Copyright © 2009 Tech Science Press CMES, vol.41, no.2, pp.147-175, 2009

An automated approach for solution based mesh adaptation to enhance numerical accuracy for a given number of grid cells
Applied to steady flow on hexahedral grids

Peter Lucas[1], Alexander H. van Zuijlen[1] and Hester Bijl[1]

Abstract: Mesh adaptation is a fairly established tool to obtain numerically accurate solutions for flow problems. Computational efficiency is, however, not always guaranteed for the adaptation strategies found in literature. Typically excessive mesh growth diminishes the potential efficiency gain. This paper, therefore, extends the strategy proposed by [Aftosmis and Berger (2002)] to compute the refinement threshold. The extended strategy computes the refinement threshold based on a user desired number of grid cells and adaptations, thereby ensuring high computational efficiency. Because our main interest is flow around wind turbines, the adaptation strategy has been optimized for flow around wind turbine airfoils. The proposed strategy was found to yield computationally efficient computations for flow around wind turbine airfoils as well as for other flow problems.

Keywords: Solution based mesh adaptation, computation refinement threshold, hexahedral grids, computational efficiency.

1 Introduction

Computational fluid dynamics (CFD) has developed over the last decades into a comprehensive design tool. It is now possible to compute the flow around complex aerospace configurations such as complete aircraft, helicopter and spacecraft, see for example [Kroll and Fassbender (2005); Vos, Rizzi, Darracq, and Hirschel (2002)]. This rapid increase in CFD applications has become possible by an ever increasing computer power, more efficient algorithms and progress in physical modeling [Vos, Rizzi, Darracq, and Hirschel (2002)].

[1] Faculty of Aerospace Engineering, Delft University of Technology, P.O. Box 5058, 2600 GB Delft, The Netherlands

However, even the current state of the art Reynolds averaged Navier-Stokes based CFD codes are not yet practical tools for complex design trade off studies due to the tremendous CPU costs involved, see e.g. [Simms, Schreck, Hand, and Fingersh (2001); Chaviaropoulos, Nikolau, Aggelis, Soerensen, Johansen, and Hansen (2003a); Chaviaropoulos, Nikolau, Aggelis, Soerensen, Johansen, and Hansen (2003b)] for wind turbine design. Hence, there is still a need to increase the computational efficiency of CFD codes.

In this paper we focus on grid adaptation through h-refinement. Grid adaptation can enhance the computational efficiency since the adapted mesh is only dense in the important regions. Especially for unsteady flows, where important flow phenomena can move through the domain, mesh adaptation has a potential since non-adapted grids need to be almost uniformly dense to capture all important flow phenomena. However, since the grid needs to be adapted many times for an unsteady flow field, a strict control over the mesh size growth is indispensable. Without a strict control on the mesh size growth over-refinement can easily occur, thereby reducing the potential efficiency gain, even for steady computations.

Many of the mesh adaptation strategies that can be found in the literature, however, lack this strict control, see for example [Bijl, Zuijlen, and Mameren (2005); Mavriplis (2000); Wang and Chen (2002)]. In these works the grid is over refined such that the adapted grids contain much more cells than such a case typically would require, hence the adaptation strategy likely decreased the computational efficiency. Furthermore, the total computational costs required to obtain the grid adapted solution are often omitted, see for example [Cavallo and Baker (2000); Ham, Lien, and Strong (2002); Oh, Kim, and Kwon (2002); Wang and Chen (2002)]. This makes it difficult to judge the computational efficiency of the adaptation strategy. As the actual grid adaptation and computing the solution on a series of refined grids requires computational time, it is important to evaluate the overall computational costs.

The goal of this paper is, therefore, to develop a solution based mesh adaptation strategy that enables a strict control on the mesh size growth to enable computationally efficient flow computations. In this paper a novel strategy is proposed where a strict control on the mesh size growth is achieved by computing the refinement threshold based on a target number of grid cells and number of adaptations. This is a somewhat unconventional approach because commonly the refinement threshold is set such that a certain numerical error is tried to be achieved. With the proposed strategy, however, the numerical error is tried to be minimized for a certain number of grid cells or CPU costs. This approach is very useful when the number of grid points is limited, for example for large three dimensional and / or unsteady cases.

[Aftosmis and Berger (2002)] also tackled the issue of over-refinement. They proposed ^{2}log histograms of the error indicator such that the resulting histograms better resembled a Gaussian distribution. Consequently over refinement is less likely to occur when the refinement threshold is set to the mean plus a certain percentage of the standard deviation. However, this strategy still does not enable a strict control on the mesh size growth. In this paper their strategy is, therefore, extended such that the user can a priori set the number of grid cells on the adapted grid, thereby ensuring a strict control on the mesh size growth and high computational efficiency.

In most adaptation strategies unstructured tetrahedral or hybrid meshes are used because they lend themselves well for complex bodies and are well-suited for adaptation. However, besides a possible low accuracy in the boundary layer, tetrahedral grids have a large disadvantage when mesh adaptation is applied: the constraints of a high quality and efficient grid to capture one-dimensional features by means of mesh adaptation are difficult to satisfy simultaneously [Biswas and Strawn (1998)]. This paper, therefore, uses a body-conforming octree mesh generation [Tchon, Hirsch, and Schneiders (1997)]. It combines the favorable accuracy of hexahedral meshes inside boundary layers with the ease of grid generation of unstructured meshes. Such grids consist of a few body conformal cell layers, whereas the remainder of the domain is discretized with Cartesian cells. Anisotropic refinement is easily possible by dividing hexahedral cells in one or two directions.

This paper is organized as follows: in Section 2 and 3, respectively, the methodology and basic aspects of the proposed grid adaptation strategy are discussed. In Section 4 and Section 5, respectively, the computation of the refinement threshold and the number of grid adaptations are discussed. Finally, in Section 6 the performance of the adaptation strategy is discussed, while in Section 7 the conclusions are drawn.

2 Methodology

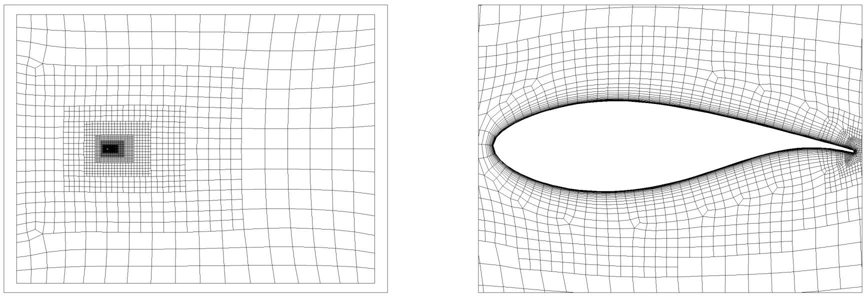
In this section details are given concerning the test cases, flow solver, error assessment and computation of grid independent values.

2.1 Test cases

For the development of the adaptation strategy, and in particular the strategy to compute the refinement threshold, we first designed the adaptation strategy for flows around wind turbine airfoils because these flows have our main interest. Thereafter the adaptation strategy has been tested for other flow problems to show its generic applicability beyond wind turbine airfoils (Section 6).

A 25% thick wind turbine airfoil (DU91-W2-250) with an angle of attack of 1 degree, a Reynolds number of $3.0 \cdot 10^6$ and a Mach number of 0.21 is used to discuss the development of the adaptation strategy. Transition from laminar to turbulent flow is defined according to the experiments by [Timmer (2001)].

2.2 Flow solver and initial grid generator



(a) Overview grid. (b) Close up showing the body conformal and Cartesian cells.

Figure 1: Grid layout as used for the grid refinement study DU91-W2-250 test case.

The flow solver (Hexstream [Numeca International (2003)]) is based on the Reynolds Averaged Navier-Stokes equations which describe conservation of mass, momentum and energy. The one-equation turbulence model of [Spalart and Allmaras (1992)] is used as a closure model. Space discretization is performed with a cell centered, conservative, finite volume scheme. The convective flux is discretized using a second order central scheme with Jameson type scalar artificial dissipation [Jameson, Schmidt, and Turkel (1981)]. Steady solutions are obtained with a time marching method using an explicit four stage Runge-Kutta scheme. Convergence is accelerated with a multigrid technique, see [Patel (2003); Patel, Léonard, Elsden, and Hirsch (2003)] and [Numeca International (2003)] for more information.

Because a Cartesian grid is used, it is not trivial to efficiently capture the boundary layer. A solution would be to deal with the triangular cells that arise at the interface as for example described in [Pasquim, B.M. and Mariani, V.C. (2008)]. However, as a matter of preference we use a hybrid grid that contains Cartesian as well as body conformal cells. In the grid generator [Numeca International (2002)] special attention is paid to obtain a smooth progression in cell size from the body conformal cells to the hexahedral cells, Fig. 1 shows the typical grid layout.

2.3 Assessment of the numerical accuracy

In order to assess the numerical accuracy of the grid adapted solutions it is important to compare them with a numerical benchmark as can also be concluded from

[Zhao, F., Zhu, S.-P., and Zhang, Z.-R. (2005)]. For numerical error assessment it is undesirable to compare grid adapted solutions with experiments because there also exists a modeling error which may bias interpretation of the results. Therefore, for all cases numerical benchmarks are obtained by means of a grid convergence study.

The relative error in lift and drag compared to the numerical benchmark values are used to judge numerical accuracy since these quantities are the main quantities of interest. Furthermore, the pressure distribution is monitored to detect possible cancellation of errors in these integral quantities.

2.4 Numerical benchmark

To construct the numerical benchmarks we use the least squares approach as advocated by [Eça and Hoekstra (2002); Eça, Vaz, Falcão de Campos, and Hoekstra (2004)]. First it is assumed that the error of a numerical prediction can be estimated by a Taylor series expansion:

$$\varepsilon_{\phi_i} = \phi_i - \phi_{\text{benchmark}} = \sum_{j=1}^{n} \alpha_j h_i^{p_j}, \tag{1}$$

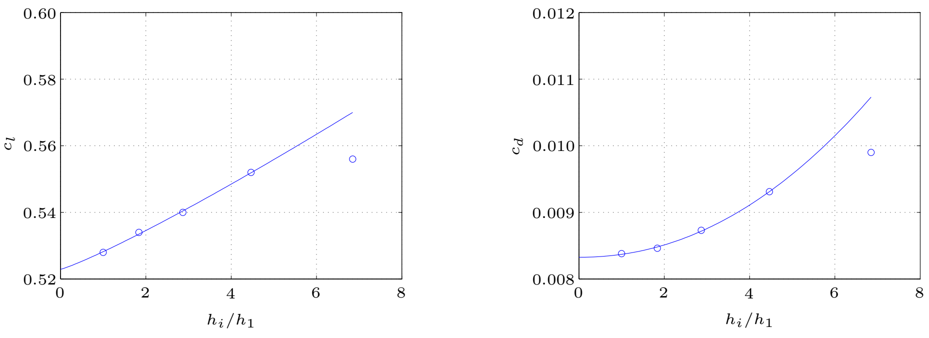
where ϕ_i is the numerical solution of any local or integral quantity on a given grid i, $\phi_{\text{benchmark}}$ is the (numerical) benchmark solution, α_j are constants, h_i is a parameter related to grid spacing, p_j is the observed order of accuracy and n is the order of the expansion. When the numerical predictions for a sequence of refined grids are in the asymptotic range, only the leading term is retained from Eq. 1, i.e.:

$$\phi_i - \phi_{\text{benchmark}} = \alpha h_i^p. \tag{2}$$

Eq. 2 requires at least three numerical predictions to solve for the three unknowns: $\phi_{\text{benchmark}}$, α and p. From Eça and Hoekstra (2002), the least squares approach is based on minimizing the function:

$$S(\phi_{\text{benchmark}}, \alpha, p) = \sqrt{\sum_{i=1}^{n_g} \left(\phi_i - \left(\phi_{\text{benchmark}} + \alpha h_i^p\right)\right)^2}, \tag{3}$$

where n_g is the number of grids available. By setting the partial derivatives of S with respect to $\phi_{\text{benchmark}}$, α and p to zero, $\phi_{\text{benchmark}}$, α and p can be computed.



(a) Lift coefficient. (b) Drag coefficient.

Figure 2: Lift and drag coefficient versus $\frac{h_i}{h_1}$ for the DU91-W2-250 test case. Open circles are results from the benchmark study whereas the solid line is the least squares approximation.

In Fig. 2 the numerical results of the grid convergence study for the DU91-W2-250 test case are shown. The solid lines show the approximation to Eq. 2 in the least squares sense, whereas the open circles are the results on the different grids. In Fig. 2 the parameter relating to cell size, h_i, is made dimensionless by dividing it by h_1, which corresponds to the finest grid. Hence, going from right to left on the horizontal axis the number of mesh cells increases. For $\frac{h_i}{h_1} = 0$ the benchmark values are obtained. The coarsest grid contains 33k cells whereas the finest grid contains 389k cells.

The height of the cells adjacent to the surface (y_0) is such that for these cells:

$$y_+ = \frac{y_0 u_\tau}{\nu} < 3, \tag{4}$$

with ν the kinematic viscosity and u_τ the friction velocity defined as:

$$u_\tau = \sqrt{\frac{\tau}{\rho}} = u_\infty \sqrt{\frac{c_f}{2}}, \tag{5}$$

where τ is the skin friction, ρ is the density, c_f is the skin friction coefficient and u_∞ the undisturbed velocity. With $y_+ < 3$ the first grid point is in the viscous sub layer which is required to have a well resolved boundary layer. The value for y_0 is the same for all computations that are performed for the DU91-W2-250 test case.

In order to construct the benchmark values, the numerical prediction on the coarsest grid is not taken into account because it is obviously not in the asymptotic range. The benchmark values are obtained for $\frac{h_i}{h_1} = 0$ and are 0.523 for the lift coefficient and 0.00833 for the drag coefficient.

3 Basic aspects of an adaptation strategy

In this section the basic aspects of the adaptation strategy are discussed. These are: 1) grid quality, 2) error indicator and 3) iterative convergence on intermediate adapted grids. We call these aspects the basic aspects of an adaptation strategy because they are well established. In Section 4 and Section 5 it is discussed how to compute the refinement threshold and to set the number of grid adaptations.

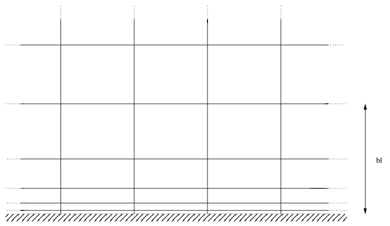
3.1 Grid quality

Other than when meshless methods are used, see e.g. [Orsini, P., Power, H., and Morvan, H. (2008)], [Mohammadi, M.H. (2008)], [Arefmanesh, A., Najafi, M., and Abdi, H. (2008)], grid quality after adaptation plays an important role because very low quality cells can lead to low numerical accuracy. Grid quality after adaptation is ensured with the following rules:

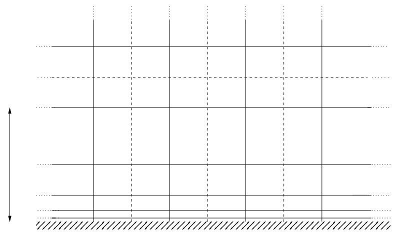
- do not allow more than 1 hanging node per cell interface;
- do not allow hanging nodes for the body conformal cells;
- do not allow neighboring cell sizes to differ more than a factor 2;
- force a smooth progression in cell sizes between the body conformal and Cartesian cells.

The first till the third rule are satisfied by propagation of cell refinement, i.e. when one of these rules is violated additional cells are refined such that the grid satisfies the quality rules. When boundary faces on the solid are refined additional vertices are added on the solid. These vertices are then reprojected on the surface to avoid unwanted flow discontinuities due to insufficient geometry resolution compared to mesh resolution, see also [Patel (2003); Patel, Léonard, Elsden, and Hirsch (2003)].

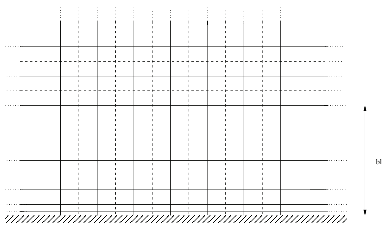
The fourth rule is satisfied by rebuilding the body conformal cell layer. First, the Cartesian cells are refined according to the refinement flags. Hereafter the body conformal cell layer is rebuild such that a smooth progression in cell size is achieved, see Fig. 3 for an illustration. Fig. 3(a) to Fig. 3(c) show refinement of the Cartesian cells and propagation of cell refinement (hanging nodes are not allowed for the body conformal cells). Fig. 3(d) shows how the body conformal cell layer is rebuild: with y_0 and the stretching ratio kept constant the body conformal cells increase in size from the wall, until the remaining part can be divided in equally sized body conformal cells.



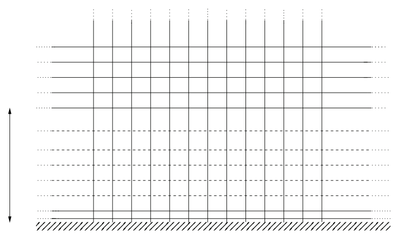
(a) Initial grid.



(b) Fictitious grid after one adaptation, body conformal cell layer not rebuild.



(c) Fictitious grid after two adaptations, body conformal cell layer not rebuild.



(d) Grid after two adaptations, layer of body conformal cells rebuild.

Figure 3: Illustration Cartesian refinement and the required refinement of the body conformal cells to ensure a smooth progression in cell size at connection body conformal and Cartesian cells. The grids shown in (b) and (c) are fictitious because the body conformal cell layer is not rebuild.

3.2 Error indicator

The error indicator serves to indicate which regions need to be refined to decrease the discretization error and to increase the numerical accuracy. Adaptation based on flow features assumes that large discretization errors occurs in areas with large gradients. For subsonic flows this is commonly assumed to be a valid assumption. However for supersonic flows one may need to rely on adaptation based on adjoints as for example discussed in Dwight (2006).

In the literature a wealth of different error indicators is available, see for example [Biswas and Strawn (1994); Cavallo and Baker (2000); Coirier and Powell (1995); Marcum (1996); Nithiarasu and Zienkiewicz (2000); Oh, Kim, and Kwon (2003);

Pirzadeh (2000); Rausch, Batina, and Yang (1992); Walsh and Zingg (1997); Walsh and Zingg (2001); Wang and Chen (2002); Warren, Anderson, James, and Krist (1991)]. Roughly these error indicators differ in two aspects: 1) the flow parameter(s) used, e.g. pressure and 2) the type of difference used, e.g. a multiplied difference. The flow parameter determines which regions will be refined, i.e. an error indicator based on the pressure will result in refinement in regions with a large pressure gradient. The type of difference controls the level of global refinement, i.e. a multiplied difference will result in more global refinement than a divided difference.

In general the error indicator is based on the pressure (or density) and/or velocity (Mach number). Sometimes an additional parameter is used to capture highly rotational regions of the flow, see for example [Pirzadeh (2000)]. A multiplied difference is commonly used to avoid too much local refinement. We therefore use the following error indicator:

$$\varepsilon_{i+\frac{1}{2}} = \overline{(||\nabla p||)}\big|_{i+\frac{1}{2}} \cdot h^x, \tag{6}$$

where the mean of $||\nabla p||$ is evaluated at cell interface $i+\frac{1}{2}$ and h is defined as:

$$h = 2\min\left(h_{\text{face}_{i+1/2},\text{center}_i}, h_{\text{face}_{i+1/2},\text{center}_{i+1}}\right). \tag{7}$$

The error indicator given in Eq. 6 has the following advantages over a regular multiplied difference: 1) multiplying with h as given in Eq. 7 rather than $h = h_{\text{center}_i,\text{center}_{i+1}}$ results in less chaotic grid refinement around cell interfaces neighboring cells of different size and 2) the mean of the norm of the pressure gradient results in more isotropic refinement which, for our spatial discretization [Patel, Léonard, Elsden, and Hirsch (2003); Patel (2003)], drastically improves numerical accuracy. Preliminary investigations revealed that $x = 1.5$ leads to high numerical accuracy. Typically values of 1 to 2 are used throughout the literature, see e.g. [Coirier and Powell (1995); Wang and Chen (2002)].

Note that the error indicator given in Eq. 6 does not capture boundary layers. However, because the body conformal cells 'follow' the refinement of the Cartesian cells (Section 3.1), refinement of the body conformal cells does occur. So far our results did not indicate that we should have used an additional flow parameter to refine boundary layers.

Table 1: Influence convergence criterium set for intermediate adapted grids on total CPU costs DU91-W2-250 test case.

conv. crit.	ε_{c_l}	ε_{c_d}	# cells	CPU costs
-3.5	3.8%	7.6%	49.9k	348
-5.0	3.8%	7.6%	49.7k	345
-6.0	3.8%	7.6%	49.6k	555

3.3 Iterative convergence on intermediate adapted grids

It is well known that the level of iterative convergence on the initial grid should be sufficient to avoid adaptation to transient features. Sometimes it is stated that the iterative convergence on the intermediate adapted grids should also be high, see for example [Cavallo and Baker (2000)].

However, since the solution does not change much after adaptation it is less likely to adapt to transient features when the level of iterative convergence is low on the intermediate adapted grids. Low iterative convergence on the intermediate adapted grids can imply significant savings in CPU costs as the intermediate adapted grids are much denser than the initial grid. Iterative convergence on the final adapted grid should again be sufficient to warrant a small iteration error.

To investigate if a low iterative convergence on the intermediate adapted grids has a positive result on the total CPU costs the same case is computed for different convergence criteria on the intermediate grids. We have used 3 different convergence criteria: a drop in nonlinear residual of 3.5, 5 and 6 orders in magnitude compared to the initial solution. Roughly speaking these convergence criteria result in a numerical accuracy for lift and drag on the intermediate adapted grids of, respectively, 1, 2 and 3 digits. The convergence criteria set for the initial and final grid is such to warrant a negligible iteration error.

From Tab. 1 it becomes clear that a too strict threshold (a drop in nonlinear residual of 6 orders compared to its initial value) for the iterative convergence on the intermediate adapted grids almost doubles the CPU costs, however does not increase the numerical accuracy. Results obtained for other cases revealed similar results.

4 Computation refinement threshold

The refinement threshold determines which and how many grid cells will be refined. It, therefore, plays a crucial role in the computational efficiency of the adaptation strategy. Common strategies to compute the refinement and coarsening thresholds found in literature [Biswas and Strawn (1994); Cavallo and Baker (2000); Coirier and Powell (1995); Ham, Lien, and Strong (2002); Mavriplis (2000); Warren, Anderson, James, and Krist (1991)] are:

- a priori set fixed thresholds;
- thresholds set by the user by trial and error;
- thresholds computed based on the assumption of a Gaussian distribution of the error indicator distribution.

Unfortunately, these strategies all suffer from serious disadvantages. A priori settings for the thresholds can easily lead to over or under refinement because the error indicator distribution is not known beforehand. Setting the refinement threshold by trial and error is inefficient since the adaptation process should be fully automated. Finally, the strategy that is based on the assumption of a Gaussian distribution (also referred to as Kallinderis's method [Kallinderis and Baron (1989)]) is automated with respect to computation of the refinement threshold. However, especially for 3-dimensional flows, the assumption of the Gaussian distribution is highly inaccurate [Aftosmis and Berger (2002)] often leading to an excessive mesh growth, see for example [Bijl, Zuijlen, and Mameren (2005)] and [Wang and Chen (2002)].

In this paper, therefore, a novel strategy is developed to compute the refinement threshold based on the user desired number of grid cells and user desired number of grid adaptations on the final adapted grid. This is a somewhat unconventional approach because commonly the refinement threshold is set such that a certain numerical error is tried to be achieved. With the proposed strategy, however, the numerical error is tried to be minimized for a certain number of grid cells or CPU costs. This approach is very useful when the number of grid points is limited, for example for large three dimensional and / or unsteady cases. The proposed strategy is an extension of the strategy proposed by [Aftosmis and Berger (2002)]. They proposed ${}^{2}\log$ histograms of the error indicator such that the resulting histograms better resemble a Gaussian distribution. Consequently over refinement is less likely to occur when the refinement threshold is set to the mean plus a certain percentage of the standard deviation.

4.1 Basic idea

The basic idea of the proposed strategy to compute the refinement threshold is that the histograms of the ${}^{2}\log$ values of the error indicator distribution corresponding to the solutions on the next adapted grids can be extrapolated from the initial grid. When it is assumed that: 1) the solution does not change with mesh refinement and 2) there is a fixed amount of cell *interfaces* that are added per refinement flag, the ${}^2\log$ histograms can be estimated because it is known how much the error indicator of a certain cell interface will shift through the histogram with mesh refinement. Therefore, imagine a certain error indicator that has a value z before mesh refinement, with z larger than the refinement threshold. After adaptation both cells neighboring this particular cell interface are halved in size in the corresponding direction. For an error indicator that is proportional to h^y it equals $z \cdot \left(\frac{1}{2}\right)^y$ after mesh adaptation. However, in terms of ${}^2\log$ we have:

$$ {}^2\log\left(z \cdot \left(\frac{1}{2}\right)^y\right) = {}^2\log z - {}^2\log\left(2^y\right) = {}^2\log z - y. \tag{8}$$

Hence, this particular error indicator has shifted y units through the ${}^2\log$ histogram. When it is also assumed that there is a fixed amount of mesh *cells* that are added per refinement flag, the number of mesh cells on the final adapted grid can be extracted from the error indicator distribution on the initial grid. Hence, the following parameters should be approximately known or tuned:

1. proportionality of the error indicator to h (y in Eq. 8);
2. number of cell *interfaces* added per refinement flag;
3. number of *cells* added per refinement flag.

Note that the first two parameters deal with histogram extrapolation, whereas the third parameter deals with the estimation of the total number of mesh cells based on the histogram extrapolation. In the next section these parameters are discussed in more detail.

4.2 Tuning parameters

4.2.1 Proportionality of the error indicator with respect to cell size

Since the solution changes with mesh adaptation, it is not exactly known how many units the error indicator will shift through the histogram. To obtain a starting point for the proportionality of the error indicator to h we assume one dimension and a solution that does not change with grid adaptation. Eq. 6 then becomes:

$$\begin{aligned} \overline{(||\nabla q||)}|_{i+\frac{1}{2}} h^{1.5} &= \overline{(||q'||)}|_{i+\frac{1}{2}} h^{1.5} \\ &= \frac{1}{2}\left(||q'_{i+1}|| + ||q'_i||\right) h^{1.5}. \end{aligned} \tag{9}$$

A Taylor series expansion of q around i is:

$$q_{i+1} = q_i + q_i' h + \frac{1}{2} q_i'' h^2 + O(h^3). \tag{10}$$

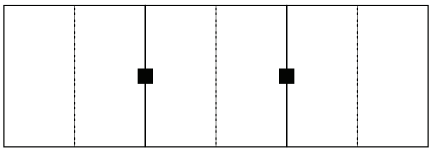
Substitution of Eq. 10 into Eq. 9 yields:

$$\overline{(||\nabla q_i||)}|_{i+\frac{1}{2}} = \left(||q_i' + q_i'' h|| + ||q_i'||\right) h^{1.5} + O(h^{3.5}). \tag{11}$$

From Eq. 11 it can be concluded that the finite difference is at least proportional to $h^{1.5}$ and at most to $h^{2.5}$. These bounds form the starting point to tune the histogram estimation in Section 4.3.

4.2.2 Number of cell interfaces added per refinement flag

Based on the ${}^2\log$ values of the error indicators it is unknown how many cell interfaces are added during adaptation, see Fig. 4 for two academic examples. In Fig. 4(a) two refinement flags result in 3 new cell interfaces, whereas in Fig. 4(b) four refinement flags result in 16 new cell interfaces. Hence, in the first situation 1.5 cell interfaces are added per refinement flag, whereas in the second situation 4 cell interfaces are added per refinement flag. Hence, we expect this ratio somewhere between 1 and 4 for two dimensional computations. In the following this ratio is referred to as the FACESADDEDPERFLAG ratio.


(a) Two refinement flags (filled squares) result in 3 extra cell interfaces (dashed lines).

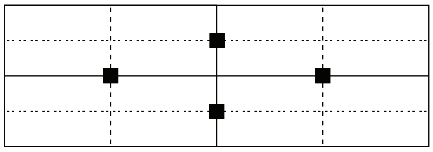

(b) Four refinement flags (filled squares) result in 16 extra cell interfaces (dashed lines).

Figure 4: The ratio of cell interfaces and cell added per refinement flag is not constant.

4.2.3 Number of cells added per refinement flag

Finally, we are not interested in estimating any histograms, but in the number of mesh *cells* on the final adapted grid. Consider again Fig. 4. In Fig. 4(a) two refinement flags result in 3 extra cells, whereas in Fig. 4(b) four refinement flags result in 12 extra cells. Hence, in the first situation 1.5 cells are added per refinement flag, whereas in the second situation 3 cells are added per refinement flag.

Table 2: Number of cell interfaces that have an error indicator larger than the refinement threshold for different assumptions of the proportionality of the error indicator (h^y) for the DU91-W2-250 test case.

		adap 1		adap 2		adap 3		adap 4	
y	FACESADDEDPERFLAG	est.	real	est.	real	est.	real	est.	real
1	2.138	8270	8271	5929	4742	3137	466	0	134
1.5	3.530	8267	8271	4000	4742	176	466	0	134
2	6.375	8268	8271	1869	4742	0	466	0	134
2.5	12.77	8275	8271	652	4742	0	466	0	134

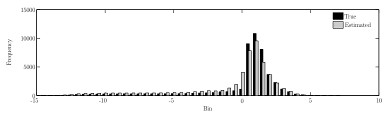
4.3 Tuning the histogram estimation

In this section the histogram extrapolation is tuned for hexahedral grids and the DU91-W2-250 test case. As discussed in Section 4.2.1 two parameters need to be tuned to estimate the error indicator distributions corresponding to the next adapted grids: 1) y (the amount of units the error indicator will shift through the histogram with refinement) and 2) the FACESADDEDPERFLAG ratio (how many cell interfaces are added for each cell interface flagged for refinement).

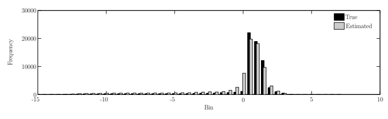
These two parameters are simultaneously tuned, which is accomplished by comparing extrapolated histograms with real histograms. In Section 4.2.1 it has been argued that y is bounded between 1.5 and 2.5. These values for y are therefore used as a starting point to tune the histogram estimation. The FACESADDEDPERFLAG ratio is then adjusted such that the number of estimated error indicators *larger* than the refinement threshold matches the real number of error indicators larger than the refinement threshold. The FACESADDEDPERFLAG ratio is computed for the extrapolation from the initial grid to the first adapted grid and then assumed to be constant.

Tab. 2 compares the *estimated* number of cell *interfaces* with the *true* number of cell *interfaces* having an error indicator *larger* than the refinement threshold versus the number of mesh adaptations. Note that the estimation of the first adaptation is indeed very accurate, since the FACESADDEDPERFLAG ratio is tuned for the first adaptation step. Since the number of cell interfaces is underestimated for $1.5 < y < 2.5$, a value of 1 for y has also been tried.

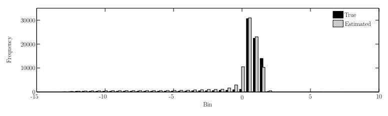
Tab. 2 shows that the extrapolation of histograms is most accurate for an error indicator that is assumed to be proportional to $h^{1.5}$ (Implying that the solution does change with mesh refinement) and a FACESADDEDPERFLAG ratio of 3.53. A FACESADDEDPERFLAG ratio of 3.53 seems large, however is not because refinement tends to be rather isotropic due to the error indicator used (see also Fig. 4(b)).



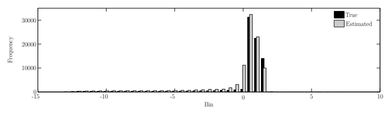
(a) Histogram after 1 adaptation: 25.4k cells.



(b) Histogram after 2 adaptations: 38.6k cells.



(c) Histogram after 3 adaptations: 46.6k cells.



(d) Histogram after 4 adaptations: 47.5k cells.

Figure 5: ${}^{2}\log$ (error indicator) histogram of adapted grids DU91-W2-250 test case: the frequency is given on the vertical axis whereas the ${}^{2}\log$ (error indicator) value is given on the horizontal axis. Error indicators that have a value larger than the refinement threshold are assumed to shift 1.5 units to the left with refinement. The refinement threshold is 1.5.

In Fig. 5 the histogram estimation is compared to the actual error indicator distribution. The refinement threshold in this case is equal to 1.5. Hence, all error indicators that have a value larger than 1.5 are flagged for refinement. Note that the part of the histogram where error indicators have a value larger than the refinement threshold is estimated more accurately than the error indicators in the bins 0 and smaller. This is because according to the estimations the frequencies of bins smaller than 0 cannot increase, while this can happen in reality due to a different proportionality of the error indicator, changes in the solution and additional refinement to satisfy the quality rules.

Note, when y is fixed, the tuning can be made fully automated, because it is possible to adjust the value for FACESADDEDPERFLAG based on the extrapolated and true histograms. It is recommended to use a value for y of 1.5 and an initial value for FACESADDEDPERFLAG of 3.53.

4.4 Tuning the number of mesh cells added per refinement flag

Tab. 3 displays the increase in mesh cells for the different grid adaptations for the DU91-W2-250 test case. From this table it can be concluded that the ratio of cells added per refinement flag (‘total / # flags’) does not differ much for the first 3 adaptations, i.e. it is between 1.50 and 1.68. After the third adaptation there is larger variation. However, most grid cells are added during the first three grid adaptations so we assume this ratio to be equal to 1.6.

Table 3: Increase in mesh cells for the DU91-W2-250 case for a certain refinement threshold.

	$a_0 \rightarrow a_1$	$a_1 \rightarrow a_2$	$a_2 \rightarrow a_3$	$a_3 \rightarrow a_4$	$a_4 \rightarrow a_5$	$a_0 \rightarrow a_5$
total	12280	13199	7982	931	277	34669
in bound. layer grid	4224	1540	506	88	0	6358
Cartesian	8056	11659	7476	843	277	28311
satisfy quality rules	139	300	231	118	73	861
# flags	8191	8271	4742	466	137	21807
total / # flags	1.50	1.60	1.68	2.00	2.07	1.59

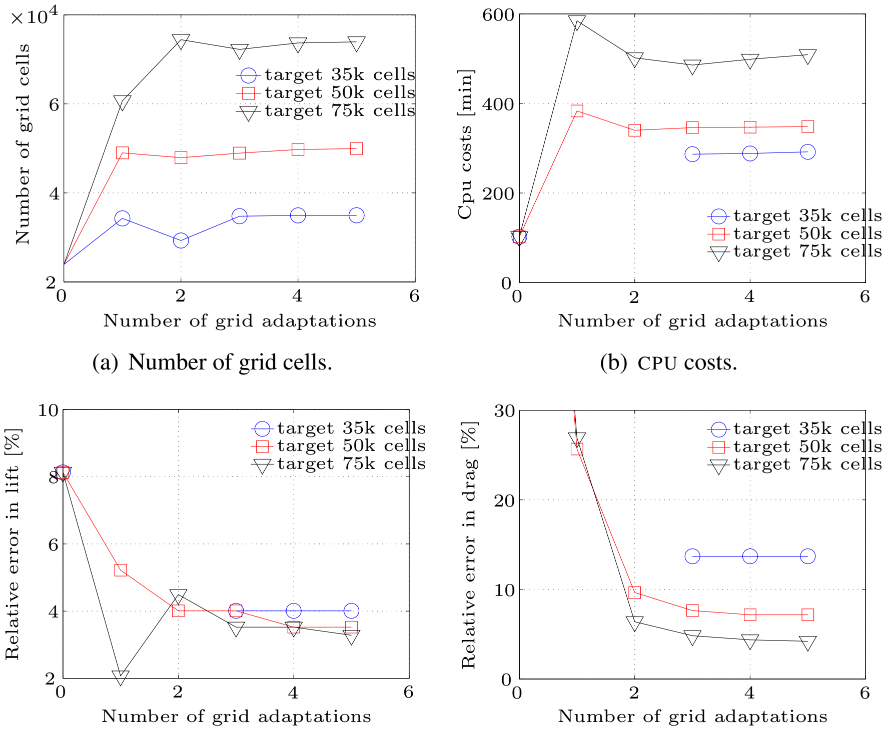
4.5 Computation of the refinement threshold

In the previous sections it has been shown that the histograms corresponding to the solutions on the adapted grids can be extrapolated based on the solution on the initial grid. This has been accomplished by assuming that the error indicator is proportional to $h^{1.5}$ and tuning the estimated number of cell interfaces added per refinement flag. The number of grid cells on next adapted grids is estimated by tuning the estimated number of grid cells added per refinement flag. Hence, based on the solution on the initial grid and the number of grid adaptations that will be performed it is possible to estimate the number of grid cells on the final adapted grid. This can also be reversed which makes it possible to compute the refinement threshold based on a user desired number of grid cells and user desired number of grid adaptations on the final adapted grid. How to determine the number of grid adaptations is topic of the next section.

5 Number of grid adaptations

It is commonly believed that an equidistribution or a strict upper bound for the error indicator distribution results in the highest numerical accuracy for a given number of mesh cells, see for example [Aftosmis and Berger (2002); Hall and Zingg (1995)]. However, to obtain an equidistribution or an upper bound, many adaptations may be required which can be computationally expensive. In this section it is investigated if a strict upper bound is advantageous for the computational efficiency.

Fig. 6(a) shows the number of grid cells as function of the number of adaptations for three different user desired settings for the number of grid cells. Because the number of grid cells is set to a certain value it should be (approximately) the same regardless the number of grid adaptations. Fig. 6(a) shows that this is the case, except for case: 'target 35k cells, 2 adaptations' and case: 'target 75k cells, 1 adaptation'. In the first case the target is not reached because the solver diverged after 1 adaptation, whereas in the latter case the target is not reached because the initial grid was too coarse[1]. Hence, we can already conclude that the strategy to compute the refinement threshold results in a grid that contains approximately the number of grid cells as desired by the user.



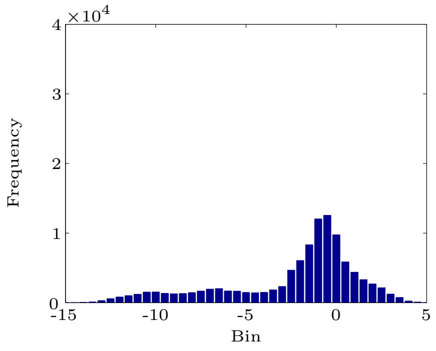
(a) Number of grid cells. (b) CPU costs.

(c) Relative error in lift. (d) Relative error in drag.

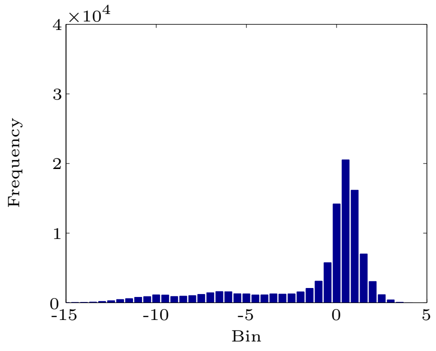
Figure 6: Number of grid cells, CPU costs and relative error in lift and drag as function of the number grid adaptations DU91-W2-250 case.

Fig. 6(b) shows the CPU costs as function of the number of grid cells. Remarkably the CPU costs are rather insensitive to the number of grid adaptations. This is because the cost driver for these cases is to drive the nonlinear residual on the final

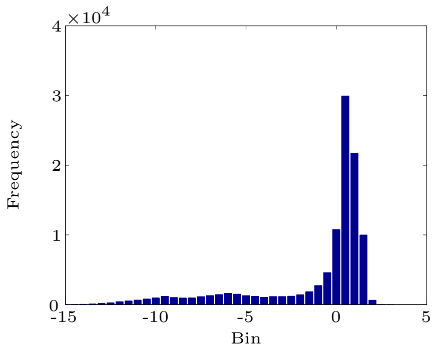
[1] The maximum number of grid cells after 1 adaptation is less than 4 times the number of grid cells of the initial grid because of different refinement strategies for the body conformal and Cartesian cells.



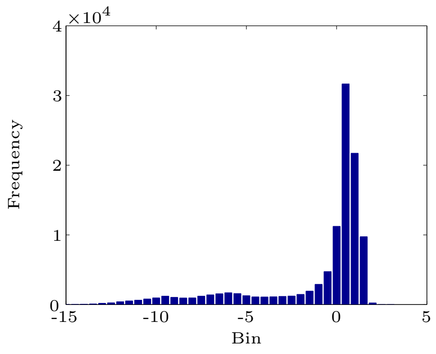
(a) target 50k cells, 1 adaption (51.9k cells, ref. thres. is 1.04).



(b) target 50k cells, 2 adaptions (48.7k cells, ref. thres. is 1.31).



(c) target 50k cells, 3 adaptions (52.6k cells, ref. thres. is 1.56).



(d) target 50k cells, 4 adaptions (53.8k cells, ref. thres. is 1.56).

Figure 7: Final histograms of the error indicator distribution for different settings DU91-W2-250 case. True number of grid cells and refinement threshold are given between curley brackets.

grid below a certain threshold. It is, furthermore, striking that the CPU costs are largest for the cases when only 1 grid adaptation is made. Apparently the cases that use more grid adaptations result in a faster convergence on the final grid. This may well be explained by a better initial solution on the final adapted grid because of more intermediate iterations.

Fig. 6(c) and Fig. 6(d), furthermore, respectively show the relative error in lift and drag. First of all these figures show that the adaptation strategy is consistent, i.e. the larger the target set for the number of grid points, the higher the numerical accuracy. The reduction in error is impressive when the number of grid adaptations is increased from 1 to 3 grid adaptations, whereas the reduction in error going from 3 to 5 grid adaptations is marginal. The low relative error in lift against a rather large (more than 100 percent) error in drag for a target of 75k cells and 1 grid adaptation is because the pressure distribution for this case is rather accurate except for a wrong prediction of the stagnation pressure.

Finally, in Fig. 7(a) to Fig. 7(d) the error indicator distributions are shown for the same target of grid cells (50k), however for different number of grid adaptations. To meet the target set for the number of grid cells the refinement threshold is different for these cases. From these figures it follows that the larger the number of grid adaptations, the stricter the upper bound.

By comparison of Fig. 6(c) and Fig. 6(d) with Fig. 7(a) to Fig. 7(d) we can conclude that the numerical accuracy is highest when a strict upper bound is created. Furthermore, because the CPU costs are virtually independent on the number of grid adaptations, we can conclude that also the computational efficiency is highest when a strict upper bound is created. Hence, to optimize the results obtained with the adaptation strategy the number of grid adaptations should be chosen such that a strict upper bound is created for the user desired number of grid cells.

6 Performance adaptation strategy

In this section the computational efficiency of the adaptation strategy is addressed: how does it compare with results that have been obtained on manually optimized grids. The adaptation strategy is therefore applied to three different flow problems: 1) the DU91-W2-250 case, 2) a transonic viscous flow over a NACA 0012 airfoil and 3) a three dimensional inviscid flow over an AGARD wing.

For these cases we have used the following settings: 1) the number of grid cells is a priori defined 2) the number of grid adaptations is such that (approximately) a strict upper bound is created and 3) the iterative convergence criterion on the intermediate adapted grids is set to -3.5. Once more, our approach is rather unconventional because the number of grid cells is set and not the numerical accuracy. This approach is useful when the number of allowed grid cells is limited, for example for large three dimensional or unsteady flows. All cases have the same iterative convergence on the final (adapted) grid and are obtained on a Pentium 4 2.6 GHz processor. Finally, we strive for engineering order of accuracies (approximately 5 percent numerical error).

6.1 Subsonic flow over a wind turbine airfoil

This test case is the DU91-W2-250 case used previously. Although the difference in final histogram is minimal (see Fig. 7), we have applied 5 grid adaptations to create a strict upper bound for the error indicator distribution. In Tab. 4 two adaptation cases are compared with 2 grids that have been used for the grid convergence study.

Table 4: Computational efficiency of the adaptation strategy DU91-W2-250 test case.

case	ε_{c_l}	ε_{c_d}	CPU costs	# mesh cells
initial grid	8.1%	106%	35.5 min	23.9k
5 adapt., target 50k cells	3.5%	7.2%	348 min	49.9k
5 adapt., target 75k cells	3.3%	4.2%	509 min	73.9k
man. opt. 1	5.3%	10%	514 min	50.6k
man. opt. 2	3.0%	4.7%	917 min	78.9k

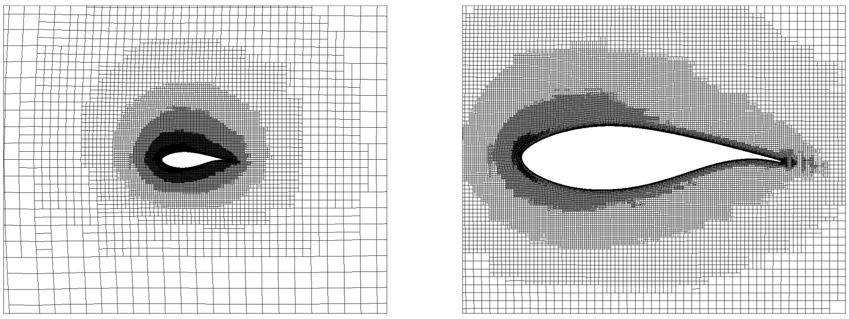

(a) Overview. (b) Close up.

Figure 8: Adapted grid DU91-W2-250 test case: ‘5 adap., target 50k cells’.

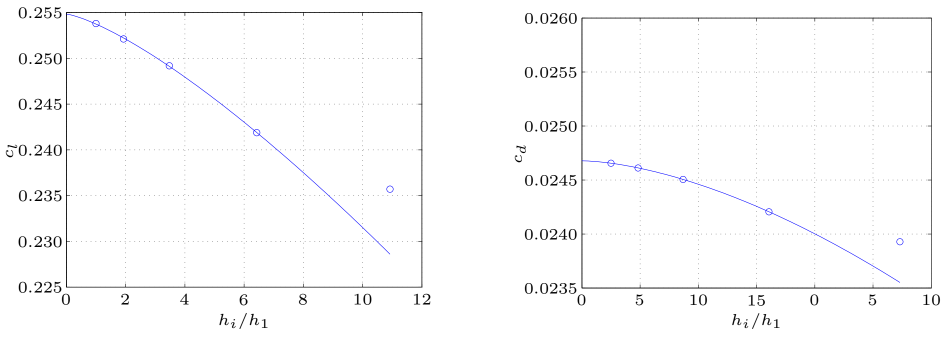
From Tab. 4 it follows that the adaptation strategy performs very well compared with the results obtained on the manually optimized grids (which required a substantial amount of man hours). For the same order of numerical accuracy, the computing time required is much lower (approximately 30 to 40 percent) when the adaptation strategy is employed. Fig. 8 shows the adapted grid for case ‘5 adapt., target 50k cells’.

6.2 *Viscous transonic flow over NACA0012 airfoil*

The NACA test case concerns viscous flow with a Reynolds number of $1.0 \cdot 10^7$ and a Mach number of 0.80 over a NACA0012 airfoil at an angle of attack of 1.25 degrees. This case has a supersonic region on the lower and upper side of the airfoil.

6.2.1 Numerical benchmark

In the same way as described in Section 2.4 the grid convergence study is performed, Fig. 9 shows the results. The coarsest grid contains 98k cells and the finest 1073k. For the lift coefficient a value of 0.25482 is found and for the drag coefficient a value of 0.024678.



(a) Lift coefficient. (b) Drag coefficient.

Figure 9: Lift and drag coefficient versus $\frac{h_i}{h_1}$ NACA test case. Open circles are results from the benchmark study whereas the solid line is the least squares approximation.

6.2.2 Results and discussion

As already discussed in Section 3.2 solution based grid adaptation should be applied with great care when supersonic / transonic flows are computed. This is because the sharpness of the shock tends to increase with adaptation. Hence, although a multiplied difference is used, the error indicator in the shock will hardly decrease with grid adaptation. We, therefore, do not try to achieve a strict upper bound for the error indicator distribution, however limit the number of grid adaptations to 3 to avoid over refinement of the shocks.

In Tab. 5 a comparison is made between the computational cost and numerical accuracy obtained on the manually optimized grids and with the adaptation strategy. First note that the numerical accuracy obtained on the initial grid is rather high. Fig. 10, however, shows that this is caused by cancellation of errors, i.e. the shock is not captured on the initial grid.

Tab. 5 shows that also for the NACA case the proposed strategy to compute the refinement threshold works very well, i.e. the number of grid cells on the final adapted grid is almost equal to the target set.

Table 5: Efficiency adaptation strategy applied to the NACA test case.

case	ε_{c_l}	ε_{c_d}	# cells	CPU costs
initial grid	5.72%	1.65%	17.7k	68.3 min
3 adap., target 100k cells	4.85%	1.07%	97.9k	538 min
3 adap., target 150k cells	2.65%	0.268%	148k	865 min
man. opt. 1	5.08%	1.90%	167k	433 min
man. opt. 2	2.22%	0.688%	309k	690 min

From Tab. 5 it, furthermore, follows that for the same numerical accuracy the adaptation strategy leads to slightly larger CPU cost than obtained on the manually optimized grids. However, for these cases the CPU costs to manually optimize the grids have been omitted. Hence, many hours of gridding have been saved. The adaptation strategy, therefore, also performs well for this supersonic case.

Finally, Fig. 11 shows the adapted grid for case '3 adap., target 150k cells'. The 'sharp' edges from finer to coarser cells are caused by the layout of the initial grid.

6.3 Euler subsonic flow over AGARD *wing*

The AGARD test case concerns three dimensional inviscid flow with a Mach number of 0.80 over an AGARD 445.6 wing at an angle of attack 5 degrees. The AGARD wing consists of a symmetric NACA 65A004 airfoil with a sweep angle of 45° and a taper ratio of 0.66.

6.3.1 Numerical benchmark

In the same fashion as described in Section 2.4 the grid convergence study is performed. Fig. 12 shows the results for the Euler flow over the AGARD wing. The coarsest grid contains 151k cells and the finest one 5480k cells. For the lift coefficient a value of 0.01266 and for the drag coefficient a value of $7.455 \cdot 10^{-4}$ is found. Note that grid convergence is less smooth than for the previous cases. We believe this is caused by different grid quality from one grid to another. Due to the nature of our unstructured grid generator low orthogonal cells are formed around sharp edges.

6.3.2 Results and discussions

When the number of dimensions is increased from 2 to 3 the tuning parameters in the strategy to compute the refinement threshold cannot be kept constant, because a cell can now be subdivided into 8 children. In the same fashion as described in

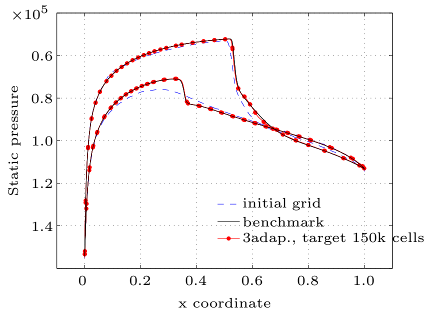
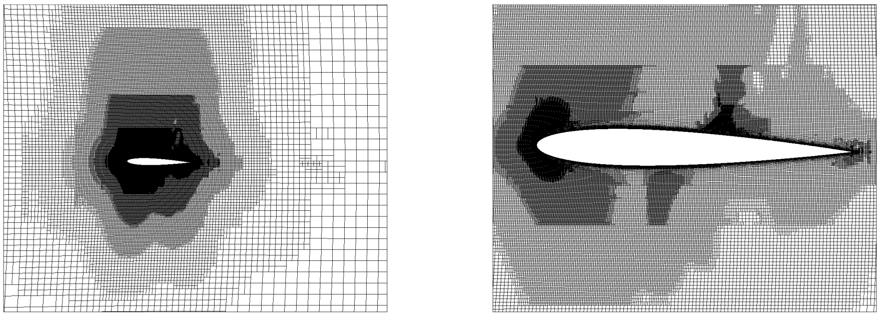
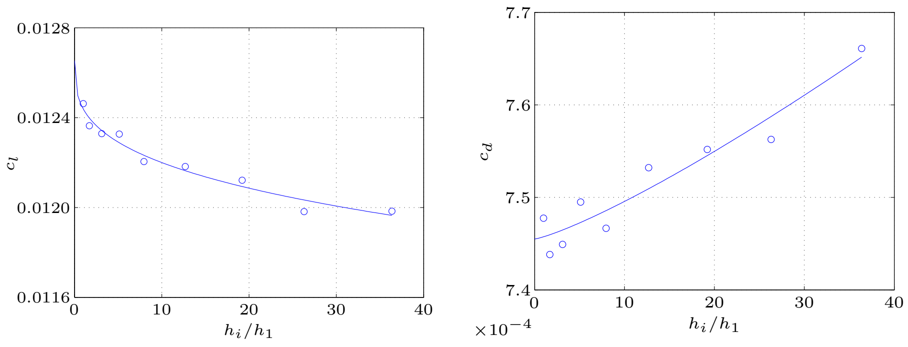


Figure 10: Pressure distribution NACA airfoil. Shock is not captured on initial grid.



(a) Overview. (b) Close up.

Figure 11: Adapted grid NACA case: '3 adap., target 150k cells'.



(a) Lift coefficient. (b) Drag coefficient.

Figure 12: Lift and drag coefficient versus $\frac{h_i}{h_1}$ AGARD case.

Section 4.3 these parameters are tuned to the following values: 'FacesAddedPerFlag ratio' = 40 and 'ratio of cells added per refinement flag' = 2.6. The proportionality of the error indicator with respect to cell size is not modified, because the same error indicator is used.

Although the difference is small, for the AGARD case and for a target number of grid cells of 100k and 200k, 1 or 2 grid adaptations yield the most strict upper bound, see Fig. 13. We believe this is caused by decreasing grid quality when several grid adaptations are performed. The proposed strategy to compute the refinement strategy, however, also works well in three dimensions as can be concluded from Fig. 14, i.e. for all settings for the number of grid adaptations the user desired target for the number of grid cells is achieved.

In Tab. 6 the computational efficiency of the adaptation strategy for the AGARD case is addressed. From this table it can be concluded that the adaptation strategy also performs well for this three dimensional case, i.e. for the same computational costs the numerical accuracy achieved with the adaptation strategy is higher. Furthermore, the gain in user comfort is significant: manually optimizing the grids for this test case has been a rather time consuming phase.

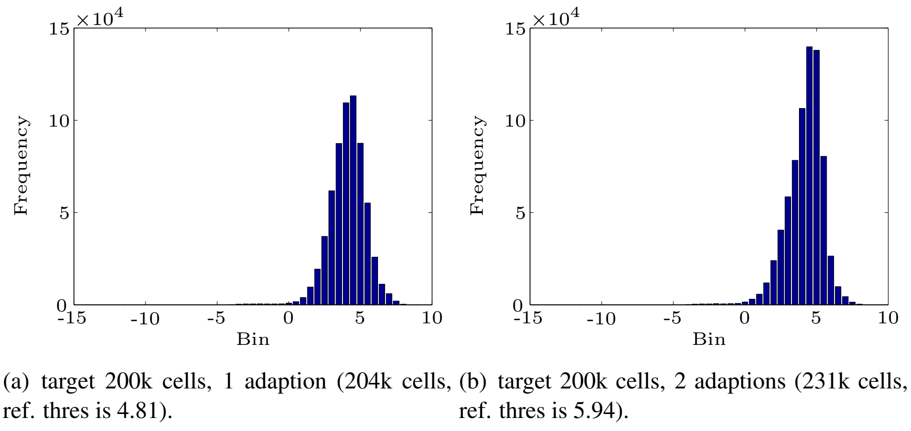
Table 6: Efficiency adaptation strategy applied to the AGARD case.

case	ε_{c_l}	ε_{c_d}	# cells	CPU costs
initial grid	11.8%	1.08%	57.0k	4.0 min
1 adap., target 100k cells	10.6%	1.49%	109k	16 min
1 adap., target 200k cells	5.57%	2.08%	204k	36 min
man. opt. 1	5.30%	16.2%	124k	35 min
man. opt. 2	5.30%	2.75%	151k	60 min

7 Conclusions

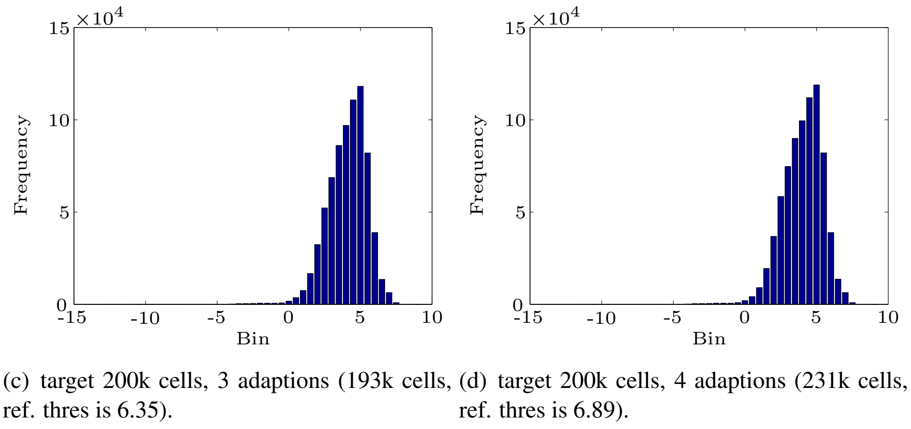
Goal of this paper is to enable numerically accurate and computationally efficient grid adaptation for flow around wind turbine airfoils. Key point is the development of a new strategy to compute the refinement threshold that is fully automated and enables a strict control on the mesh size growth.

In this paper it has been shown that the refinement threshold can be computed based on the user desired number of grid adaptations, the user desired number of grid cells and the error indicator distribution corresponding to the solution on the initial grid. Furthermore, it has been shown that the adaptation strategy enables computationally efficient and numerically accurate subsonic flow computations. Also for the



(a) target 200k cells, 1 adaption (204k cells, ref. thres is 4.81).

(b) target 200k cells, 2 adaptions (231k cells, ref. thres is 5.94).



(c) target 200k cells, 3 adaptions (193k cells, ref. thres is 6.35).

(d) target 200k cells, 4 adaptions (231k cells, ref. thres is 6.89).

Figure 13: Final histograms of the error indicator distribution for different settings AGARD case. The most strict upper bound is obtained when only 1 grid adaptation is applied. True number of grid cells and refinement threshold are given between curley brackets.

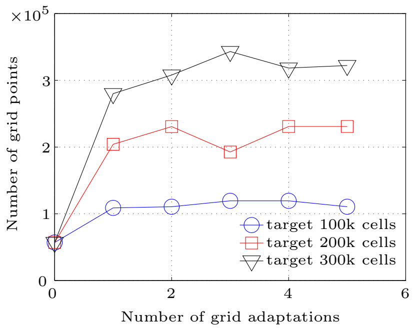


Figure 14: Number of grid cells as function of the number of grid adaptations AGARD case.

supersonic case, with the number of grid adaptations limited to avoid over refinement of the shock, the adaptation strategy has been shown to yield computationally efficient and numerically accurate flow computations.

Grid quality for three dimensional cases is a candidate for further improvement.

References

Aftosmis, M.; Berger, M. (2002): Multilevel error estimation and adaptive *h*-refinement for cartesian meshes with embedded boundaries. In 40th *AIAA Aerospace Sciences Meeting and Exhibit*, Reno NV. AIAA Paper 2002-0863.

Arefmanesh, A.; Najafi, M.; Abdi, H. (2008): Meshless local petrov-galerkin method with unity test function for non-isothermal fluid flow. *CMES: Computer Modeling in Engineering & Sciences*, vol. 25, no. 1, pp. 9–22.

Bijl, H.; Zuijlen, A. v.; Mameren, A. v. (2005): Validation of adaptive unstructured hexahedral mesh computations of flow around wind turbine airfoil. *International journal for numerical methods in fluids*, vol. 48, no. 9, pp. 929–945.

Biswas, R.; Strawn, R. (1994): A new procedure for dynamic adaption of three-dimensional unstructured grids. *Applied Numerical Mathematics*, vol. 13, pp. 437–452.

Biswas, R.; Strawn, R. (1998): Tetrahedral and hexahedral mesh adaptation for cfd problems. *Applied Numerical Mathematics*, vol. 26, pp. 135–151.

Cavallo, P.; Baker, T. (2000): Efficient delaunay-based solution adaptation for three-dimensional unstructured meshes. In *AIAA* 38th *Aerospace Sciences Meeting & Exhibit*, Reno, NV. AIAA Paper 2000-0809.

Chaviaropoulos, P.; Nikolau, I.; Aggelis, K.; Soerensen, N.; Johansen, J.; Hansen, M. e. a. (2003): Viscous and aeroelastic effects on wind turbine blades. the viscel project. part 1: 3d navier-stokes rotor simulations. *Wind Energy*, vol. 6, pp. 365–385.

Chaviaropoulos, P.; Nikolau, I.; Aggelis, K.; Soerensen, N.; Johansen, J.; Hansen, M. e. a. (2003): Viscous and aeroelastic effects on wind turbine blades. the viscel project. part 2: Aeroelastic stability investigations. *Wind Energy*, vol. 6, pp. 387–403.

Coirier, W.; Powell, K. (1995): An accuracy assessment of cartesian-mesh approaches for the euler equations. *Journal of Computational Physics*, vol. 117, pp. 121–131.

Dwight, R. (2006): *Efficiency Improvements of RANS-Based Analysis and Optimization using Implicit and Adjoint Methods on Unstructured Grids*. PhD thesis, School of Mathematics, University of Manchester, 2006.

Eça, L.; Hoekstra, M. (2002): An evaluation of verification procedures for cfd applications. In 24th *Symposium on naval hydrodynamics*, Fukuoka, Japan.

Eça, L.; Vaz, J.; Falcão de Campos, G.; Hoekstra, M. (2004): Verification of calculations of the potential flow around two-dimensional foils. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 42, no. 14, pp. 2401–2407.

Hall, D.; Zingg, D. (1995): Viscous airfoil computations using adaptive grid redistribution. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 33, no. 7, pp. 1205–1210.

Ham, F.; Lien, F.; Strong, A. (2002): A cartesian grid method with transient anisotropic adaptation. *Journal of Computational Physics*, vol. 179, pp. 469–494.

Jameson, A.; Schmidt, W.; Turkel, E. (1981): Numerical solutions of the euler equations by finite volume methods with runge-kutta time stepping schemes. AIAA Paper 81-1259.

Kallinderis, Y.; Baron, J. (1989): Adaptation methods for a new navier-stokes algorithm. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 27, no. 1, pp. 37–43.

Kroll, N.; Fassbender, J. (2005): *MEGAFLOW - Numerical flow simulation for aircraft design*, volume 89 of *Notes on numerical fluid mechanics and multidisciplinary design*. Springer. ISBN: 3-540-24383-6.

Marcum, D. (1996): Adaptive unstructured grid generation for viscous flow applications. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 34, no. 11, pp. 2440–2443.

Mavriplis, D. (2000): Adaptive meshing techniques for viscous flow calculations on mixed element unstructured meshes. *International Journal for Numerical Methods in Fluids*, vol. 34, pp. 93–111.

Mohammadi, M.H. (2008): Stabilized meshless local petrov-galerkin (MLPG) method for incompressible viscous fluid flows. *CMES: Computer Modeling in Engineering & Sciences*, vol. 29, no. 2, pp. 75–94.

Nithiarasu, P.; Zienkiewicz, O. (2000): Adaptive mesh generation for fluid mechanics problems. *International Journal for Numerical Methods in Engineering*, vol. 47, pp. 629–662.

Numeca International (2002): *User manual on Hexpress*. Technical report, Brussel, 2002. website: http://www.numeca.com/index.php?id=29.

Numeca International (2003): *User manual on Numeca's Flow Integrated Environment*. Technical report, Brussel, 2003.

Oh, W.; Kim, J.; Kwon, O. (2002): Numerical simulation of two-dimensional blade-vortex interactions using unstructured adaptive meshes. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 40, no. 3, pp. 474–480.

Oh, W.; Kim, J.; Kwon, O. (2003): Time-accurate navier-stokes simulation of vortex convection using an unstructured dynamic mesh procedure. *Computers & fluids*, vol. 32, pp. 727–749.

Orsini, P.; Power, H.; Morvan, H. (2008): Improving volume element methods by meshless radial basis function techniques. *CMES: Computer Modeling in Engineering & Sciences*, vol. 23, no. 3, pp. 187–207.

Pasquim, B.M.; Mariani, V.C. (2008): Solutions for incompressible viscous flow in a triangular cavity using cartesian grid method. *CMES: Computer Modeling in Engineering & Sciences*, vol. 35, no. 2, pp. 113–132.

Patel, A. (2003): *Développement d'un solveur* RANS *adaptif sur maillages non-sturcturés hexaédriques*. Ph.d., Université Libre de Bruxelles, Brussels, 2003.

Patel, A.; Léonard, B.; Elsden, M.; Hirsch, C. (2003): A parallel multigrid adaptive industrial flow solver on all-hexahedra unstructured meshes. In *International conference on adaptive modeling and simulation (*ADMOS*)*, Barcelona, Spain.

Pirzadeh, S. (2000): A solution-adaptive unstructured grid method by grid subdivision and local remeshing. *Journal of Aircraft*, vol. 37, no. 5, pp. 818–824.

Rausch, R.; Batina, J.; Yang, H. (1992): Spatial adaptation of unstructured meshes for unsteady aerodynamic flow computations. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 30, no. 5, pp. 1243–1251.

Simms, D.; Schreck, S.; Hand, M.; Fingersh, L. (2001): Nrel unsteady aerodynamics experiment in the nasa-ames wind tunnel: a comparison of predictions to measurements. Technical report nrel/tp-500-29494, NREL, Colorado, 2001.

Spalart, P.; Allmaras, S. (1992): A one-equation turbulence model for aerodynamic flows. In *AIAA* 30th *Aerospace Sciences Meeting & Exhibit*, Reno, NV. AIAA Paper 92-0439.

Tchon, K.; Hirsch, C.; Schneiders, R. (1997): Octree-based hexahedral mesh generation for viscous flow simulations. In *AIAA CFD conference*, Snowmass, CA. AIAA Paper 97-1980.

Timmer, W. (2001): Some aspects of high angle-of-attack flow on airfoils for wind turbine application. In *EWEC 2001 European Wind Energy Conference*.

Vos, J.; Rizzi, A.; Darracq, D.; Hirschel, E. (2002): Navier-stokes solvers in european aircraft design. *Progress in aerospace sciences*, vol. 38, no. 8, pp. 601–697.

Walsh, P.; Zingg, D. (1997): On the accuracy of viscous airfoil computations using solution-adaptive unstructured grids. In *AIAA* 35th *Aerospace Sciences Meeting & Exhibit*, Reno, NV. AIAA Paper 97-0329.

Walsh, P.; Zingg, D. (2001): Solution adaptation of unstructured grids for two-dimensional aerodynamic computations. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 39, no. 5, pp. 831–837.

Wang, Z.; Chen, R. (2002): Anisotropic solution-adaptive viscous cartesian grid method for turbulent flow simulation. *American Institute of Aeronautics and Astronautics Journal, Inc.*, vol. 40, no. 10, pp. 1969–1978.

Warren, G.; Anderson, W.; James, L.; Krist, S. (1991): Grid convergence for adaptive methods. In *AIAA* 10th *Computational Fluid Dynamics Conference*, pp. 729–741, Honolulu, Hawaii. AIAA Paper 91-1592-CP.

Zhao, F.; Zhu, S.-P.; Zhang, Z.-R. (2005): Numerical experiments of a benchmark hull based on a turbulent free-surface flow model. *CMES: Computer Modeling in Engineering & Sciences*, vol. 9, no. 3, pp. 273–285.