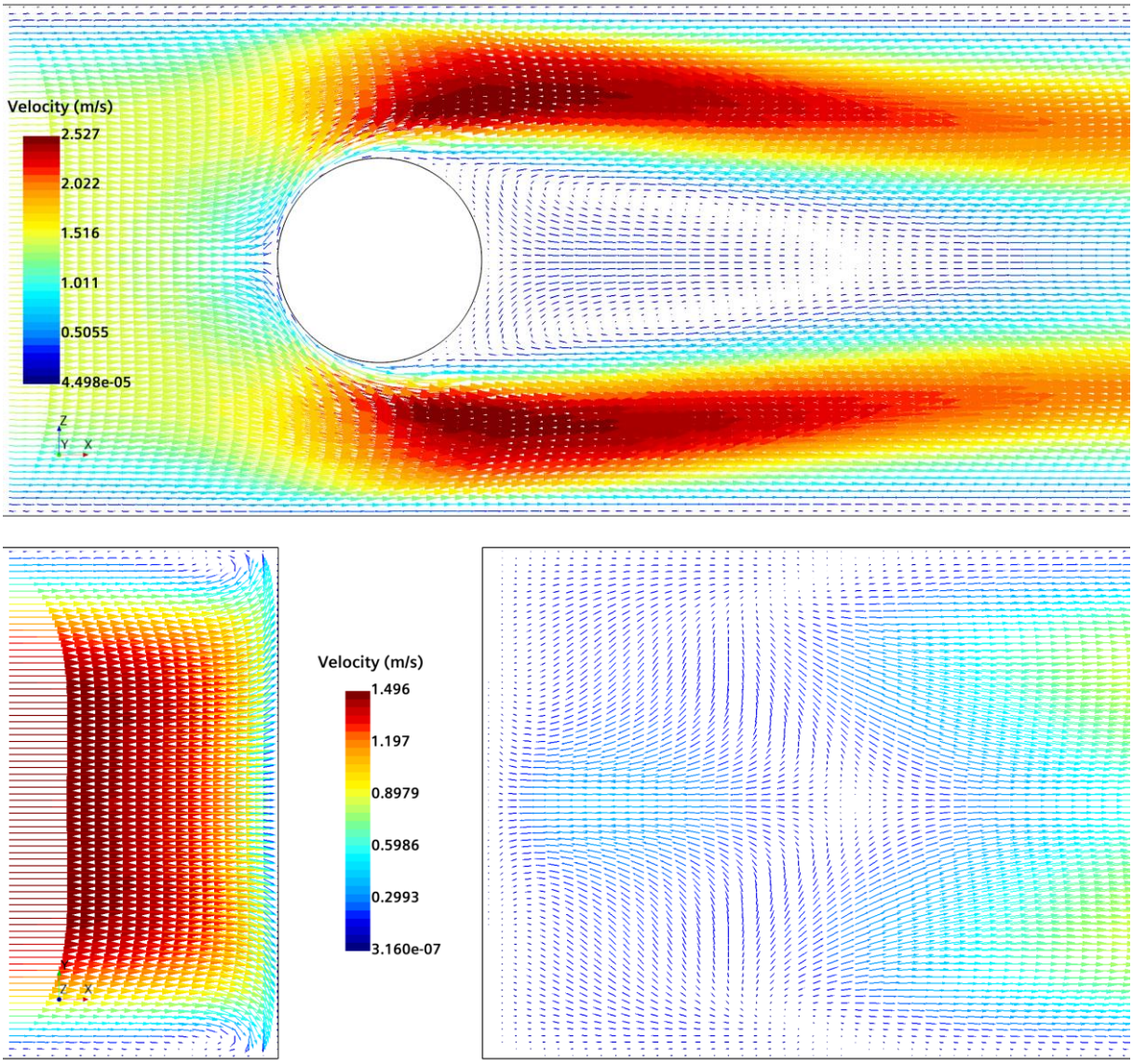

Fig. 5: Velocity vectors in two longitudinal symmetry planes, computed using level-4 trimmed hexahedral grid: $y = 0$ (upper) and $z = 0$ (lower).

Figure 6 shows pressure distribution in the vertical symmetry plane, computed on level-4 polyhedral and trimmed hexahedral grid. Only very small differences can be spotted on some contour lines – the solutions are practically identical. The isobars around cylinder show that the

highest pressure is found at the front stagnation point, where the flow impinges onto cylinder wall, and the lowest pressure is located at cylinder sides. Pressure recovers to some extent on the downstream side, but due to viscous losses (that lead to flow separation and the recirculation zone), it is much lower at the downstream stagnation point than at the upstream one.

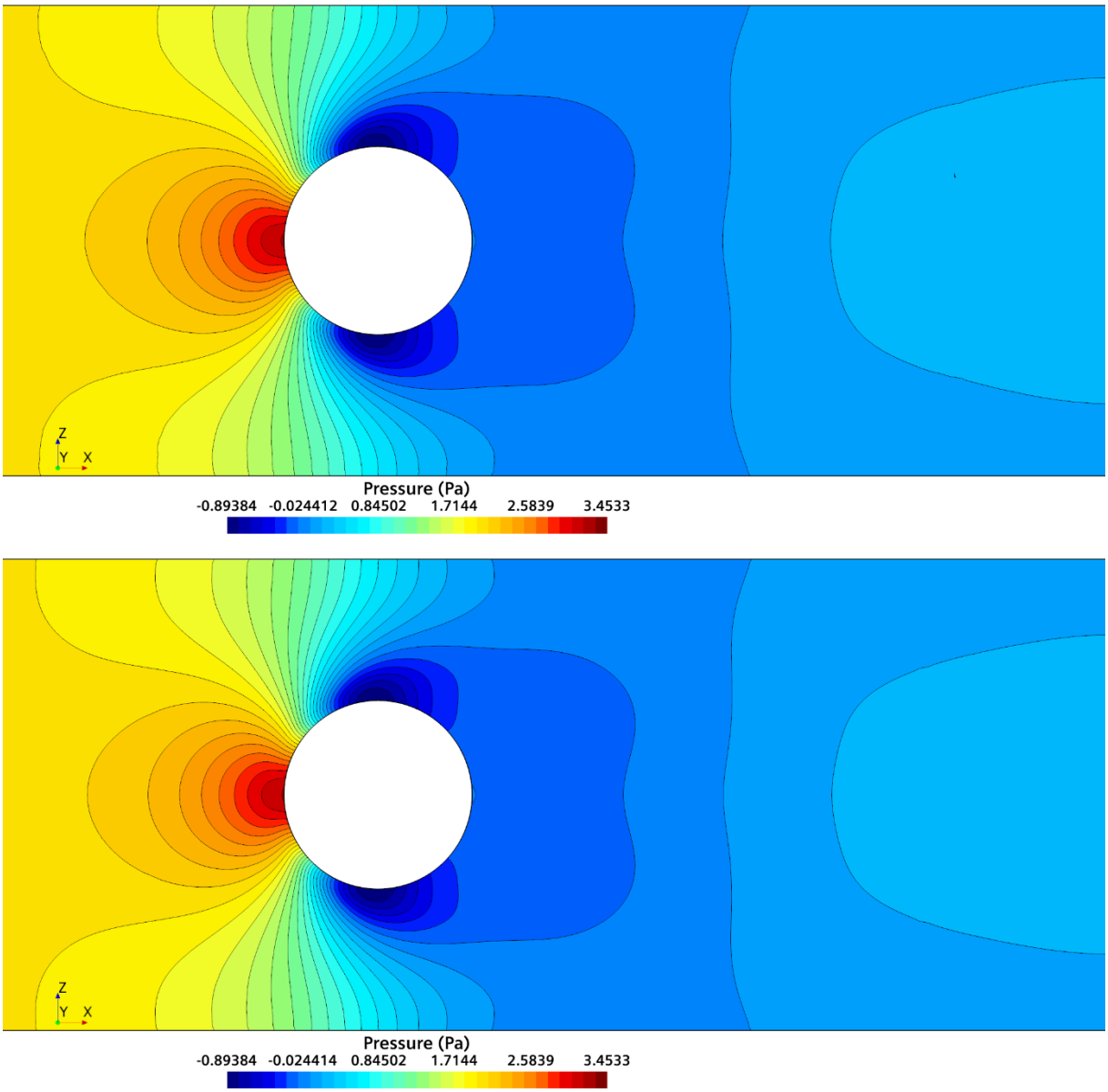

Fig. 6: Pressure distribution in the longitudinal symmetry plane $y = 0$, computed using level-4 grids: polyhedral (upper) and trimmed hexahedral (lower). Flow is from left to right.

Figure 7 shows pressure distribution in the vertical symmetry plane, computed on level-3 polyhedral and tetrahedral grid. The minimum and the maximum pressures differ only at the fourth decimal place – the solutions are almost identical. Only the contour lines far ahead of cylinder are less smooth in the case of tetrahedral than on polyhedral grid.

Figure 8 shows the profiles of u_x velocity along one line in the vertical symmetry plane 0.875 D downstream of cylinder, and along one line in the horizontal symmetry plane 1.875 D

downstream of cylinder axis, computed on all 4 polyhedral grids. With the refinement factor of 1.5, one expects from a 2nd-order method that the difference between solutions obtained on consecutive grid levels reduces by a factor 2.25 (square of the refinement ratio). By comparing relative distances between curves for different grids, one can indeed confirm the expected behavior. For the profile along the vertical line, the differences between solutions obtained on all grids but the coarsest one are very small; the peak values are under-predicted on the coarsest grid by ca. 2%, while the profiles from grids 3 and 4 can hardly be distinguished in the graph.

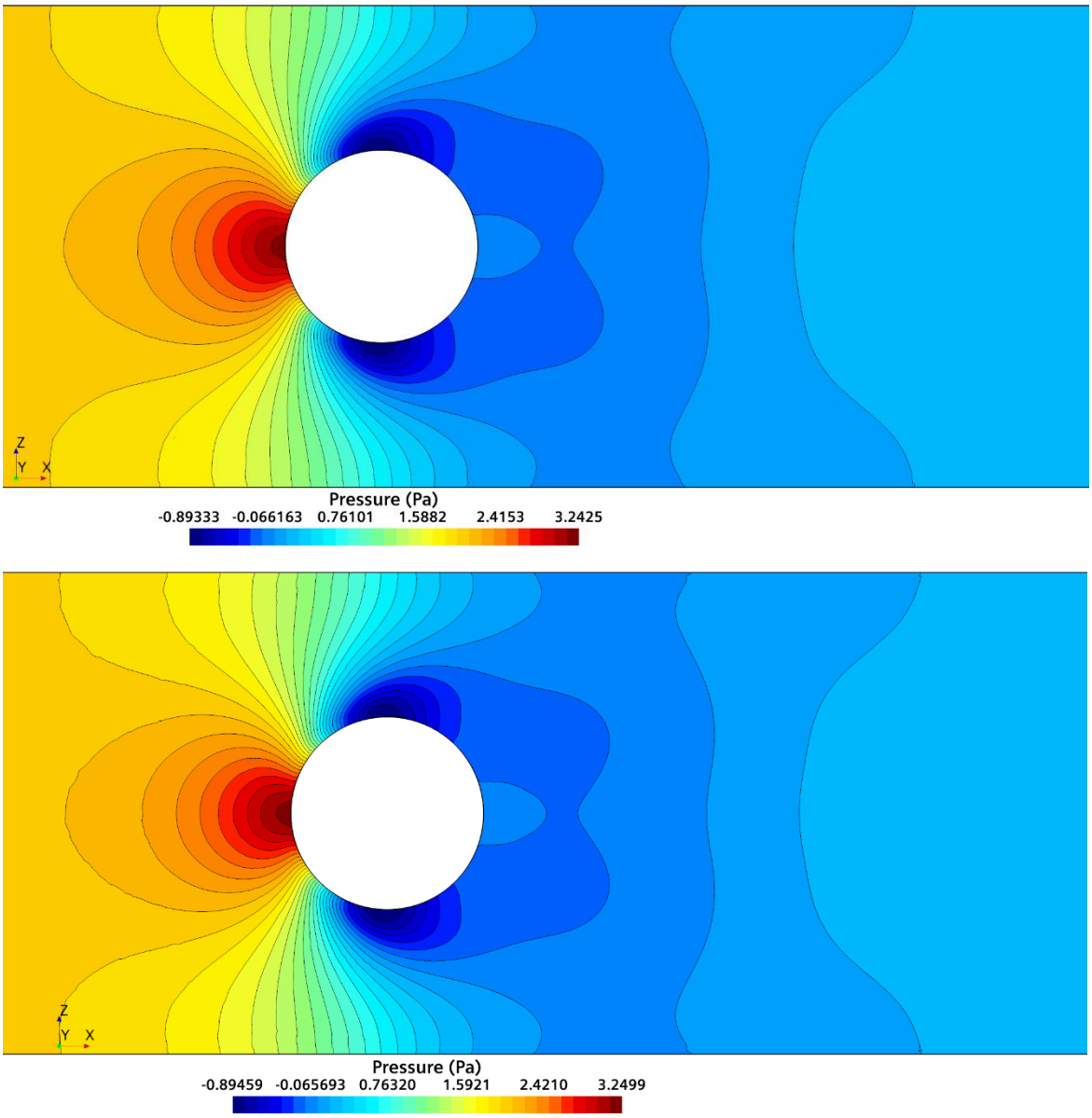

Fig. 7: Pressure distribution in the longitudinal symmetry plane $y = 0$, computed using level-3 grids: polyhedral (upper) and tetrahedral (lower).

The differences are better visible in the horizontal section further downstream from cylinder, inside recirculation zone: the peak values are here one order of magnitude smaller than the mean velocity in the duct and thus even small differences are clearly distinguishable. The difference

between the coarsest and the next finer grid is appreciable, while differences between solutions from grids 2, 3 and 4 are relatively small.

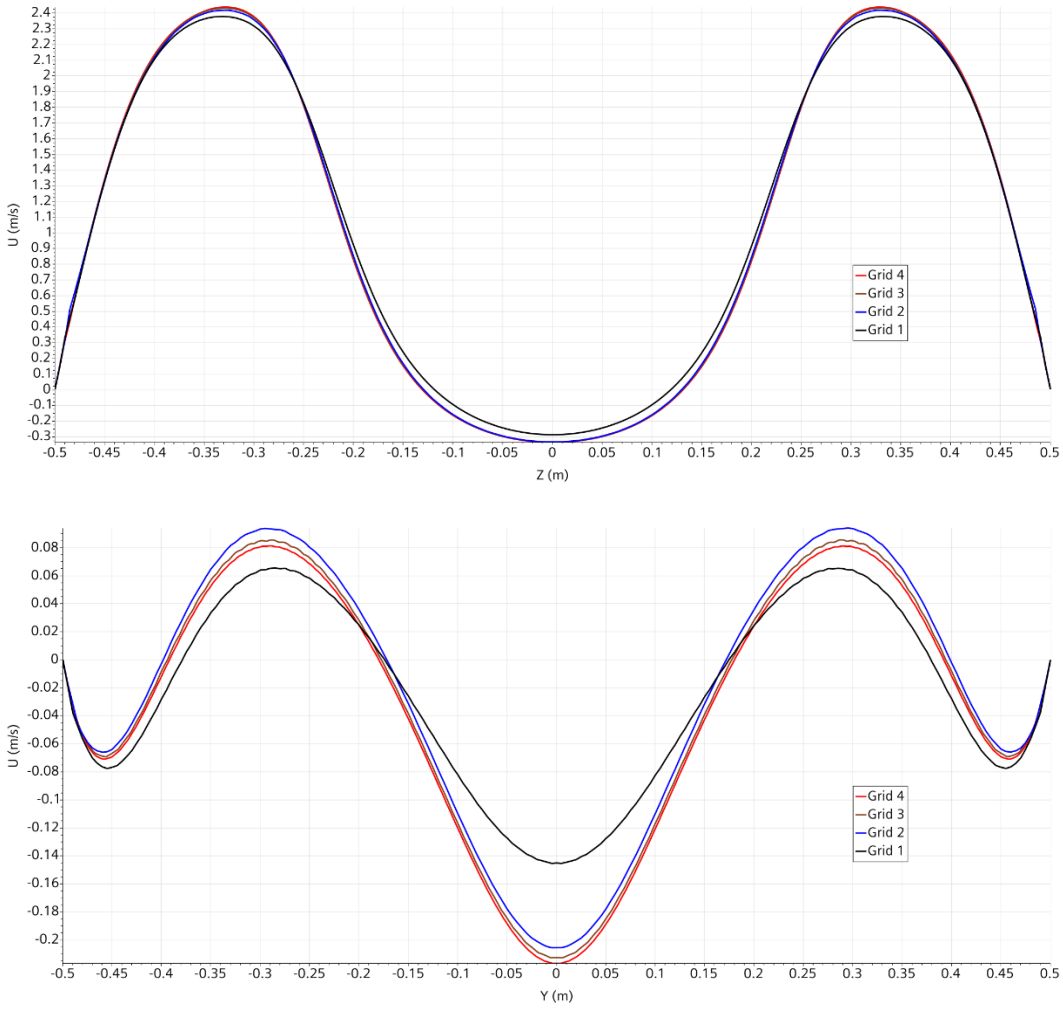

Fig. 8: Profiles of u_x-velocity downstream of cylinder, computed on 4 systematically refined polyhedral grids: $x = 0.35$, $y = 0$ (upper) and $x = 0.75$, $z = 0$ (lower)

Figure 9 shows the comparison of the same velocity profiles computed on finest grids of different type. The differences are smaller than the difference between solutions on the two finest grids of the same type, see Fig. 8. Especially the profiles along the vertical line fall exactly on top of each other, which is why only the result from polyhedral grid is shown as line and the others are presented by symbols. Even in the horizontal line across the recirculation zone, lines from polyhedral and trimmed hexahedral grids at level 4 are almost identical. The line representing tetrahedral grid is from level 3, and it differs for this profile a little from the other two lines. As shown above for pressure distribution (cf. Fig. 7), solutions from polyhedral and tetrahedral grids at the same level are practically the same, so no doubt that results from level-4 tetrahedral grid would also match level-4 results from the other two grid types.

This confirms that, when the grid is sufficiently fine, the same grid-independent solution will be obtained, no matter which type of computational grid is used. What differs is the effort required to generate the grid and to solve the Navier-Stokes equations. Simcenter STAR-CCM+

generates all three grid types fully automatically, so user effort is the same in all cases. The actual computing time for grid generation is – in this case – substantially shorter than the time needed to solve the equations, which is why we do not bother to present such timings.

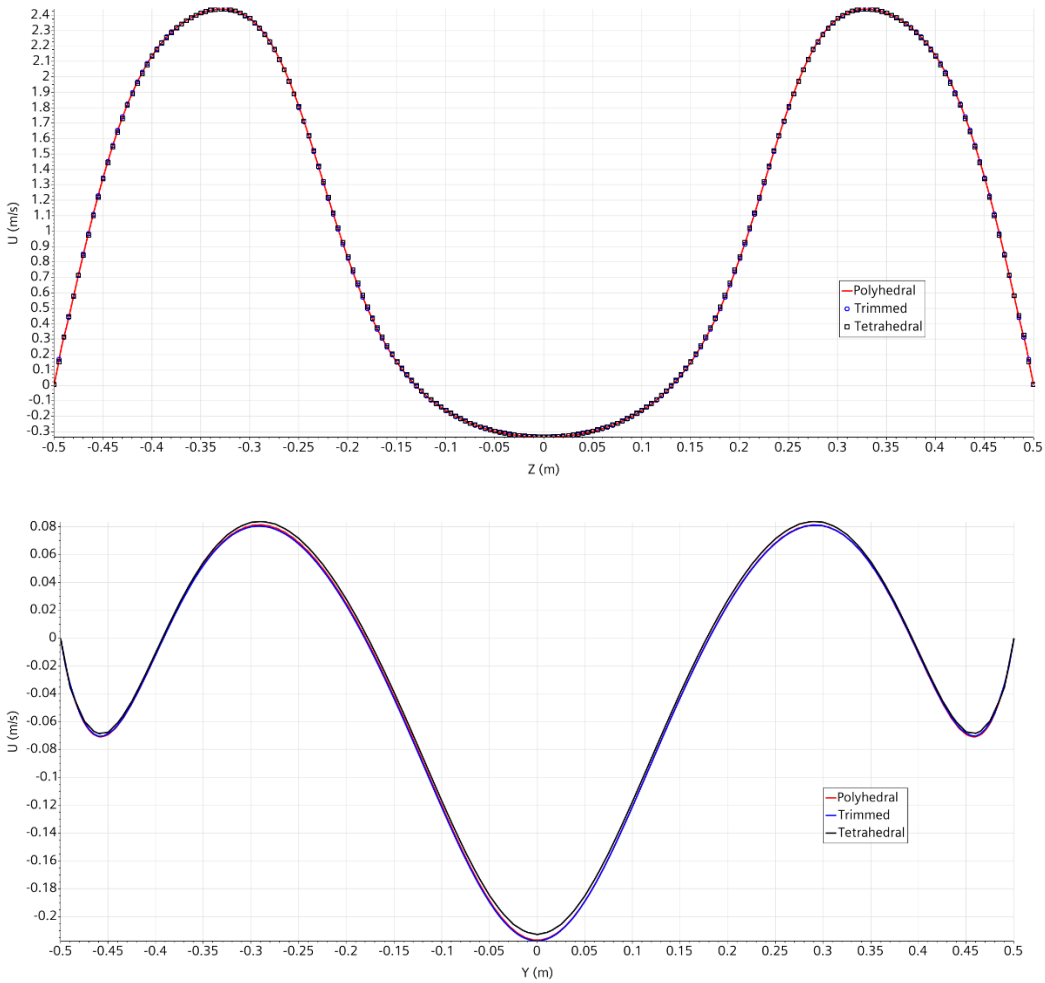

Fig. 9: Profiles of u_x-velocity downstream of cylinder, computed on three different grid types: $x = 0.35$, $y = 0$ (upper) and $x = 0.75$, $z = 0$ (lower).

The times needed to solve the Navier-Stokes equations are presented graphically in Fig. 10 as CPU-time (in seconds) vs. number of cells. On coarse grids (number of cells of the order of million or less), the lowest computing effort per cell results when trimmed hexahedral grids are used. However, the computing time per cell increases fastest for trimmed hexahedral grids, followed by polyhedral grids: for this test case, they are even at about 4.5 million cells. For finer grids, polyhedral grids need less computing time. Note that we are here looking at computing time per cell required to reach the convergence criterion – not per iteration. It is obvious that the least computing effort per cell and iteration is required for tetrahedral cells, because they have only 4 faces (and thus 4 neighbors, i.e., 4 non-zero off-diagonal coefficients in the matrix). Hexahedral cells have 6 neighbors, and polyhedral cells, on average, twice as many. However, as can be seen from Fig. 3, polyhedral grids need the lowest number of iterations to reach the specified convergence criterion (which was the same for all grids, and the under-relaxation factors as well as all parameters for the linear equation solver from AMG-

family were also the same in all computations). Less than 700 iterations were needed to obtain solutions on all 4 polyhedral grids (less than 200 on level 4), while ca. 1100 iterations were needed for all 4 trimmed hexahedral grids (ca. 350 on the finest level). Tetrahedral grids need many iterations to converge on coarse levels, but the slope is slightly more favorable as the grid gets finer. However, as seen in Fig. 7, solutions from the same grid level for polyhedral and tetrahedral grids are practically identical, but the number of cells differs by a factor 6 in favor of polyhedral grids.

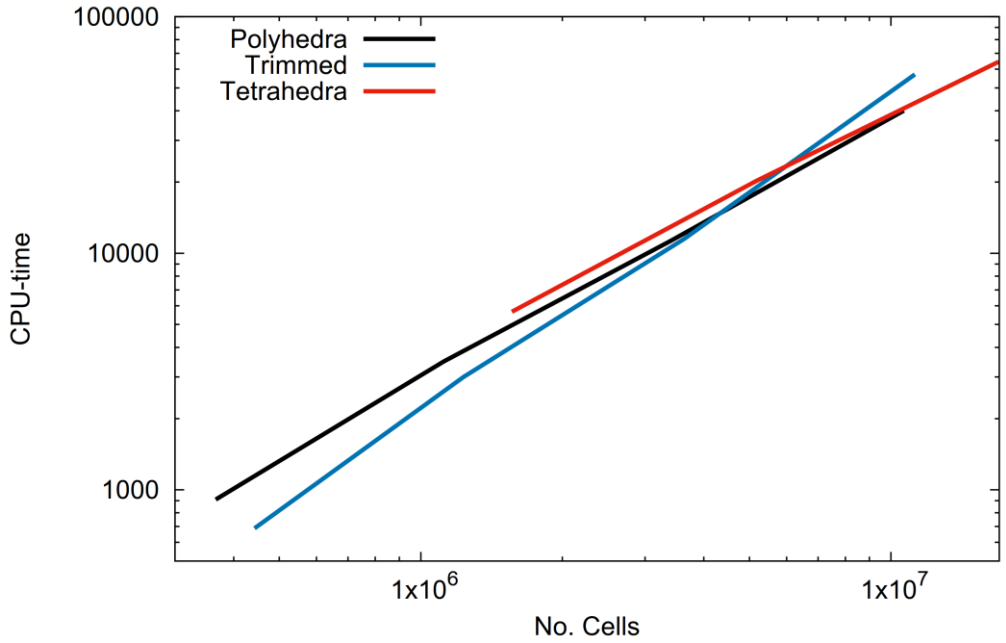

Fig. 10: Computing time (in seconds) as a function of the number of cells for three different grid types (because on each finer level the solution is initialized with the result from the next coarser level, computing times include the effort from all coarser grid levels).

While the results will vary somewhat from case to case, the findings from this study are quite representative and can be taken as representing many typical simulations of internal incompressible flows.

Note that the above statements are valid when the same kind of approximations is applied on all grids (here: midpoint rule for integral approximations, linear interpolation or extrapolation, linear shape functions for gradient approximation). The ratios might be different if one applied discretizations specially tuned to one particular grid type.