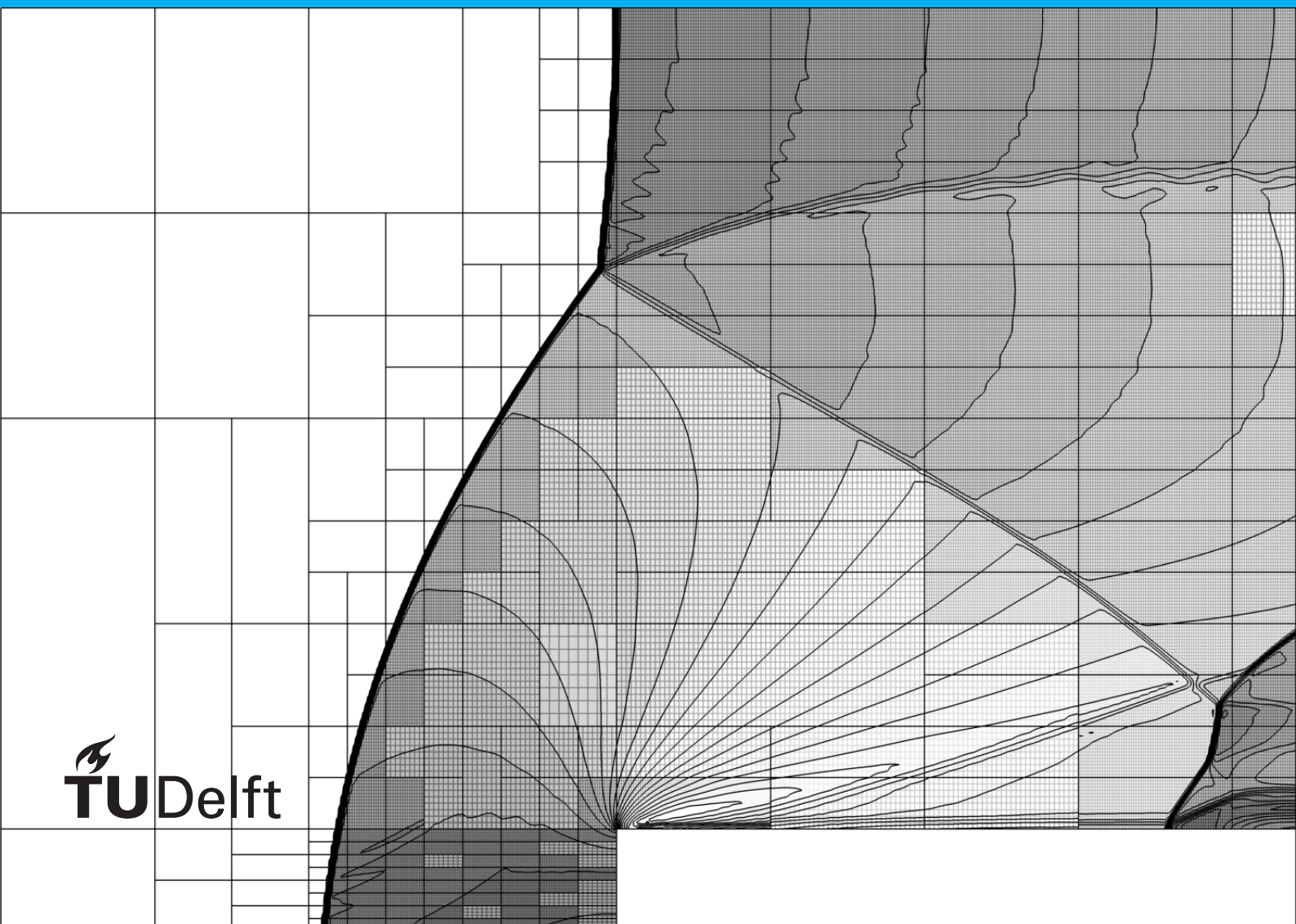
Error-estimate based adaptive mesh refinement

A user-independent approach for LES

Nils Barfknecht



TU Delft

Error-estimate based adaptive mesh refinement

A user-independent approach for LES

by

Nils Barfknecht

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Tuesday April 14, 2020 at 1:00 PM.

Student number:	4150198	
Project duration:	January 15, 2019 – April 14, 2020	
Supervisor:	Prof. Dr.-Ing. habil. S. Hickel	TU Delft
Thesis committee:	Prof. Dr. D. A. von Terzi	TU Delft
	Dr. S. J. Hulshoff	TU Delft
	X. Li	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This thesis started with my personal goal of obtaining a deeper understanding and hands-on experience with a state of the art CFD code. Right from the beginning, I was thrown into deep-end: Getting a grasp of INCA, a CFD code with a formidable size and complexity, but also making significant decisions very early on in the project was not an easy task. Nevertheless, I am pleased with the outcome of this project on an academic as well as personal level.

I want to thank Prof. Dr.-Ing. habil. Stefan Hickel for all his support during the entire project. Especially his enthusiasm for this thesis was highly appreciated. Without our productive (bi-)weekly meetings and his INCA support, this project would not have been possible. Also, many thanks to Xiaodong Li, who attended almost all of the meetings and provided further support throughout the project. I am looking forward to my time ahead as a PhD-student here at the Aerodynamics and Wind Energy department at the Aerospace Faculty of TU Delft under the supervision of Prof. Dr. Dominic von Terzi and Prof. Dr.-Ing. habil. Stefan Hickel. During my master's thesis, I really started to enjoy doing research, and I am excited about the possibility of continuing this endeavor also in the next step of my professional career.

Special thanks to all my friends that I have met here during my time at TU Delft and the fantastic time we had together. The numerous board game evenings and pub quizzes we had together is something to be remembered. In addition, I would like to thank all members of my student team Forze for the incredible experience of building the fastest hydrogen-electric sports car on the planet. The countless hours we have spent working together on the project led to a road of success, which is truly astonishing. My girlfriend was also a steady source of support in the years we have been together. Last but not least, I would also like to thank my family in Germany for their support throughout my time as a bachelor and master student. Knowing that someone is standing behind you in the decisions that you take was extremely valuable and crucial for my success.

Nils Barfknecht
Delft, April 2020

Abstract

Computational fluid dynamics (CFD) has become an indispensable tool in research and engineering. Even though the performance of computers is increasing at unfathomable speeds, the calculation of many fluid problems remains to be challenging. With the emerging widespread use of Large Eddy Simulation (LES), the computational cost of simulations has further increased. Nowadays, the successful application of CFD is highly dependent on the user. A crucial factor that influences computational performance, as well as the accuracy and reliability of the simulation results, is the underlying spatial discretization of CFD problems. Adaptive mesh refinement has the potential to address these issues, by automatically creating a computationally-efficient and user-independent mesh.

The objective of this Master's thesis is to improve the computational efficiency of LES CFD simulations by developing and testing a novel more user-independent AMR error sensor. A series of novel error sensors are proposed that are based on an error estimate, which is obtained during run-time by comparing the results of a fine and coarse-grained simulation. A popular reference error sensor based on the curvature of velocity magnitude is used as a comparison in three different test cases: Mach 3 flow over a forward-facing step, flow over a two-dimensional cylinder at Re=100, and flow over periodic hills at Re=10595.

All error sensors performed similarly for the Mach 3 flow over a forward-facing step. Essential flow features such as the bow shock, shock reflections, and slip lines were captured and well resolved. Differences in performance were mainly attributed to the control of the adaption routine. Anisotropic refinement led to a further error reduction in the range of 10 to 15%. Performance in the laminar two-dimensional cylinder case varied substantially. The novel error sensor based on the percentage error in the solution performed the best, leading to up to five times the computational savings in comparison to the reference indicator. Across all tests, anisotropic refinement was not able to lead to any computational savings when considering the run-time of the problem. The performance of all error sensors was underwhelming for the flow over periodic hills. The error sensors failed to refine the upper wall, which, without wall-function, lead to a significant error in the solution.

The overall performance of some novel error sensors was shown to be promising. Large computational savings in a robust user-independent AMR routine were presented for the compressible and laminar cases. However, more research is required for the successful applications in turbulent flows. Arguments were presented that highlight why adaption based on a target mesh size in conjunction with mesh coarsening is superior to regular threshold-based adaption. Directions and possible solutions to make adaption for turbulent flows successful are given as well.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

List of Symbols

Symbol	Description
A	Anisotropic energy-based error function
$C_{l,\max}$	Maximum lift coefficient
C_d	Drag coefficient
C_s	Smagorinsky constant
h	Grid length/Cell spacing
h	Fine grid length (subscript, superscript)
H	Coarse grid length (subscript, superscript)
I	Prolongation operator
J	Functional
k	Turbulent kinetic energy
k_{res}	Resolved turbulent kinetic energy
k_{sgs}	Sub-grid turbulent kinetic energy
L	Physical integral scale
l	Pseudo integral scale
M	Mass matrix
N	Numbers of cells
Nx	Numbers of cells in x-direction
Ny	Numbers of cells in y-direction
Nz	Numbers of cells in z-direction
n	Normal vector
p	Pressure
p	Order of numerical method
R	Residual of flow a problem
Re	Reynolds number
r	Subgrid-resolution
S	Cell face area
S	Rate of strain
St	Strouhal number
T	Time at last time step
T	Problem specific time scale
t	Time
t_{sim}	Simulated time
U	State variable/Flow solution
u	Velocity
u_{ref}	Problem specific reference velocity
x	Spatial coordinate
x/h	Non-dimensionalized x-length
y/h	Non-dimensionalized y-length
α	Design variable (for adjoint)
γ	Ratio of specific heats

Δ	LES filter width
ΔV	Cell volume
ϵ	Rate of dissipation of turbulence energy
ϵ	Error
ϵ_{AMR}	Error sensor adaption threshold
ϵ_{LES}	Total error of LES simulation
ϵ_{num}	Numerical error in LES simulation
ϵ_{SGS}	LES sub-grid scale error
ϵ_t	Total dissipation
ϵ_μ	Molecular dissipation
η	Kolmogorov length scale
μ	Dynamic Viscosity
ν	Kinematic viscosity
ν_t	Turbulent viscosity
ρ	Density
τ	Transformed time (for adjoint)
τ	Truncation error
τ	Shear stress
τ_{AMR}	Time between AMR calls
Φ	Error sensor
Ψ	Adjoint variable
Ω	Domain

List of Abbreviations

AMR	Adaptive mesh refinement
BiCG	Biconjugate gradient stabilized
CFD	Computational fluid dynamics
CFL	Courant–Friedrichs–Lewy number
CPU	Central processing unit
DNS	Direct numerical simulation
ETE	Error transport equation
FEM	Finite element method
FTT	Flow through times
IB	Immersed boundary
LES	Large eddy simulation
LTE	Local truncation error
MPI	Message passing interface
PI	Proportional-integral controller
PTC	Poisson-timestep-cell metric
RANS	Reynolds-averaged Navier-Stokes
SGS	Subgrid-scale
TKE	Turbulent kinetic energy
TVD	Total variation diminishing

1 Introduction

1.1. What is adaptive mesh refinement?

Computational fluid dynamics has become an indispensable tool for the prediction and calculation of fluid flow. In CFD, equations describing the physics of fluids, such as the Navier-Stokes equations, are used. Numerical methods are employed to solve the underlying differential equations on a computational domain. In contrast to experiments, CFD requires only enough computational power and, of course, a suitable computer code. Therefore CFD offers a cheap alternative to labor-intensive and resource-expensive experimental testing. Further, it provides a complete flow solution for the whole domain. Enabling unprecedented insight and understanding of fluid flow.

Since the introduction of the transistor and the integrated circuit, the computational power has increased manifold, from 1994 to 2014, the combined computational power of the world's fastest supercomputers has doubled approximately every 15 months [6]. The ever-growing computational power has allowed the computation of increasingly complex flow problems. However, challenges exist, especially in turbulence modeling, while Reynolds-averaged Navier–Stokes (RANS) turbulence models are nowadays considered industry-standard; they lack accuracy in vortex dominated and transitional flows [7]. Since the computational cost of Direct Numerical Simulation (DNS) scales with the power of three with the Reynolds number, reliance on turbulence modeling will be required for the foreseeable future in high Reynolds number flows [8]. Current research in turbulence modeling focuses heavily on Large Eddy Simulations (LES), where the turbulence up to a particular length scale is resolved, and turbulence below this scale is modeled. LES simulations are generally considered to be superior in accuracy in comparison to RANS simulations but are computationally much more expensive.

A discrete grid, comprised of cells, is used to solve the Navier-Stokes equations numerically. With an increasing number of cells, the accuracy of the solution increases, but unfortunately, the computational costs do as well. At a certain point, a grid-independent result is obtained, increasing the number of cells will not lead to a significant change in the result anymore. Striving for a grid-independent result can lead to an excessive amount of computational cost; this is especially true in the case of LES simulations. To alleviate this problem, CFD cases have to be set up smartly so that the computational costs are reduced as much as possible. One approach to this is a method called adaptive mesh refinement (AMR).

Errors in CFD simulations can either be numeric or stem from deficits in the modeling of, for example, turbulence. The numerical error depends on the size of the grid and the temporal discretization. A decrease in these two scales will yield a reduction in the numerical error. Research has,

however, shown that across the whole computational domain, the error contribution is not uniform, especially when only considering an output of interest, such as the drag of an airfoil [9]. Therefore different areas of the mesh have varying importance on the solution of the CFD simulation. The idea of adaptive mesh refinement is to exploit this observation by only spatially and temporarily resolving those parts of the domain that are essential for the solution. This approach allows saving up to orders of magnitude of cells in comparison to a uniformly refined mesh [10]. To visualize this result, a shock-dominant problem is considered, such as a blast simulation. It is apparent that for an accurate solution the shock front has to be well resolved, regions in which no shock front is present, however, require a much lower number of cells.

To conclude, in AMR, the mesh is refined in regions of interest or regions of importance and coarsened in regions with little impact on the quantities of interest. However, deciding where to refine and to coarsen and also how to refine and to coarsen is a question that can not be quickly answered. There is no 'best' way of accomplishing this, and different problems might require different approaches. With the broader adoption of LES, new questions in the field of AMR arise. LES computations are intrinsically transient, which render specific AMR methods prohibitively expensive; this is especially true for output-based methods [11]. Further, LES introduces another error source, namely the turbulence model. The goal of this project is to improve the solution accuracy of LES CFD simulations for a certain amount of computational cost by using user-independent, reproducible adaptive mesh refinement.

1.2. AMR for user-independent mesh generation

Much research for AMR in the 80s and 90s was driven by the need to simulate shock-wave problems [12]. Uniform refinement of the entire domain to capture the shocks would have been way too costly. However, another compelling argument for AMR lies in the aim of having a user-independent CFD experience. Nowadays, meshing is mostly still done manually using meshing software, it is therefore almost guaranteed that if one specific CFD problem is given to e.g. 10 different users, they will all create a different mesh and thus also obtain different results. Thus AMR is not only advantageous for creating a mesh that saves computational time but also to create a mesh that is user-independent and thus easily reproducible by different people to get consistent results. Yet another motivation for AMR, closely connected to the prior one, is the computation of entirely new flow problems, thus simulations where no a priori knowledge about the solution and flow structures is available, in that case, a manual meshing is guess-work at best. In such a case, a uniform mesh might be the only feasible approach. This, however, might be too costly to compute. Automatic mesh generation will also save tremendous amounts of user-time during the pre-processing stage of a CFD problem. One should note that there are some semi-automatic mesh generation packages available, such as snappyHexMesh from OpenFOAM (`https://www.openfoam.com/`). With these, it is possible to automatically create a mesh according to a configuration file. This can be very powerful and helpful when one has to compute very similar cases over and over again, e.g., in an optimization setting, nevertheless, this input file is again handcrafted and highly specific to a particular problem. To conclude, AMR can not only be seen as a necessity for computational cost savings but also as a solution to obtain automatic mesh generation.

New challenges in AMR have arisen with LES becoming more and more mainstream, at least in the academic world. Whereas previously often steady-state problems were encountered, now every problem becomes with LES inherently transient. This provides additional challenges. In transient problems, adjoint-based AMR, a potent approach that can be regarded as an almost ideal candidate for an AMR error-sensor[1] in steady-state flow problems, becomes infeasible due to the extreme computational cost which arises from the need to integrate the adjoint solution across all time-steps

[1] In case the reader is not familiar with AMR and error sensors, in particular, one should consult Chapter 2.

back in time.

One fundamental problem that is closely connected to AMR is the question of: What is a reliable CFD simulation? Or LES specific: What is a reliable LES simulation? Substantial research has been done for this problem, see, e.g., [13]. The judgment of the reliability, thus, to determine how far the result is away from the actual correct result, shares striking similarity to AMR. AMR is usually based on mesh adaption driven by an error sensor that aims at estimating or indicating the error in the solution. The literature study highlights some of these methods used to judge the quality of a LES and how they could and are used for mesh adaption. Since both topics are closely related and the question of what is a good LES simulation is still open for debate; also, the question of what is a good AMR routine remains unsolved. So far the question whether a mesh (or simulation) is good, is mostly answered by experience of the user and lengthy grid convergence studies.

Another aspect of AMR is the cost-accuracy trade-off in a simulation. For the user, it is often not possible to establish, before an AMR simulation is started, how long the simulation is going to take, or what the error is going to be. Since AMR influences these two points precisely, a good AMR routine should be able to give information on both. However, most methods fail to make a statement about any of these two. Within the most common AMR method, an error function threshold is defined. The mesh is adapted until all cells drop below this threshold. Usually, a straightforward error sensor such as the undivided difference of velocity is used. By employing such an approach, it is neither possible to make an a priori or a posteriori statement about the accuracy of the simulation, nor is it possible to beforehand estimate the computational cost of the simulation. One is stuck performing multiple simulations to establish grid convergence, and computational cost establishes itself during the simulation with the consequence that one might have to terminate the simulation because it becomes too costly. From this discussion, it becomes clear that AMR is still a field of many questions. The next section highlights how this thesis will add to the body of knowledge with the ultimate goal to at some point have found answers to all of the described challenges.

1.3. Research objective and thesis outline

With the conclusions of the previous section in mind the main research objective has been defined for this master thesis:

Improve the computational efficiency of LES CFD simulations by developing and testing a novel more user-independent AMR error sensor.

Due to the broadness of the objective, first, a suitable research direction had to be defined. For this purpose, an extensive literature study was performed that reviewed the current state of the art in adaptive mesh refinement (Chapter 2). The outcome of this review laid the foundation for developing new ideas and directions to accomplish the research goal (Chapter 3). A set of research questions has been defined that is used as a red thread throughout this thesis. Since a user-independent AMR routine has to work for a large variety of problems with different flow physics, three different benchmark problems have been chosen. The Mach 3 flow over a forward-facing step [2], flow over a two-dimensional cylinder at Re=100 [14], and flow over periodic hills at Re=10595 [15]. The first problem is used to investigate the AMR routine's behavior in compressible flow. The second and third problem is used to verify the performance in a laminar and turbulent flow. As an extra complication, both cases contain an immersed boundary. The results of all three test cases are presented in Chapter 4, 5, and 6 respectively. Based on the outcome of the literature a popular error sensor has been chosen as a benchmark that is used as a reference for the newly developed methodology.

A set of research questions has been defined to guide the research.

1. Are the current state of the art error sensors already able to provide a systematical approach

to user-independent and reproducible adaptive mesh adaption?

Rationale: This question aims at assessing the current state of the art of AMR. It also serves the purpose of establishing a foundation that will be used for the development of new error sensors. This question will thus be answered at the end of the literature study.

2. For all three test cases and isotropic refinement, how does the resulting solution error compare in terms of cell count and computational cost for the novel and reference error sensors?

3. For all three test cases and anisotropic refinement, how does the resulting solution error compare in terms of cell count and computational cost for the novel and reference error sensors? [2]

Rationale: One of the main goals of AMR is to save computational cost. Benchmarking the error sensors is, therefore, crucial. Additional cost savings can often be obtained by employing anisotropic mesh adaption in which the cells are refined in particular directions only. The solution error is also of major importance for user-independence since user-independence implies that an AMR routine must also lead to a correct result.

4. Should mesh adaption be performed every physical time step, or can the computational efficiency be increased by performing adaption merely every N time steps. [3]

Rationale: By using LES, every problem becomes inherently transient. The flow is thus not fully developed from the onset of the simulation and might need to develop first (e.g., the wake behind a cylinder). Therefore, a strategy might be required that spreads adaption in time. Some error sensors might also be dependent on a specific amount of elapsed time for their successful working, such as described in Chapter 3.

5. Will all investigated error sensors lead to a robust adaption routine?

Rationale: Error sensors might show many unwanted behaviors, such as being extremely sensitive to the initial grid, causing never-stopping adaptions, or adapting unwanted regions. A well-behaving error sensor is essential for a user-independent AMR routine.

6. Should adaption be based upon the instantaneous or time-averaged state variables for the turbulent flow?

Rationale: Due to the chaotic nature of turbulent flow, it is expected that instantaneous flow variables might not be a good candidate to drive adaption. Averaging the state variables is most likely going to result in a computationally more efficient mesh.

7. What is the computational overhead of the tested error sensors?

Rationale: If the actual error sensor is too expensive to compute, AMR might not lead to any meaningful savings in computational cost.

8. What kind of qualitative adaption characteristics do the investigated error sensors show for all three test cases?

Rationale: This question aims at describing the behavior of the error sensors. Do some error sensors prefer to refine boundary layers, while others are much more likely to refine wakes? Again, for user-independent mesh adaption, error sensors should capture all essential flow phenomena in all test cases.

[2] Due to time constraints, it was unfortunately not possible to investigate anisotropic refinement for the flow over periodic hills.

[3] This question was also answered in the context of the Richardson and ETE-type error senors presented in Chapter 3.

The outcome of this thesis, however, is not limited to these research questions. Throughout the thesis, generally applicable remarks and recommendations are made based on the results and experiences obtained. These are summarized and Chapter 7, which also gives directions for future research.

2

Literature Review

Even though there is extensive review literature available within the AMR community, actual introductory texts into the topic are very scarce. This literature review has the aim to briefly summarize and explain the most important directions of adaptive mesh refinement with a particular focus on error sensors for LES (Large Eddy Simulation). The outcome of this literature study will lie the foundation for the development of the theoretical framework of this Master's thesis. This literature study is, therefore, an essential part for the fulfillment of the research objective.

First, an introduction into the different AMR adaption strategies is given, i.e., describing the methods that exist to modify a simulation to obtain a higher accuracy solution. Next LES specific difficulties are discussed concerning AMR. A review of the three main branches of error sensors is given afterward. Particular focus is put on their applicability for transient problems with LES modeling. As the last part of any AMR routine, a review of flag functions is given. Finally, the findings are summarized in a trade-off table, and a direction for future research is chosen.

2.1. Overview of AMR adaption routines

An AMR routine consists of three parts: an adaption strategy, an error sensor, and a flag function. In the continuing discussion, the word 'adaption' will be used highlighting the fact that a mesh can not only be refined but also coarsened and thus modified in both directions.

The adaption strategy determines how the mesh is altered. Common methods are H, R, and P-adaption. In H-adaption, a mesh cell is split into smaller cells, or multiple small cells are merged into a larger one. In R-adaption, the number of cells stays constant, and the cell connectivity is maintained as well, but they are stretched and skewed to effectively enrich regions in need of high refinement and coarsen regions requiring only low cell density. In P-adaption, the mesh is not altered, but different numerical techniques with a different order of accuracy are chosen. In a Finite Element Method (FEM) framework basis functions of different orders are selected.

The second ingredient of the AMR routine is the error sensor. An error sensor quantifies the error that every piece of the computational grid introduces into the flow solution due to the underlying discretization or flow model. In general, there are three big branches of error sensors. The first branch of these is based on the idea of using physical arguments for the adaption routine. This encompasses the tracking of flow features and structures such as shocks with, e.g., large flow gradients. The second branch takes a more rigorous mathematical approach and tries to estimate the real error introduced by the used discretization schemes. The third branch is based on the goal-based mesh adaption using an adjoint formulation. The idea is to have an error sensor that can provide information about which mesh regions influence the accuracy of a particular output of interest.

The last part of an AMR routine is the flag function, while the error sensor has put a numerical value on the error contribution of every mesh cell, the error function does not decide what do to with this value. The flag function ultimately determines which regions to adapt and which regions should remain unchanged.

2.2. Adaption strategies

Popular strategies for AMR are H, R, and P-adaption. The former change the mesh, whereas the latter changes the numerical scheme. The adaption strategy was determined by the solver which was used for this thesis: INCA (see Section 3.1) is based upon a block-structured mesh and has H-adaption capability. For completeness, a small introduction is given into the general aspects of H-adaption. In this strategy, the mesh is altered through the addition and removal of cells by splitting and merging operations. All these operations work with a fixed integer. H-adaption can be used for isotropic and anisotropic refinement and can also be used on structured and unstructured meshes. H-adaption is very flexible concerning its initial mesh since the total number of cells is not intrinsically fixed. Even a poor initial mesh that might be massively over- or under-refined can be transformed into an optimal mesh. In contrast to other methods, changes to the mesh are locally constrained within the domain. Operations such as interpolation due to adaption happen thus only locally. Parts of the mesh not in need of adaption thus remain unchanged. Two complications inherent to the H-adaption is the need for ghost cells and only an incremental change in mesh size. The former can create substantial overhead in cell count, whereas the latter might lead to over- or under-refinement when the error only slightly violates a certain error sensor. In general, there are three different types, cell-based, patch-based and block-based H-adaption. In block-based H-adaption, the domain is split into separate blocks, each with their own quantity of cells and dimensions. Any adaption operation acts on the entire block and not merely on individual cells. With this approach, it is possible to create meshes that can be effectively used for parallel computation since only the blocks have to be distributed on different processes [16]. This approach also avoids excessive amounts of ghost cells. In contrast to this approach sits cell-based adaption (tree-based

adaption), every cell can be individually adapted. The lineage of every cell is often stored in a quad- or octree. Within the cell-based framework, one has much flexibility regarding the mesh and local features can be precisely refined resolved, leading to minimum cell count. Unfortunately, precisely these properties can lead to very significant fragmentation and excessive quantities of ghost cells, creating ample storage and communication overhead [17]. Patch-based adaption alleviates the excessive fragmentation and overhead caused by cell-based adaption by grouping cells in so-called patches. A grouping algorithm collects cells of the same level flagged for adaption; it then sorts these cells into rectangular patches and adapts the patch as a whole. Some patch-based methods allow for stacking patches on top of each other in any orientation [18]. R-adaption is a method, in contrast to H-adaption, where no cells are created or destroyed during adaption. Instead, the cell count is defined a priori. The cells are shifted and warped in space. Therefore cells are accumulated in regions of interest and are moved away from regions that are considered to be a small error contributor. Apart from the advantage of aligning the mesh with features, the mesh data structure also remains unchanged, thus solver without AMR support can easily be coupled with R-adaption routines [19]. In P-adaption, not the mesh is changed but the order of the discretization. This means in a FEM context varying the polynomial degree of the elements [20]. Combinations of these methods also exist and allow to combine the advantages of the respective methods [21, 22].

2.3. Error sensor

The output of the error sensor is used to determine where mesh adaption is required. In literature, often, the terms error indicator and error estimator are used, sometimes also interchangeably. In this literature study, the term error sensor is used as a generic term for both. Following Jasak and Gosman [23], we define an error estimator as a method that claims to represent the actual error between exact and discretized solution. An error indicator, in contrast, merely indicates which cells might introduce errors, there is no claim that an error indicator either actually measures the error, is based on rigorous mathematical foundations, or is effective. Summarizing, the error sensor is a generic term for both error indicator and estimator.

This chapter attempts to review the major error sensors found in the literature. A particular emphasis is put on the applicability of standard error sensors for LES. It should be noted in advance that all error indicators have their strengths and weaknesses. Right now, there is no single best indicator. One can divide error indicators and estimators into three branches. The first branch is based around the idea of tracking flow features and structures such as vortices and large gradients in the domain. Numerical error functions comprise the second branch, they either try to estimate the actual error in the solution or use numerical values such as the residual to drive adaption. The last branch is based on the goal-based mesh adaption using an adjoint formulation. These error sensors try to forecast which cells need to be adapted to reduce the solution error in an output of interest or functional.

2.4. LES specific issues when using AMR

LES is distinct from DNS in that it does not only contain a numerical but also a modeling error. Thus, an error is introduced in the turbulence model due to the discretization of the flow problem. This is a consequence of the LES turbulence model having a filter size, which is usually dependent on the grid size of the problem. For conceptualization, the total error in a LES computation with respect to the true solution of the considered equations is

$$\epsilon_{LES} = \epsilon_{num} + \epsilon_{SGS}\,. \tag{2.1}$$

It should be noted that ϵ_{num} and ϵ_{SGS} are usually correlated [24]. From this equation, one can

predict the occurrence of error cancellation. Both error terms do not necessarily have the same sign. Therefore it is possible to obtain a low overall error while numerical and modeling error are large [10]. Qualitative predictions for the total error can be obtained by considering the sub-filter resolution (r):

$$r = \frac{h}{\Delta}, \tag{2.2}$$

where h is the mesh spacing and Δ the filter size of the LES turbulence model. If $r << 1$, then the numerical error is negligible in comparison to the modeling error. In this case, one could speak about a grid-independent LES. When choosing $r \approx 1$, more turbulence scales could be resolved while keeping the cost of the simulation constant. Even though this reduces the modeling error, the numerical error can become in turn substantial [25]. In practical LES applications, r is usually chosen to be one [26]. Based on these observations, the concept of grid-independent implicit LES is questionable. In case for $r = 1$ and $h \Rightarrow 0$ LES will turn into DNS. Choosing $r << 1$ to obtain a LES solution with negligible numerical error and thus a 'grid independent result' is only possible in explicit LES.

The question of: What is a good LES solution? Is therefore subject to discussion and not answered yet [24, 25, 27–29]. Some authors argue that the quality of a LES simulation should, therefore, be seen in the context of the convergence of a functional [30].

2.5. Physics/Feature-based error indicators

Feature-based error sensors are one of the earliest and most straightforward methods that have been adopted in the AMR community. This error sensor does not compute the actual error in the solution but makes the simple assumption that regions with a large error indicator value contribute significantly towards the error [31]. Popular choices for a feature-based error sensor are, for example, the undivided difference velocity or pressure gradient. In such a case, the assumption is made that if a larger than average gradient in a state variable exists over a mesh cell, then this mesh cell will contribute significantly towards the error in the flow solution. This is obviously a substantial assumption, and therefore feature-based error sensors are also regarded as somewhat heuristic since they do not calculate the error but merely indicate where an error in the flow solution might stem from. They are also called feature-based error indicators. The possibility of separating the large and small scale turbulence in LES has lead to the development of LES specific feature-based error sensors.

Feature-based indicators are usually easy to understand, use and implement further; they are often cheap to compute. They have a long track record and have been used with success in the past for various flow problems [12]. Unfortunately, they are not universal and must be used empirically, each flow problem requires a different error function, and the choice is very much dependent on the flow conditions. Therefore, knowledge about the problem to be solved must be known in advance. Further, there is neither a guarantee that the adapted areas actually contribute towards the error nor that important regions are left out for adaption. Due to these shortcomings, it is questionable if feature-based indicators can be user-independent. The wealth of experience provided by the CFD user is vital to choose the proper indicator for a specific flow problem.

2.5.1. Gradient based methods and others

Probably the easiest error sensor is the simple use of the undivided difference of a variable of interest as the error function. Classical variables are in this context, for example, pressure and velocity:

$$\Phi = h^n \left| \frac{\partial^n u}{\partial x^n} \right|. \tag{2.3}$$

Where h represents the cell size of the mesh. Instead of just using the first derivatives, also the second or higher-order derivatives can be used. Early studies using this approach are, for example, found in Dannenhoffer, III and Baron [32], where the use of first and second differences of the state variables for AMR refinement around an airfoil in transonic flow conditions was investigated. Gradient and curvature-based error functions are even after over 30 years the only ready to use methods available methods in Ansys Fluent, a very generic CFD package [33]. As one could also expect, a combination of derivatives has been used [12, 34]. Even though this family of indicators is very simplistic, they still seem to work well across a wide range. In Hertel *et al.* [19], a simple gradient of velocity was compared against LES specific indicator. It was found that the gradient of velocity consistently delivered a lower error and thus outperformed LES specific indicators in a test case of flow over periodic hills. Another approach is to track coherent flow structures such as in the vortex tracking criterion by Kamkar *et al.* [35], Gou *et al.* [36, 37].

2.5.2. LES specific indicators

Section 2.4 discussed the LES specific characteristics arising when using AMR. In the feature-based indicator group, a variety of special error functions for LES have been developed. All of them exploit the extra SGS-scale information obtained from the filtering operation.

A classic approach to exploit the extra information obtained from the SGS is the comparison of resolved and modeled parameters. Pope [38] gives the well-known recommendation that

$$\Phi_{k,res} = \frac{\langle k_{res} \rangle}{\langle k_{res} \rangle + \langle k_{sgs} \rangle} \tag{2.4}$$

should be above 80% for a well-resolved LES. In the formula, $\langle \cdot \rangle$ represents an averaging operation. A method on how to calculate the components is presented in [19, 39]. Davidson [40] tests this criterion for a fully developed channel flow for a Reynolds number of 4000 based upon friction velocity and half channel height. It was found that even though the resolved TKE was well above 85%, significant errors in the solution were present. A variation of this error function is

$$\Phi_{sgs} = \frac{\langle k_{sgs} \rangle}{\langle k_{res} \rangle + \langle k_{sgs} \rangle}. \tag{2.5}$$

Two practical issues with this error sensor were pointed out by Hertel *et al.* [19]. First, Φ_k might be large close to the wall (due to a large k_{sgs}) and thus suggest adaption. This, however, can have an impact on the validity of the wall function. Secondly, it was reported in case of the laminar limit and coarse grids, that $\Phi_k = 1$, due to the employed method for calculating k_{sgs}. Therefore, a modification is proposed:

$$\Phi_{k,tot} = \frac{\langle k_{sgs} \rangle}{\max\limits_{\Omega_c} \left(\langle k_{res} \rangle + \langle k_{sgs} \rangle \right)} \tag{2.6}$$

or

$$\Phi_{k,c} = \frac{\langle k_{sgs} \rangle}{\langle k_{res} \rangle + \langle k_{sgs} \rangle + C \max\limits_{\Omega_c} \left(\langle k_{res} \rangle + \langle k_{sgs} \rangle \right)}, \tag{2.7}$$

where C is a constant and is suggested to be 0.1. By taking the maximum total TKE in the flow domain instead of the local, one can alleviate the aforementioned problems. However, it introduces the problem that the error sensor will become in large parts of the domain extremely small. Therefore only the most dominant regions will be suggested for adaption. Weak, but important features will be left completely out for adaption. Another interesting variation to Equation 2.4 is given by Celik *et al.* [27]. In the proposed approach, the subgrid-scale TKE is obtained by Richardson extrapolation. This also allows modeling the contribution of numerical dissipation on the TKE.

$$\Phi_{k,res} = \frac{k^{res}}{k^{tot}}, \tag{2.8}$$

where

$$k^{tot} = k^{res} + k^{SGS} + k^{num}. \tag{2.9}$$

k^{num} is the so-called residual kinetic energy due to numerical dissipation. The assumption is made that kinetic energy due to dissipation and the sub grid scale both scale with the grid spacing h and the order of the scheme p:

$$k^{tot} - k^{res} = k^{SGS} + k^{num} = k^{eff\_SGS} = a_k h^p. \tag{2.10}$$

The proportionality constant a_k is determined by using Richardson extrapolation:

$$\Phi_{k,res} = \frac{k^{res}}{k^{res} + \frac{k_2^{res} - k_1^{res}}{\alpha^p - 1}\left(\frac{h}{h_2}\right)^p}, \tag{2.11}$$

where $\alpha = h_1/h_2$. If k^{num} and k^{SGS} have different orders, then the index of resolution quality can be adjusted by modeling both terms independently:

$$k^{tot} - k^{res} = k^{eff\_SGS} = ah^p + b\Delta^q, \tag{2.12}$$

which gives the adjusted $\Phi_{k,res}$. In this case, three grids are required to obtain both proportionality constants:

$$\Phi_{k,res} = \frac{k^{res}}{k^{res} + ah^p + b\Delta^q}. \tag{2.13}$$

Yet another indicator comparing resolved and SGS information is the subgrid-activity parameter introduced by Geurts and Fröhlich [25]. It is the fraction between the turbulent dissipation and total dissipation. $\Phi_s = 0$ corresponds to a DNS and $\Phi_s = 1$ corresponds to a LES at infinite Reynolds number:

$$\Phi_s = \frac{\langle \epsilon_t \rangle}{\langle \epsilon_t \rangle + \langle \epsilon_\mu \rangle}, \tag{2.14}$$

where according to the Smagorinsky turbulence model, one can express $\langle \epsilon_t \rangle$ and $\langle \epsilon_\mu \rangle$ as

$$\langle \epsilon_t \rangle = \langle 2\nu_t \bar{S}_{i,j} \bar{S}_{i,j} \rangle, \tag{2.15}$$

$$\langle \epsilon_\mu \rangle = \langle 2\nu \bar{S}_{i,j} \bar{S}_{i,j} \rangle. \tag{2.16}$$

Equation 2.14 can be approximately rewritten as the ratio of turbulent viscosity to total viscosity [27]:

$$\Phi_\nu = \frac{\langle \nu_t \rangle}{\langle \nu_t \rangle + \nu} = \frac{1}{1 + \frac{\nu}{\langle \nu_t \rangle}}. \tag{2.17}$$

Hertel *et al.* [19] tested both indicators (Equation 2.14 and 2.17) in an AMR routine and found that both methods lead to an almost identical grid. Unfortunately, using Equation 2.17 leads to very unsatisfying adaption for flow over periodic hills. The adaption routine did not decrease the error but, as a matter of fact, even increased it with respect to the initial grid. Celik *et al.* [27] criticizes the subgrid-activity parameter on two further points. Even though the turbulent dissipation can be calculated with the Smagorinsky model, it is difficult to separate it from numerical dissipation. Second the argument is made that usually $\nu_t >> \nu$ and thus $\Phi_\nu \approx 1$ and therefore insensitive to grid resolution. This could give an explanation for the poor performance of this error function. Data given by Hertel *et al.* [19] indeed hints at the fact that Φ_ν is very high and uniform across the domain.

For the Smagorinsky model the subgrid shear stress can be calculated as:

$$\tau_{12}^{mod} = -\nu_t \left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x}\right). \tag{2.18}$$

Based upon this, Hertel *et al.* [19] proposes an error function comparing modeled to total shear stress:

$$\Phi_\tau = \frac{\langle \tau_{12}^{mod} \rangle}{|\langle \tau_{12}^{mod} \rangle| + |\langle \bar{u}'' \bar{v}'' \rangle|}. \tag{2.19}$$

The absolute terms are taken for the denominator to avoid cancellation of the terms in case of different signs. It was reported that this error sensor proved to be more robust and not susceptible to erroneous adaption as it was the case with Φ_k.

In Hertel *et al.* [19] Φ_ν, Φ_τ, $\Phi_{k,C}$, $\Phi_{k,tot}$ are compared in an R-refinement strategy for a flow over periodic hill against the simple undivided difference of velocity error function. For predicting the averaged streamwise velocity at different stations, the undivided velocity difference turned out to be the best criterion. Φ_τ, $\Phi_{k,C}$ and $\Phi_{k,tot}$ were also able to improve the solution but turned out to be inferior for predicting the correct velocity profile. Adaption based upon Φ_ν even lead to a worse result in comparison to the baseline mesh. The velocity error function also proved to be very competitive for predicting the Reynolds shear stress at different locations. It was, however, beaten by Φ_τ, which is not very surprising. $\Phi_{k,C}$, $\Phi_{k,tot}$ again lead to an improved solution but still were not competitive. Φ_ν once again leads to worse results. Concluding, one can say that, for this case, a LES-specific criterion does not lead to a significantly better result in contrast to a simple generic velocity error function.

Celik *et al.* [27] proposes a quality index for LES. This concept is based upon the subgrid activity parameter by Geurts and Fröhlich [25]. The aim is to derive specific mesh requirements that are representative of a 'good' LES. In a similar fashion to the subgrid activity parameter, a quality index based upon the Kolmogorov scale is proposed:

$$LES\_IQ_\eta = \frac{1}{1 + \alpha_\eta \left(\frac{h}{\eta_k} \right)^m}. \tag{2.20}$$

Another index is based on the subgrid-scale viscosity:

$$LES\_IQ_\nu = \frac{1}{1 + \alpha_\nu \left(\frac{\langle \nu_{t,eff} \rangle}{\nu} \right)^n}. \tag{2.21}$$

The parameters $\alpha_\eta, \alpha_\nu, m, n$ are determined using the following set of assumptions. For a DNS it is assumed that $h \approx 1\eta_k$ and $\langle \nu_{t,eff} \rangle / \nu = 1$. Further, for LES the assumption is made that $h \approx 25\eta_k$ and $\langle \nu_{t,eff} \rangle / \nu = 20$. The thresholds for the quality index are set to be 80% for LES and 95% for DNS. Solving Equation 2.21 for the constants leads to: $m \approx 0.5, \alpha_{eta} \approx 0.05, n \approx 0.53, \alpha_\nu \approx 0.05$. These values are valid for $Re = 1200$. Equations to derive these constants for any Reynolds number are given in Celik *et al.* [27]. The argument that $h \approx 25\eta_k$ is obviously a very strong limitation. It is again derived by the aim to resolve 80% of the turbulent kinetic energy at a given Reynolds number. According to Pope [38] this is the case when $L/\Delta \approx 12$ for the sharp cutoff filter and $L/\Delta \approx 17$ for the Gaussian filter. Celik *et al.* [27] simply assumes an average value of both.

The original motivation for the LES_IQ criterion was the need for a solution verification tool for LES. The classical question is whether the LES solution is well and sufficiently resolved. In that framework, it has, for example, been used by di Mare *et al.* [29] to investigate LES flow in a combustion engine flow. LES_IQ was compared against the classical criterion of resolved to total TKE. Both methods performed similarly in rating the 'goodness' of different flow regions in the domain. The LES_IQ criterion translates itself naturally into an error function for AMR. Especially for LES_IQ_η the direct link between error function value and grid spacing can be seen as advantageous. Unfortunately, no reference was found where this criterion was used as an error function for AMR, even

though, as mentioned, this application seems obvious. As stated before, the performance of the subgrid-activity parameter is poor. Therefore the effectiveness of LES_IQ$_\nu$ might be questioned.

Further criticism can be expressed in the 'arbitrariness' and robustness of the derived equations. At the heart of the required tuning parameters is the assumption that a good LES must resolve 80% of all TKE, whereas a DNS should resolve 95%. The recommendation stems from Pope [38] and has already shown to provide insufficient adaption. Another rather heuristic approach is to take a mean value for the physical integral length scale to filter width ratio. Further, a constant ratio of L/l is derived. According to Gamard and George [41], this is only true for very high Reynolds numbers. The Reynolds number dependency of L/l from Gamard and George [41] could be implemented as an improvement. For the derivation of LES_IQ$_\nu$, an even larger variety of assumptions is made. The Smagorinsky constant is assumed to be a fixed values, but depending on the flow type, this is an insufficient assumption, e.g., in shear flows. Also, a model for the numerical dissipation was introduced but without much evidence on its correctness. Last but not least, the recommendation that $\frac{h}{\eta_k} \approx 25$ is extremely conservative and would lead to unfeasible large grids, even though it has been pointed out that $\frac{h}{\eta_k}$ is a function of the Reynolds number.

Gant [42] reviews two techniques in which the required filter size of the LES simulation is derived from a prior RANS computation. The idea is to derive the turbulent length scale from the RANS simulation and relate it via 'best practice' to the filter size. One can write the turbulent length scale as:

$$l_I = \frac{k^{3/2}}{\epsilon}. \tag{2.22}$$

The coefficients, for example, can be obtained from a standard $k-\epsilon$ turbulence model. Following Pope [38] it is proposed that $l_I/\Delta > 12$ for a sharp cut-off filter. In case of a Gaussian filter the limit changes to $l_I/\Delta > 17$. The approach is, therefore, similar to the one from Celik *et al.* [27] and again assumes that 80% of the TKE should be resolved. The second proposed method is to calculate the Kolmogorov scale from the prior RANS and then use the results of Celik *et al.* [27] to calculate the grid size of the LES simulation:

$$\eta = \left(\frac{\nu^3}{\epsilon}\right)^{1/4}, \tag{2.23}$$

where $\Delta/\eta < 25$ for good LES resolution Celik *et al.* [27], which, as discussed before, is extremely conservative. In general, the approach to derive the LES mesh from the RANS solution has to be seen with skepticism. Both indicators could also be calculated on the fly from the corresponding LES simulation. The from RANS derived mesh size is practically only valid in a statistically stationary flow. Under-resolving unsteady effects not captured by the RANS-suggested adaption areas might introduce significant errors in the simulation. The proposed approach, however, has the merit to supply a LES problem with an initial optimized mesh. This could be used for bootstrapping error sensors, which are dependent on a not too coarse initial mesh. Additionally, one could estimate the required cell count for an expensive LES via a cheap RANS computation. Since the computational cost for RANS is much smaller than for LES, the proposed method can deliver 'quasi-a-priori' information on how many cells are required for a well-resolved LES.

An error function based upon the idea that LES resolves turbulent production and models turbulent dissipation was presented by Hertel *et al.* [19]:

$$\Phi_{P_k} = -\sum_{i,j} \langle \bar{u}_i'' \bar{u}_j'' \rangle \frac{\partial \langle \bar{u}_i \rangle}{\partial x_j}. \tag{2.24}$$

This criterion can be further motivated by the observation that the gradient of streamwise velocity and the shear stress were able to provide beneficial adaption for the periodic hill case in Hertel *et al.*

[19]. This criterion effectively combines them. Accurate results were obtained for the velocity and shear stress distribution. The error sensor was compared against other LES specific error sensor (Φ_ν, Φ_τ, $\Phi_{k,C}$, $\Phi_{k,tot}$). Φ_{P_k} outperformed all of them by a great margin.

Toosi and Larsson [10] introduces a directional energy indicator measuring the small-scale energies in the flow. The small scale velocity ($\bar{u}_i^*$) in direction n can be approximately obtained by:

$$\bar{u}_i^{*,(n)} \equiv \bar{u}_i - \hat{\bar{u}}^{(n)} \approx -\frac{\Delta_n^2}{4} n^T (\nabla\nabla^T \bar{u}_i) n, \tag{2.25}$$

where $\bar{u}_i$ is the unfiltered velocity. The energy in that direction is then measured by:

$$\Phi = A(n) = \sqrt{\left\langle \bar{u}_i^{*,(n)} \bar{u}_i^{*,(n)} \right\rangle}, \tag{2.26}$$

where $\langle\rangle$ is a suitable averaging operator such as time- or phase-average. The main result is, therefore, that the Hessian or curvature of the velocity can be related to the small scale energies in LES. An interesting result that shows why using a simple second derivative of velocity can be justified by LES specific arguments.

Another successful AMR application was presented by Daviller *et al.* [43], an adaption routine based on resolved and unresolved viscous dissipation in a LES framework was used to predict the pressure loss in a swirl injector. First, the transport equation for the kinetic energy was inspected. It was deduced that the viscous dissipation controls part of the total pressure and thus pressure loss. Based upon this observation, it is suggested to use the viscous dissipation as the error function. Specifically, it is decided to use the time-averaged sum of resolved ϕ and unresolved viscous dissipation φ.

Hindi *et al.* [44] presents a LES specific AMR criterion for sprays based upon the Stokes number. For $St >> 1$ it can be assumed that the eddies will not affect the motion of the droplets. The aim is to resolve all eddies that are capable of dispersing the droplets of the flow.

2.6. Numerical error estimators

Numerical error estimators aim at estimating the actual error within the solution. This is different from error indicators, which merely hint towards regions with a significant error. It, however, could be argued that every error indicator is also an error estimator, just one which gives poor error estimates and is not based upon a rigorous mathematical formulation. Sources for numerical errors are: modeling error, round-off error, iterative convergence error, statistical sampling error, user/programming error and discretization error [45, 46]. Often the discretization error is the largest, and some consider it also the most difficult to compute [47]. The discretization error has two components, one which is locally-generated and one that is transported throughout the computational domain. Unfortunately, error estimation for hyperbolic partial differential equation systems is not straightforward. In contrast, for elliptic problems, a wealth of literature and methods have been developed [1]. Examples can be found in Zienkiewicz and Zhu [48] and Verfürth [49]. This section focuses on the three popular methods of Richardson-, Tau-extrapolation and the error transport equation approach. Other numerical error estimates include the moment error estimate by Haworth *et al.* [50] and the element residual error estimate by Jasak [31].

2.6.1. Richardson-extrapolation

Richardson-extrapolation is a technique that can be used to estimate the discretization error in a flow solution. The idea of the Richardson-extrapolation is to calculate a flow solution on a fine and a coarse grid. With the known order of the solver, one can infer the error with respect to the exact

solution. One can derive [18]

$$\Phi_\epsilon = |U - U_H| = \frac{|U_H - U_h|}{\left(\frac{h_H}{h_h}\right)^p - 1}, \tag{2.27}$$

where possible choices for h can be, for example, the ratio of cell volume and surface area [51]:

$$h = \frac{dV}{\sum_f |S|}, \tag{2.28}$$

and U being the state vector. In the most straightforward application of Richardson-extrapolation, the solution is calculated on two different sized grids. However, the same technique can also be used to estimate the error due to the temporal discretization. In this case, three solutions are required, one coarse solution, a solution on a finer grid and temporally finer one. Instead of different sized grids, one can alternatively also use numerical methods of different orders (P-refinement) [47]. A necessary condition for the success of the Richardson extrapolation is that both simulations lie within the asymptotic range [52]. This is a requirement that is probably never satisfied in LES. Further, from the inspection of the equations, it can also be seen that the order of the discretization scheme has to be known, in practical applications, the order of magnitude of the numerical scheme is however not always known and can depend on the flow conditions and the types of boundary conditions used. If the order of the discretization is not known, it can be calculated by using an additional simulation.

In the context of AMR, Richardson-extrapolation was first used by Berger and Colella [18], Berger and Oliger [53]. Other applications include work by Jasak and Gosman [23]. Its principal advantages are that it is a quite generic and problem-independent error sensor. Depending on the application, the implementation is also rather lightweight. Further, it is not dependent on the flow solver. Roy [47] is a good introductory text for Richardson extrapolation.

The application of Richardson-extrapolation in a LES framework is challenging due to the additional contribution of the modeling error introduced by the low-pass filtering. Mitran [54] attempted to use Richardson-extrapolation in a LES simulation. However, it was found that the AMR routine would become unbound and would always converge towards a DNS simulation. Unfortunately, no specific information is given on how exactly the routine was implemented. Another elaborate discussion about Richardson-extrapolation used in conjunction with LES is given by Klein [24]. Due to the additional modeling error, an algorithm is presented in which three simulations are used to separate the numerical and the modeling error. A fine grid solution, a coarse grid solution (half the mesh size) and a solution in which the model term is reduced (by 50%) are required. The derivation of the error is analogous to the standard case:

$$2(U_2 - U_1) = c_m h^m, \tag{2.29}$$

$$\frac{U_3 - U_1}{1 - 2^n} = (c_n + c_m) h^n \qquad m = n, \tag{2.30}$$

$$\frac{(U_3 - U_1) - 2(U_2 - U_1)(1 - 2^m)}{1 - 2^n} = c_n h^n \qquad m \neq n, \tag{2.31}$$

where n is the order of magnitude of the numerical scheme and m is the order of magnitude of the turbulence model. In the research of Klein [24] a Smagorinsky model was used, which was assumed to have second-order dissipation. According to Klein [24] it is difficult to produce three simulations in which all lie in the asymptotic range. Further, the presented method fails to work when implicit LES filtering is used because then modeling and numerical error interact and are hard to distinguish. Brandt [52] used the approach from Klein [24] with the same SGS model in a fully-developed turbulent channel flow at $Re \approx 6800$. It was found that the Richardson extrapolation assumption

of an asymptotic region was not sustained across all y+ values, but only fulfilled partially. For example, the model error was asymptotic in the log layer. This lead to a large discrepancy between true total error obtained from comparing the solution with a DNS reference solution and the error estimation by Richardson extrapolation. The results, therefore, confirm the experience from Klein [24] regarding the problems of obtaining a solution lying the asymptotic range.

2.6.2. Tau-extrapolation

τ-extrapolation is a variant of Richardson-extrapolation that does not require an entirely new solution on a different sized grid. Instead, a rough grid solution is constructed from the value on the fine grid. It exploits the fact that a solution obtained from a fine grid will usually lead to higher residuals on the coarse grid [55, 56]. For a general problem of the form

$$R(U) = 0 \tag{2.32}$$

the residual will be zero for the exact solution U. If the exact solution is substituted into the discretized problem $R_h()$ with mesh spacing h, the LTE (local truncation error) is obtained:

$$LTE_H = \frac{R_h(I^h U)}{h^d}, \tag{2.33}$$

where d is the dimension of the problem and I^h the operator that restricts the exact solution U on the grid h. The local truncation error is then the quantity used for mesh adaption. Problematic is, however, that the exact solution has to be known, which is obviously, for a practical problem, never the case. Therefore, one uses the fact that for a converged discrete solution, the residual on the discrete domain is usually zero:

$$R_h(U_h) = 0\,. \tag{2.34}$$

Further, the assumption is made that $U_h \approx U$. Restricting the fine solution on the coarse-grained mesh gives the approximate LTE analogous to Equation 2.33:

$$LTE_H = \frac{R_H(I_h^H U_h)}{h_H^d}\,. \tag{2.35}$$

Since the LTE decreases with the order of numerical method according to

$$|LTE| \leq Ch^p\,, \tag{2.36}$$

where C is a constant, one can see that for smaller mesh sizes, the LTE will tend toward zero. As for Richardson extrapolation, the error needs to lie in the asymptotic range. Otherwise, no reliable error estimate is obtained. The advantage of τ-extrapolation is that the restriction operation and subsequent evaluation of the residual are very cheap to compute. It, therefore, provides cost savings over regular Richardson-extrapolation. However, the argument can be made that a coarse-grained problem is so cheap in comparison to the primal problem, that the cost savings are not significant. Application of τ-extrapolation for AMR can be e.g. found in Aftosmis and Berger [1]. Unfortunately, no resource was found that compared τ and Richardson-extrapolation explicitly against each other. Therefore, no conclusion can be drawn on their relative performance. However, Fraysse *et al.* [57] compares τ-extrapolation against output-based and feature-based adaption for a variety of transonic airfoil flows. It was found that in terms of error versus cell count, τ-based adaption performs slightly worse than adjoint output-based adaption but better than feature-based adaption. Syrakos *et al.* [58] found that a more efficient error function is obtained when the truncation error is multiplied with the volume of the cell.

2.6.3. Error transport equation (ETE)

An equation can be derived that describes the transport of the discretization error throughout the computational domain. By, e.g., considering the incompressible Navier-Stokes equations:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j x_j}, \tag{2.37}$$

one can define the exact solution as the sum of the solution of the discrete problem and an error:

$$u = u_h + \epsilon, \qquad p = p_h + \epsilon_p. \tag{2.38}$$

Substituting this yields:

$$\frac{\partial \epsilon_i}{\partial t} + (u_h)_j \frac{\partial \epsilon_i}{x_j} + \frac{1}{\rho}\frac{\partial \epsilon_p}{\partial x_i} - \nu \frac{\partial^2 \epsilon_i}{\partial x_j x_j} + \epsilon_j \frac{\partial (u_h)_i}{x_j} + \epsilon_j \frac{\partial \epsilon_i}{x_j} = -R(U_h), \tag{2.39}$$

where $R(U_h)$ is the differential residual, which arises since the discrete solution U_h does not satisfy the continuous problem exactly. One can see that the equations for the ETE are the same as for the primal problem but with an additional source term. In the case of a Dirichlet boundary condition of the primal problem, the error is set to zero for the ETE problem. The Neumann boundary conditions are treated the same as in the primal problem. Since the differential residual contains the continuous operator R(), it cannot be computed directly. The key is to find a proper approximation for it. Options are, for example, a higher-order discrete operator or a higher-order reconstruction operator [46, 59].

By solving the ETE, one obtains the discretization error ϵ, similar to the Richardson-extrapolation approach. For an unsteady simulation, the primal problem and the ETE can both be advanced simultaneously. Where simultaneously means that the ETE lags a certain number of time steps behind the primal problem so that the time derivative can be evaluated. The ETE can also be solved sequentially after the primal problem has been solved. This, however, does not make much sense when using AMR during the simulation, and additionally, the entire solution history must be saved, which would require a tremendous amount of storage or memory.

2.7. Goal-oriented error indicators/estimators

The third big family of AMR error functions is built around the idea of goal-based adaption. Instead of reducing the error in the entire domain, only the part of the mesh is refined, which contributes to the actual error in a quantity of interest (functional). In a flow problem like

$$R(U, \alpha) = 0 \tag{2.40}$$

and a functional J dependent on the mesh which is denoted by α

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial U}\frac{dU}{d\alpha} + \frac{\partial J}{\partial \alpha} \tag{2.41}$$

one can formulate the adjoint problem as

$$\left[\frac{\partial R}{\partial U}\right]^T \Psi = \left[\frac{\partial J}{\partial U}\right]^T, \tag{2.42}$$

here $dJ/d\alpha$ denotes the total derivative. Instead of the highly inefficient approach to adopt a mesh cell (α) and evaluate $\partial U/\partial \alpha$, which requires the recomputation of the entire problem, the adjoint approach allows to efficiently calculate the influence of any α on one functional. Using the adjoint, however, means that a change in the functional will always require the solution of a new adjoint

problem. Since there are more cells in the domain than quantities of interest an engineer would like to investigate, solving an adjoint problem is very advantageous. Solving a problem with AMR can require a couple of subsequent mesh adaption steps. The adjoint equation has, therefore, to be solved multiple times, which is very expensive. Ding [60] shows an approach where the computational cost is drastically decreased by smartly reusing parts of the old adjoint solution. Other acceleration techniques have e.g., been proposed by Tyson *et al.* [61]. An influential method of adaptive mesh refinement using the adjoint approach was introduced by Venditti and Darmofal [9] based on the works of Pierce and Giles [62]. This paper demonstrates a practical implementation of goal-based adaptive mesh refinement.

By considering the objective J on a coarse and fine mesh one can show that the difference in a functional between both meshes is the adjoint-weighted residual:

$$J_h(U_h^H) - J_h(U_h) \approx \underbrace{(\Psi_h^H)^T R_h(U_h^H)}_{\text{Computable correction}} + \underbrace{\left[R_h^\Psi(\Psi_h^H)\right]^T (U_h - U_h^H)}_{\text{Error in computable correction}} \tag{2.43}$$

with R_h^Ψ being

$$R_h^\Psi = \left[\left.\frac{\partial R_h}{\partial U_h}\right|_{U_h^H}\right]^T \Psi_h^H - \left(\left.\frac{\partial J_h}{\partial U_h}\right|_{U_h^H}\right)^T . \tag{2.44}$$

Another advantage is the defect correction, which is obtained 'for free' after the adjoint problem has been solved. With the defect correction, the error in the functional can be decreased [63]:

$$\tilde{J}_h(U_H) = J_h(U_h^H) - (\Psi_h^H)^T R_h(U_h^H) . \tag{2.45}$$

For a transient simulation, the adjoint equation changes slightly [11]. To derive the unsteady set of adjoint equations, we consider the unsteady residual

$$R(\bar{U}) \equiv M\frac{dU}{dt} + R(U) = 0 \tag{2.46}$$

and the unsteady functional

$$\bar{J} \equiv \int_0^T J(U(t),t)dt + J_T(U(t)) , \tag{2.47}$$

one can obtain

$$M\frac{\delta\Psi}{\delta\tau} + \left[\frac{\partial R}{\partial U}\right]^T \Psi + \left[\frac{\partial J}{\partial U}\right]^T = 0 , \tag{2.48}$$

where $\tau = T - t$, and M is the mass matrix. The transient adjoint problem is, in principle, the steady adjoint problem plus another term accounting for unsteady effects. One can see that the adjoint has to be solved backward in time. This is a critical property of the adjoint in unsteady problems. As stated in the previous section, it is very costly to solve the adjoint equations. Whereas before, the adjoint equation is only solved when required for a mesh adaption, now the adjoint equation has to be solved every single time step, even if one is merely interested in adaption occasionally. Since it must be solved backward in time, the solution of all time steps must be stored as well. The high computational cost in a transient setting makes this approach unsuitable for LES. For steady-state problems, e.g., RANS, goal-based adaption can, however, be seen as a compelling method and can be potentially considered to be the best approach to calculate the error sensor. Other approaches found in the literature of using adjoint for AMR include the combined ETE-adjoint approach by Tyson *et al.* [61], the entropy-based adjoint problem of Fidkowski and Roe [64] and the goal-oriented dissipation-based error function from Dwight [65, 66].

2.8. Flag functions

In case one has an 'ideal' error estimate at his disposal, one still has to decide on how to adapt the mesh. The flag function takes the error and other information such as the cell size as the input and outputs the adaption action, i.e., for an H-refinement routine if a cell should be refined, coarsened, or kept in its original size. The equidistribution principle by de Boor [67] is a popular approach used in the AMR world. The aim is to create a mesh in which every cell has the same error sensor value. The assumption is thus made that all individual errors in the computational domain are equally crucial for the simulation. Depending on the goal of the simulation, this can be over-conservative e.g., when only the accuracy of a functional is considered. For H-refinement, in contrast to R-refinement, one has only discrete adaption steps, and thus equidistribution cannot be achieved precisely.

A flag function can be constructed to use the raw values of the error sensor to flag cells for adaption. This is, however, not necessarily a smart choice because error sensors are often dimensional. Therefore a common approach is to non-dimensionalize the output from the error function by rescaling it between zero and one [43]:

$$\Psi^* = \frac{\Psi - \Psi_{\min}}{\Psi_{\max} - \Psi_{\min}}. \tag{2.49}$$

Other approaches are presented by Hertel *et al.* [19]. Rescaling also has drawbacks. In case of a large difference between the smallest and largest error indicator value e.g., due to an outlier, such a rescaling operation will compress the bulk of the error indicators. Thus rescaling improves the predictability of the error values but can also introduce new problems. Another popular flag function is the fixed fraction approach. Instead of determining fixed error bounds, one merely specifies that every adaption step a fixed fraction of cells, e.g., with the highest error, is flagged for refinement [68]. Dannenhoffer, III and Baron [32] proposes to automatically select the fixed fraction threshold based upon a knee point in the error sensor histogram.

Instead of changing the mesh level by incremental steps (in H-adaption), multilevel adaption aims at changing the mesh size by more than one step per adaption iteration. When the rate of convergence ω of the underlying discretization is known a priori or was determined by e.g., Richardson-extrapolation, one can derive the target mesh size in a simple way [9]:

$$h_H = h_h \left(\frac{\epsilon}{\epsilon_0} \right)^p, \tag{2.50}$$

where h_H is the mesh size before adaption and h_h the mesh size after an adaption, ϵ is the error level, and ϵ_0 the target error level. It is the author's opinion that a good flag function has the same order of importance as the error sensor itself. While there are some good works available that treat the flag function such as Aftosmis and Berger [1], Nemec *et al.* [69], there was no literature found that rigorously reviewed and investigated this aspect. Perhaps also because a clear demarcation between the error sensor and flag function is not always possible since both are very much interconnected. Nevertheless, an adaption routine comprised of an 'ideal' error sensor, but a 'bad' flag function cannot yield good mesh adaption. With the more and more widespread use of CFD, it can not be assumed that every CFD user is a CFD-expert. Therefore, it would not only be important to see which flag function can theoretically lead to the best result but also to investigate which flag function in the hands of an (inexperienced) user leads to the best practical result. One can conclude that the flag function is an integral part of a user-independent AMR routine and thus overall CFD experience.

2.9. Synthesis and chosen direction

One primary outcome of this literature study was the realization that it is tough to compare different approaches with each other. This is mainly due to unstandardized approaches within the literature.

Sometimes merely the output of the error sensor is compared to e.g., the exact error, while in other instances, more practical applications are shown [19, 31]. Therefore, it is not very easy to judge from literature which approaches lend itself the most for achieving user-independent AMR. Since this thesis also has the aim to test novel error sensors, the question naturally arises which of the three families of error sensors should be used and developed further. To make this decision somewhat systematic, a list of properties has been developed that entails properties that describe, to the author's opinion, an *ideal error sensor*. This list is partly influenced by [31].

An ideal error sensor should

1. be order(s) of magnitude cheaper to compute than the primal solution, irrespective of whether the simulation is steady-state or transient.
2. be independent of the flow problem at hand. Work for a wide range of very different CFD cases.
3. require virtually no input from the CFD user.
4. be able to quantify the spatial, temporal, and modeling error reliably. Rather over-estimate than under-estimate the error. [1]
5. be based on a rigorous mathematical basis and not heuristic in nature.
6. be independent of the underlying solver or the numerical techniques used. Implementation should not be prohibitively complicated.
7. work equally well for refinement and coarsening.
8. be numerically robust and stable.
9. give an a priori estimate on the required CPU time to obtain a certain error level.
10. be able to either target the error of a functional or the error of the entire flow field.
11. give reliable results also on coarse meshes.
12. lead to a computationally efficient mesh.

With these criteria used as a metric, Table 2.1 was created. In this table, the general properties of feature, numerical, and output-based error functions are given. The information was compiled from the findings of this literature study. Novel ideas and improvements to these error functions might, therefore, change their respective properties.

From the table, it becomes clear that there is no single best error sensor. Instead, every group has their respective strengths and weaknesses. Feature-based indicators are usually easy to implement, cheap to compute, and can, when carefully chosen, lead to excellent adaption. However, their successful application is very much dependent on the correct choice for a particular flow problem. A requirement that can usually only be fulfilled when a wealth of a priori knowledge is available and thus makes this method unsuitable for user-independent refinement.

Error estimators are, in this respect, much more independent of the problem. Their property of establishing a thorough mathematical link with the actual error is an important point when fielding such an error sensor. An AMR routine can be justified much better, e.g., in an engineering context, when it is based upon a thorough mathematical foundation than in comparison to a completely

[1] Even though an ideal error estimator would always report the exact error, this is in practice not achievable. A slight over-estimation in the error value is, in such a case, advantageous. It will lead to more conservative adapted mesh i.e., with more cells than necessary, however this way, it can be guaranteed that the problem is not underresolved.

Table 2.1: Summary of the properties of feature-based, numerical and output-based error functions. The results are based on the findings from the literature study.

Property	Feature-based	Numerical	Output-based
1	Cheap in all cases	From cheap to expensive, the latter especially for LES	Expensive, same order as primal problem, prohibitively expensive for transient cases
2	Usually highly specific to the problem at hand	Can be completely problem independent	Only functional is problem specific
3	Specific error sensor must be selected.	Mostly only error thresholds	Functional must be selected a priori
4	Only heuristic indications	Can provide for all a reliable estimate, difficult for LES	Spatial and temporal
5	Usually heuristic, some based on problem specific derivations	Rigorous mathematical foundation if the mesh is already very fine	Rigorous mathematical foundation
6	Independent	Substantial amount of FEM specific methods	Independent to specific
7	No information	No information	No information
8	Robust and stable	No information	Adjoint problem can pose challenges
9	Possible precomputation with RANS	No	No
10	Only entire flow field	Only entire flow field	Functional and flow field
11	No problems reported in literature, seem to work also on coarse meshes	Richardson-like methods: needs asymptotic range	Coarse mesh might lead to erroneous results, especially in defect correction
12	Mesh quality depends on a specific error sensor for a specific problem	There is indication that the resulting mesh efficiency can lie between feature-based and output-based adaption [57]	Can lead to very a efficient mesh

heuristic error indicator that seems to work okay but without proper explanation why it is doing so. A potential challenge, however, is the need for Richardson and $\tau-$extrapolation to have at least two solutions located within the asymptotic range. The question is thus how these methods will perform in complex industrial flow applications. There is literature that treats Richardson-extrapolation in a LES application, but it cannot be regarded as plentiful. As a matter of fact, for τ-extrapolation, no literature was found at all that also used LES. A question for LES is also how to forecast the modeling error, while Richardson-extrapolation can potentially achieve this for explicitly filtered LES it always

comes with the considerable drawback of an additional problem computation on an equal-sized grid, thus significantly increasing the computational cost. Concluding, even though error functions are more thoroughly formulated, the application to LES does not seem to be trouble-free. They might, however, be a competitive candidate for a user-independent adaption routine. Their computational cost is not negligible, especially for ETE type error sensors.

The big advantage of output-based AMR methods has to be seen in their ability only to adapt regions of the mesh, which are absolutely necessary for the error of the functional, a property that can neither be found in feature-based nor numerical error sensors. At first glance, it, therefore, seems to be the ideal pick for an error function, especially because it usually comes with a free and powerful defect correction that can significantly enhance the accuracy of the functional. Also, output-based error functions can be considered to be problem independent in the sense that very little knowledge is needed about the fluid conditions in the flow problem. The engineer must merely know what the functional should be. However, with the high cost in transient problems, the original goal, namely saving CPU time and memory can be described as optimistic.

Based on the gathered literature, it was decided to continue with numerical error estimators. Coarse meshes might impose a challenge for these groups of error sensors, but they are the only ones that, at the same time, can combine user-independence with affordability. Output-based error sensor have the potential to be the *ideal* error sensor. However, for LES, the memory requirements are unfortunately just way too high. Feature-based indicators are perhaps the best *practical* error sensors for specific problems. But exactly due to this property, they exclude themselves due to the goal of user-independence. Generic LES specific error indicators, such as resolved to total TKE, were not further considered due to their apparent poor performance.

3

Theoretical framework

In this chapter, the theoretical framework is laid out. First, an introduction into the CFD code INCA is given. Subsequently, the H-adaption approach, the error sensors, and the flag function, which are used within this thesis, are discussed. Last but not least the prolongation and restriction operators are treated.

3.1. INCA solver

For this thesis, the INCA (http://www.inca-cfd.com/) CFD code is used, a general-purpose CFD solver for LES and DNS. It can solve the compressible and incompressible Navier-Stokes equations on collocated and staggered Cartesian structured grids. It contains numerous advanced features such as combustion and cavitation modeling. Within the scope of this thesis, the compressible Euler equations have been solved for the Mach 3 shock over a forward-facing step in Chapter 4. The incompressible Navier-Stokes equations were solved for the flow over the two-dimensional cylinder and the flow over periodic hills in Chapter 5 and 6 respectively. For the compressible simulation, a collocated grid with AMR refinement factor 2 was used, whereas, for incompressible simulations, a staggered grid was employed with refinement factor 3. For the discretization in space, the ALDM (Adaptive Local Deconvolution method) presented in Hickel *et al.* [70] was used. The time-stepping is performed with a third-order Runge-Kutta TVD scheme as described in Gottlieb and Shu [71]. The Poisson-equation arising in the incompressible cases is solved using the BiCGSTAB method [72] with a SIP pre-conditioner [73]. The method used to solve the compressible case is described in Hickel *et al.* [74].

3.2. INCA's block-structured mesh

The INCA solver is built around a block-structured mesh and supports H-adaption. INCA's mesh is strictly block-structured, meaning that every block is perfectly placed along the boundaries of its neighboring blocks. INCA's block connectivity is explicitly stored within every block's data type and is, for example, not implicitly derived by an oct-tree. With the H-adaption routine, it is possible to refine, coarsen, split, and merge blocks. The mesh of every block can be described by a level. An unaltered block has, by definition, a level of 0, every single refinement step incrementally increases the level. INCA supports anisotropic meshes as well. In that case, every block can have a different level in all of its three principal directions. Since INCA uses a structured mesh, it cannot conform directly to any arbitrarily shaped wall in the fluid domain (apart for any rectangular geometry aligned with the mesh's principal directions and block boundaries). A second-order conservative immersed interface method is used to circumvent this limitation. An elaborate discussion can be found in Meyer *et al.* [5].

3.3. Auxiliary adaption criteria

While a good solution-based adaption criterion is essential for any successful deployment of AMR in a CFD code, an additional aspect is the necessity for 'auxiliary' criteria, which are solely needed for practical purposes. Despite their heuristic nature, they are crucial for the working of INCA's adaption routine. In the following, the auxiliary criteria used in this thesis are presented.

3.3.1. Alternating block splitting and refining criterion

Even though INCA supports meshes that have jumps in the level of more than one between two adjacent blocks, for INCA's adaptive mesh refinement, the maximum difference in block level is set to one. This requirement is born due to practical reasons. Since, in a block-structured mesh, all adaption actions can only act on the entire block, an effective H-adaption routine must entail block splitting. Otherwise, especially when there is only one starting block, all adaptive mesh refinement would be equivalent to just uniform refinement action on the entire domain. One necessity for block-splitting is, however, that there are enough cells within a block. It is trivial to see that in the limit case of a block merely containing only one single cell, every splitting attempt is futile. Therefore, in INCA an alternating approach is chosen where after every mesh refinement a splitting action must follow.

3.3.2. Cut-cell criterion

The cut-cell adaption method identifies all cells which contain an immersed boundary and subsequently flags them for refinement when their cell size is above a specific limit. It thus represents a refinement criterion based upon geometric considerations. Even though this criterion is not based upon any mathematical foundation, it purely exists due to practical reasons. When an initial mesh is created with a starting cell size much larger than the characteristic length of the geometry, e.g., the diameter of a cylinder, then the immersed boundary approach fails. Therefore, a geometry must be resolved by at least a couple of cells before more sophisticated refinement criteria can be used.

3.3.3. Balance criterion

The balance criterion ensures that a block has along either of its boundary surfaces only neighboring blocks of the same mesh level. A violation of this criterion is visualized in Figure 3.1. A large block is shown in (A). Along its right boundary, it has two neighboring blocks of different mesh sizes (B and C). Such a mesh is not allowed in INCA since it only supports ghost cells with a constant normal mesh size along a block boundary. The balance criterion will, therefore, flag the large coarse block (A) to be split and thus ensures that all ghost cells have a constant normal size along every boundary of a block.

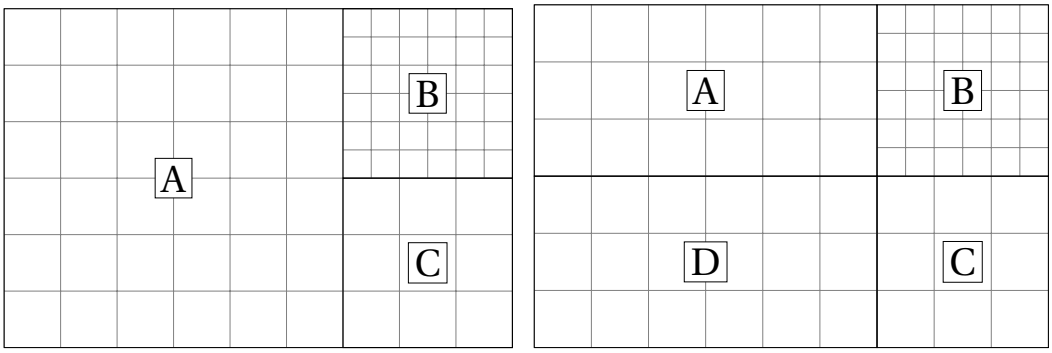


Figure 3.1: Visualization of a mesh violating the balance condition. Left before balance criterion, right after balance criterion has been invoked.

3.4. The H-adaption pipeline

The H-adaption pipeline entails all the procedures which are required for a successful AMR step. It is shown in Figure 3.2 and Algorithm 1. When using error estimates, the setup is as follows: Two simulations, a coarse and a fine one, are computed simultaneously. Both are separated by exactly one mesh level. Advancement in time is done with the same time step τ. This ensures that no interpolation in time is required when the error estimate, e.g. Equation 3.5, has to be computed. Due to the synchronization, the coarse simulation is advanced with a smaller CFL number. Theoretically, this means that the spatial and temporal error cannot be separated distinctively. However, practically it is assumed that the discretization error in space is more significant than the temporal error. For all test cases, a CFL number of one is used for the fine simulation. Temporal errors should, therefore, generally be rather small. After an elapsed time of τ_{AMR}, the adaption procedure is invoked. Due to the discretization in time, adaption does not happen exactly at τ_{AMR} but rather at the first time step that has surpassed the time τ_{AMR}. After the execution of the adaption procedure outlined in Algorithm 1, the time step τ and the fine state vector are sent to the coarse simulation. This resets the coarse simulation to the state of the fine one, any misalignments in the solution that happened during the time τ_{AMR} are that way eradicated. Finally, the same block splitting and refinement action is performed on the coarse and fine mesh. In the case of turbulent flow, the procedure mentioned above is slightly altered. Error sensor evaluation and solution exchange still take place every τ_{AMR},

but adaption does not. That way, error sensor samples are taken, and a mean can be calculated. This is important for turbulent flow since it is too chaotic to obtain a smooth error sensor distribution, see Chapter 6. Adaption happens as usual after N error sensor samples have been taken. For the evaluation of the reference error sensor, the coarse simulation is not required. The remaining part of the adaption pipelines remains unchanged.

Algorithm 1 shows all tasks that are performed during an AMR call. During the very first AMR call, the cut-cell criterion is invoked to guarantee a minimum mesh resolution of the immersed boundary. Otherwise, the first step is the calculation of the error sensor. Subsequently, the flag function will calculate an adaption action based upon the error sensor output. The adaption action is then checked against the 'alternating refinement-split criterion' and, if necessary, changed accordingly to ensure alternating block splitting and refining. Now the block is either split or refined. In the end, the 'balance criterion' checks the mesh and can also call for a refinement and split action to ensure proper grid connectivity. The entire adaption pipeline is traversed twice. This ensures that a bad block will definitely be refined. If the pipeline is only traversed once, then a block might have been identified for refinement, but due to the 'alternating refinement-split criterion' is split first. Thus, a second pass will capture this block again and then subsequently refine it.

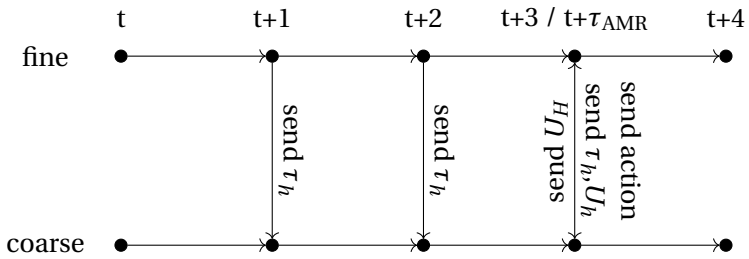


Figure 3.2: Overview of the overall adaption routine for error estimate based adaption.

Algorithm 1 H-adaption procedure.

```
for after every τ_AMR do
    for all blocks do
        if first ever AMR call then
            cut-cell criterion
        end if
        calculate error sensor
        calculate flag function
        invoke alternating criterion
        perform block split or block refine
        invoke balance criterion and if necessary adapt the mesh
        reset coarse simulation and send adaption action
    end for
end for
```

3.5. Error sensor

For this thesis, error sensors based on error estimates are used. First, the classical 'Richardson-extrapolation' error estimate is presented. Subsequently, new formulations for an error estimate based upon the error transport equation are described.

3.5.1. Richardson-extrapolation

Using the approach of Richardson-extrapolation, an estimate of the actual error of the solution can be obtained:

$$\epsilon \approx \left| U_H - U_h|_h^H \right| . \tag{3.1}$$

Where $U_h|_h^H$ is the restriction of the fine solution on the coarse mesh. In this thesis, the velocity magnitude has been used for U. This error estimate is not only used for the Richardson-type error sensor but also for the ETE terms and therefore constitutes an essential ingredient. For practical reasons, the output of this error estimate was set to zero when an immersed boundary intersected a coarse grid cell. The reason being that corresponding fine mesh cells may lie outside, on the or in the immersed boundary. The choice of an appropriate restriction operator becomes then non-trivial and would otherwise lead to erroneously high errors:

$$\phi = \left| u_H - u_h|_h^H \right| . \tag{3.2}$$

While this estimate is helpful for a *particular* simulation, it is probably not handy when considering a variety of different problems. The magnitude of this error estimate deviates based upon which state variable is used e.g., pressure or velocity. Further, the magnitude also deviates based upon the problem at hand, such as low versus high Reynolds number flow. Next to this, errors, in regions where the magnitudes of the state variables are significant, will usually be more substantial than in regions of low magnitude. Thus, errors in small scale structures are rendered unimportant. Therefore, the aim should be to non-dimensionalize the error estimate to make it independent of the problem. A simple approach is to calculate the relative error:

$$\phi = \frac{\left| u_H - u_h|_h^H \right|}{\left| u_h|_h^H \right|} . \tag{3.3}$$

While this approach provides an effective non-dimensionalization, it comes with the danger of causing singularities in regions where the state variable is zero. An obvious example is the flow velocity close to a wall. In these regions, unrealistically large error estimates are produced. To remedy this issue, the relative error equation is adjusted to:

$$\phi = \frac{\left| u_H - u_h|_h^H \right|}{\max\left(\left| u_h|_h^H \right| , 0.1|u_{\text{ref}}| \right)} . \tag{3.4}$$

This is a typical approach that is also shown in other references, such as in Hertel *et al.* [19]. Alternatively one can also merely use the reference velocity:

$$\phi = \frac{\left| u_H - u_h|_h^H \right|}{u_{\text{ref}}} , \tag{3.5}$$

yielding an non-dimensional absolute error estimate. Last but not least, an error sensor is proposed, which multiplies the error in velocity magnitude with the cell size. With the finite volume method in mind, the idea is that an error in a tiny cell might only have a minor impact on the solution. However, this approach might also lead to erroneous results, e.g., when the bad cell is located in a boundary layer or when there is a large cluster of cells with a high error that is artificially reduced by the cell volume of each individual cell. The error sensor thus becomes:

$$\phi = \left| u_H - u_h|_h^H \right| dV . \tag{3.6}$$

Similarly Equation 3.6 can be non-dimensionalized as:

$$\phi = \frac{\left| u_H - u_h|_h^H \right| dV}{u_{\text{ref}} L_{\text{ref}}^3} . \tag{3.7}$$

3.5.2. Error transport equation

A set of novel error sensors is proposed based on the individual terms of the Error transport equation depicted in Equation 3.8. Instead of solving the ETE directly to obtain the error estimate ϵ, the aim is to use the individual terms as an error sensor. The hope is to highlight regions with high error transport and production and see what kind of influence these regions have on the solution accuracy. The ETE is

$$\frac{\partial \epsilon_i}{\partial t} + (u_h)_j \frac{\partial \epsilon_i}{x_j} + \frac{1}{\rho}\frac{\partial \epsilon_p}{\partial x_i} - \nu \frac{\partial^2 \epsilon_i}{\partial x_j x_j} + \epsilon_j \frac{\partial (u_h)_i}{x_j} + \epsilon_j \frac{\partial \epsilon_i}{x_j} = -R(U_h) = 0\,. \tag{3.8}$$

When the size of the grid becomes smaller, the discrete solution u_h will move towards the exact solution u, and thus the error vanishes. The discrete solution, however, can only approach the exact solution when, in the limit case, all terms involving the error in the momentum equation vanish. The three chosen terms stem from the advection term of the Navier-Stokes equations. They are:

$$\phi = \sum_{i=1}^{3} \left| \epsilon_j \frac{\partial \epsilon_i}{x_j} \right|, \tag{3.9}$$

$$\phi = \sum_{i=1}^{3} \left| \epsilon_j \frac{\partial (u_h)_i}{x_j} \right|, \tag{3.10}$$

$$\phi = \sum_{i=1}^{3} \left| (u_h)_j \frac{\partial \epsilon_i}{x_j} \right|. \tag{3.11}$$

Equation 3.9 and 3.10 can be considered to be production terms while Equation 3.11 is an advection term. Since Equation 3.9 contains the multiplication of two errors, it remains to be seen if this term is negligibly small. Absolute values are used to avoid error cancellation. The error ϵ in the respective terms is computed using the Richardson-extrapolation approach. The error in the state variables can be approximated as:

$$\epsilon_i \approx u_{H,i} - u_{h,i}|_h^H\,. \tag{3.12}$$

The derivatives were calculated with a simple second-order central finite difference scheme. The dimension of this error sensor is:

$$[\phi] = \frac{L}{T^2}\,, \tag{3.13}$$

where L and T are the length and time dimensions, respectively. A straightforward non-dimensionalization of this error estimate comprises the reference values u_{ref} and L_{ref}:

$$\hat{\phi} = \frac{\phi}{\frac{u_{\text{ref}}^2}{L_{ref}}}\,. \tag{3.14}$$

3.5.3. Reference error indicator

The aim of this thesis is not only to show that a particular error sensor can be used to obtain a solution that is computational more efficient than simple uniform refinement. It is also to show that the proposed error sensors hopefully perform better than a typical reference error sensor. Since INCA is second-order accurate, the first derivative of the state variables should be exact. Therefore, the second undivided difference in velocity magnitude has been chosen (see Section 2.5.1):

$$\phi = \max\left(\frac{\partial^2 u_i}{\partial x_i^2} (h_i)^2 \right). \tag{3.15}$$

The derivative was calculated using a simple second-order central finite difference scheme. The error indicator was calculated for x and y-direction, respectively, and then passed to the flag function. No non-dimensionalization has been implemented.

3.6. Flag function

The task of the flag function is to use the error sensor output of every cell and convert it into a mesh adaption action. This happens according to two different criteria, one based upon considerations regarding the block-based mesh approach and one purely based upon the error sensor.

Four main operations can be performed on a block: merging, splitting, refining, and coarsening. Since coarsening is not considered for this thesis, the flag function has the authority to decide whether a block should stay as it is, be split or refined. Block splitting is essential to reduce the number of cells in a block prior to a possible mesh refinement action.

Algorithm 2 Refine when exceeding threshold.

```
1: for all blocks do
2:     if (φ >= threshold) .and. (was last time split) then
3:         flag block for refinement
4:     else if (φ >= threshold) .and. (was last time refined) then
5:         flag block for splitting
6:     else
7:         leave block unchanged
8:     end if
9: end for
```

The easiest way of flagging a cell for refinement is by checking if it violates a simple error threshold (Algorithm 2). This approach comes, however, with the drawback that even a minute violation of the threshold from a single cell can lead to the refinement of a potentially huge block. Even though one can argue that the error in a single cell might numerically pollute the entire fluid domain, for practical reasons, it is desirable to filter out these cells. This is especially true when the error in the cell is not present for the entire solution time but only at single time steps. Therefore, the following algorithm is proposed, where a block is only flagged for refinement when a certain percentage of cells within the block violates the error threshold. There are instances, however, where this approach can lead to problems. If a block merely clips a region if interest e.g., an immersed boundary, adaption might be prevented. Another example would be a block that encompasses larges regions of the fluid domain, including a wall with a boundary layer. The number of cells in the boundary layer might be very small in comparison to the total cells in the block.

Algorithm 3 Refine when enough cells exceed threshold.

```
1: for all blocks do
2:     calculate fraction of cells having: (ϵ >= threshold)
3:     if (fraction >= fraction threshold) .and. (was last time split) then
4:         flag block for refinement
5:     else if (fraction >= fraction threshold) .and. (was last time refined) then
6:         flag block for splitting
7:     else
8:         leave block unchanged
9:     end if
10: end for
```

Directionality is straight forward to implement in the flag function when this information is already contained in the error sensor. This is e.g., the case for the reference error sensor whose second derivatives can be calculated in all three principal directions. For the Richardson and ETE error estimates, this directional information is not available and must, therefore, be deduced from an-

other source. The proposed approach is to use the directional information from the reference error sensor.

Algorithm 4 Directionality algorithm for the flag function.

for all blocks **do**
  calculate: $\phi_1 = u_{x_1 x_1}\Delta x_1^2$, $\phi_2 = u_{x_2 x_2}\Delta x_2^2$, $\phi_3 = u_{x_3 x_3}\Delta x_3^2$
  if 1D **then**
    refine or split when allowed in direction of max(ϕ_1, ϕ_2, ϕ_3)
  else if 2D **then**
    refine or split when allowed in direction of max(ϕ_1, ϕ_2, ϕ_3)
    refine or split when allowed in direction of second_max(ϕ_1, ϕ_2, ϕ_3)
  else
    refine or split when allowed in all directions
  end if
end for

Another strategy employed by the flag function is to limit the amount of refinement for every AMR iteration. A similar strategy to the one outlined in Aftosmis and Berger [1] is followed. A mesh adaption routine that refines all cells which have been flagged for refinement will lead to rapid error reduction. However, this reduction does not act 'uniform' on the error distribution in space. This is illustrated in Figure 3.3, which shows a histogram of all error sensor values for a generic flow problem. When refinement acts on all cells that exceed the refinement threshold, all of these errors will reduce and thus eventually, after a couple of refinement actions, fall below the error threshold. However, the figure shows that the cells with the most significant error will be the latest to fall below the threshold. This means that the error is aggressively reduced in the low-error regimes of the domain, but the high-error regimes are only tackled conservatively. The practical consequence is that the cell-count in the domain will have inflated very quickly without having pushed the largest errors below the threshold. The alternative approach is to target the highest error-regimes first while neglecting low-error cells specifically. This strategy is depicted in Figure 3.4. Aftosmis and Berger [1] suggests using a decreasing refinement threshold. The initial threshold is set quite high, thus leading to refinement of only the worst cells. When the set of cells violating the threshold drops to zero, the threshold is adjusted to target the next set of cells. In this thesis, a similar approach is used: A fixed percentage of bad cells is refined every iteration step [64]. The exact workings are outlined in Algorithm 5. First, all blocks that exceed the error threshold will be sorted according to their error estimate. Subsequently, the worst blocks are chosen, selection stops when the total number of cells contained in the blocks exceed a certain percentage threshold with respect to the cell count in the blocks that contain bad cells. For every chosen block the refinement will happen as determined from the previous parts of the flag function, for every non-chosen block the refinement action will be overwritten and denied.

3.7. Restriction operators and 'destaggering'

For the calculation of the error estimate in, for example, Equation 3.5, the restriction of the fine solution on the coarse grid is required. This is realized by the restriction operator I_h^H:

$$U_h|_h^H = I_h^H U_h\,. \tag{3.16}$$

For this task, a straightforward restriction operator has been chosen. The mean of all fine cells is computed, which are encompassed by the corresponding coarse cell:

$$U_h|_h^H = \frac{\sum_{i=1}^{N} U_{h,i}}{N}\,. \tag{3.17}$$

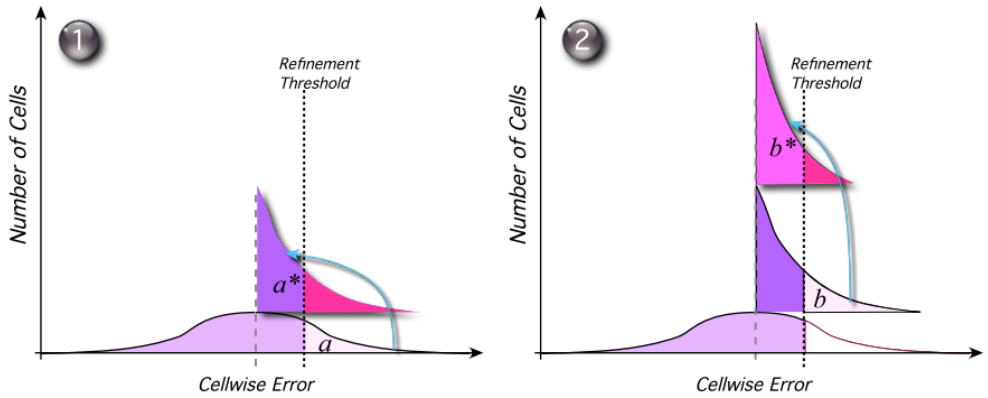


Figure 3.3: Common flag strategy that does not give priority to the cells with the highest error. Figure from Aftosmis and Berger [1].

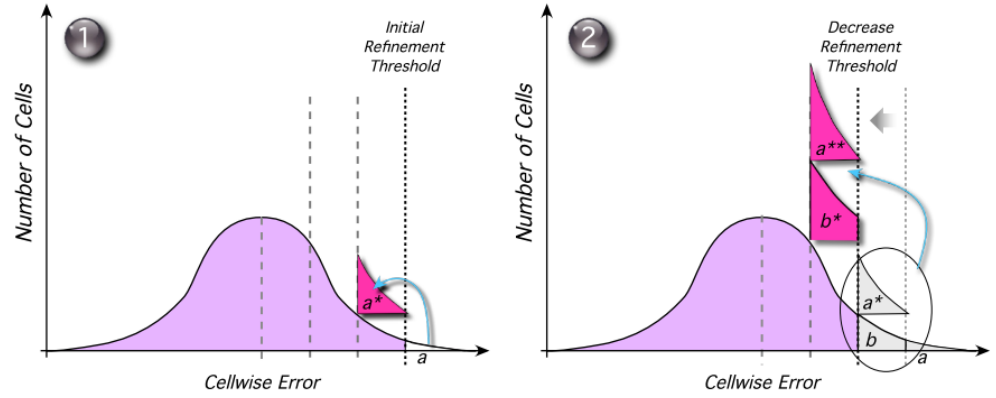


Figure 3.4: Flag strategy in which the largest errors are tackled first. Figure from Aftosmis and Berger [1].

Algorithm 5 Ordered importance refinement for the flag function.

```
sort: all blocks that exceed the error threshold in decreasing order of their error estimate
pick: blocks from the top of the list until 10% of all bad cells are chosen
for all blocks do
    if chosen then
        block will be refined
    else
        refinement is denied
    end if
end for
```

This method can be directly used when the corresponding values are cell-centered. For the incompressible cases, a staggered mesh is employed, which means that the velocities are edge-based. A 'destaggering' method has been used to transform the edge-based velocities to cell-centered ones. The reasoning behind this was first based upon programming considerations and secondly on the thought that this operation would also slightly smooth out the error and thus avoid error spikes, which might lead to erroneous adaption. For the x-velocity the 'destaggering' operation is:

$$u_{i,j,k} = \frac{u_{i-\frac{1}{2},j,k} + u_{i+\frac{1}{2},j,k}}{2}. \tag{3.18}$$

Due to reasons of how the staggered mesh is implemented within INCA, the method changes at the first index of all principle directions to:

$$u_{1,j,k} = \frac{3u_{1+\frac{1}{2},j,k} - u_{1+\frac{3}{2},j,k}}{2}. \tag{3.19}$$

The method works analogously for the y- and z-direction. A graphical representation of the index definitions is shown in Figure 3.5.

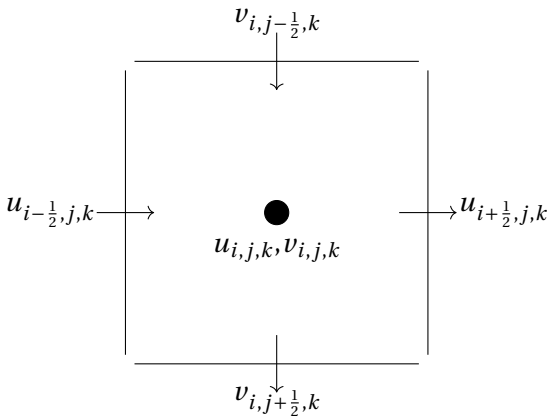


Figure 3.5: Visualization of the 'destaggering' method used to bring edge/face-based velocities to the cell center. Three-dimensional sketch is analogous.

Every time the H-adaption pipeline is executed, also when ultimately no adaption happens, the fine solution state vector is copied to the coarse simulation. There it replaces the old state vector and thus resets the coarse simulation to the fine state. For all cell-centered variables, this happens according to Equation 3.17. In the case of edge-based velocities, a different approach is used to maintain the conservation of mass. The restriction operator works by taking the mean of the edge-based velocities that coincide with the boundary of a coarse cell. An example is given for the x-velocity in a two-dimensional mesh. The other operations are analogous for y and z-direction, as well as when considering three dimensions. For the left intersecting boundary

$$u_{H,i-\frac{1}{2},j,k} = \frac{u_{h,i-\frac{3}{2},j-1,k} + u_{h,i-\frac{3}{2},j,k} + u_{h,i-\frac{3}{2},j+1,k}}{3} \tag{3.20}$$

is employed, while for the right intersecting boundary

$$u_{H,i+\frac{1}{2},j,k} = \frac{u_{h,i+\frac{3}{2},j-1,k} + u_{h,i+\frac{3}{2},j,k} + u_{h,i+\frac{3}{2},j+1,k}}{3} \tag{3.21}$$

was used. A visualization of the notation is given in Figure 3.6. The indices i,j,k either belong to the fine or the coarse mesh depending on whether subscript h or H is used.

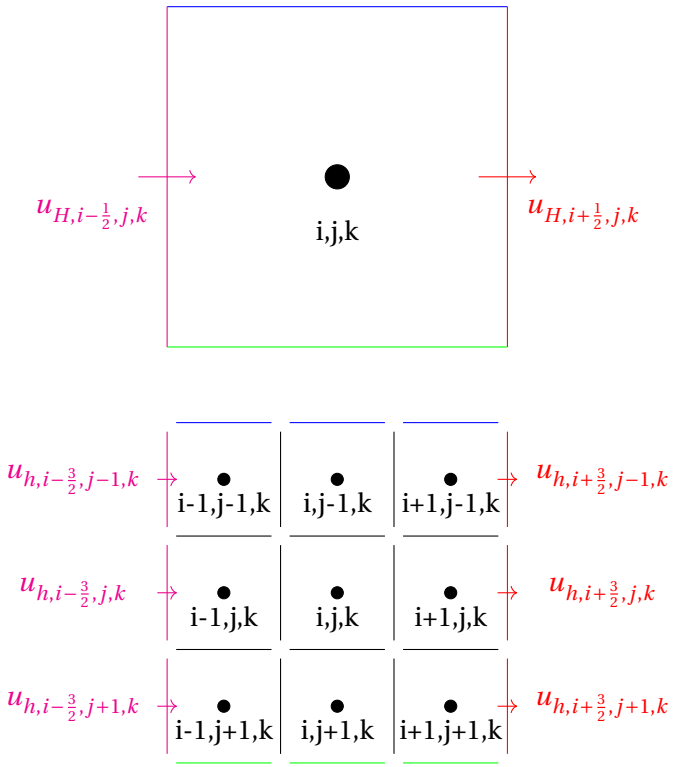


Figure 3.6: Visualization of the restriction operator used to map the edge/face-based fine solution onto the coarse grid. Top coarse mesh cell, bottom corresponding fine mesh cells in two-dimensions.

4

Mach 3 shock over a forward-facing step

As the first test case, the Mach 3 shock over a forward-facing step, initially presented in Woodward and Colella [2], is chosen. Compressible flows were one of the earliest problems solved with AMR, allowing to predict structures such as shock waves with previously unobtainable accuracy. Research has shown that even simple error sensors can be used with success for compressible flows [12]. This test problem, therefore, covers the vital branch of compressible flow as well as resembling a first gentle test case for all error sensors that should not pose too many difficulties, while providing many exciting features such as shock(reflections), slip lines, and expansion fans.

First, a description of the case setup is given. Since this test case is cheap to compute, subsequently, all investigated error sensors are computed on increasingly resolved uniform meshes, allowing for a detailed study of their performance in capturing different flow structures and their sensitivity on fine and coarse-grained meshes. Finally, actual results in the form of generated meshes and global error level are given for isotropic and anisotropic refinement.

4.1. Case setup

In this case, the compressible Euler equations have been solved for an ideal gas with $\gamma = 1.4$. The top and bottom boundaries are equipped with a slip boundary condition. Since this case is two-dimensional, a periodic boundary condition has been used for the front and rear boundary of the problem. A Riemann inflow boundary condition is used at the left boundary and an outflow condition at the right boundary. The flow entering the domain has a Mach number of 3. The domain's initial velocity is zero. The domain size and boundary conditions of the test case are shown in Figure 4.1. The domain is discretized with three blocks. This way, the slip boundary condition can be directly applied to the block boundaries with the consequence that no immersed boundary method is required. The starting mesh size for both the master and the slave simulation of all three blocks is given in Table 4.1. It was chosen in such a way that the resulting mesh has equal cell sizes across all blocks and that the cells have a width to height ratio of almost unity. Further, to avoid blocks with very few cells, the coarse mesh needed to have at least two cells per direction for the cases in which actual AMR was performed. The step corner received no special boundary condition, as it, for example, was done in Woodward and Colella [2]. Therefore an erroneous production in entropy is expected to take place at this point.

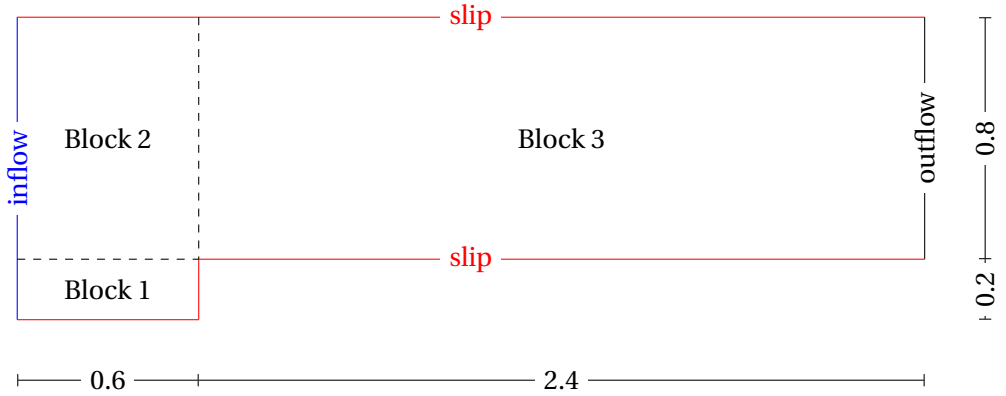


Figure 4.1: Boundary conditions and geometry for the Mach 3 shock over a forward-facing step. Periodic boundary conditions are applied in z-direction.

Table 4.1: Initial coarse and fine mesh for the Mach 3 shock over a forward-facing step.

(a) Master simulation

	nx	ny	nz
Block 1 master	16	4	1
Block 2 master	16	16	1
Block 3 master	64	32	1

(b) Slave simulation

	nx	ny	nz
Block 1 slave	8	2	1
Block 2 slave	8	8	1
Block 3 slave	32	16	1

4.2. Uniform mesh refinement with feature-based and novel error sensors

Since the test case is computationally cheap, first, the effectiveness of the error sensors was evaluated on increasingly fine uniform meshes. Simulations were performed on mesh levels 0 to 4, where 0 corresponds to half the cells of the coarse starting mesh described above. Even though this leads to blocks with as little as four cells, and would clearly not be used in any practical case, it was justified, so that the error sensor behavior can still be mapped on the smallest uniform mesh possible. As stated in Chapter 2, an error sensor might need a certain minimum mesh resolution to be of any use. Since the test case is compressible, a collocated mesh was chosen with an associated refinement ratio of 2. All simulations were run until 12 seconds of simulated time, and subsequently, the

error sensor distribution was extracted as a contour plot. An exponential color legend was chosen to avoid the plots saturating too quickly due to the wide range of error estimates within the same simulation. Table 4.2 shows the testing matrix of all computational experiments. The state variable that was used within all error sensors is velocity. For compressible problems of this kind, it might worth investigating the choice of, for example, density and temperature also.

Table 4.2: Test matrix of the investigated error sensors on increasingly fine uniform meshes.

Identifier	Error sensor	Figure	τ_{AMR} (s)
M3EF-001	Eq. 3.4 with $\|U\|$	4.3	0.25
M3EF-002	Eq. 3.9	4.4	0.25
M3EF-003	Eq. 3.10	4.5	0.25
M3EF-004	Eq. 3.11	4.6	0.25
M3EF-005	Eq. 3.5 with $\|U\|$	4.7	0.25
M3EF-006	Eq. 2.3 with $\|U\|$	4.9	0.25
M3EF-007	Eq. 3.7 with $\|U\|$	4.8	0.25

When scanning through the results of all simulations, which are depicted in Figure 4.3 to 4.8, results appear to be similar. At the initial refinement level, as expected, one can see that large parts of the entire domain contain significant error sensor values, which at the same time are also very washed out. With increasing mesh levels, the error areas become more spatially defined, and significant values can only be found in either the shock regions, at the Kelvin-Helmholtz instability (slip line), or the entropy layer. At the same time, qualitative differences between all error sensors start to vanish more and more until they become very comparably at mesh level 4. On closer inspection, some differences are revealed, which are discussed in the following paragraphs.

Bow shock: At refinement level 0, the bow shock is already so strong that it is marked as an area of large error across all simulations. The error is smeared out, but the shape is recognizable. Some error sensors are, however, clearer than others. The curvature criterion provides here the most refined indication (Figure 4.9). The Richardson error senors, shown in Figure 4.3 and 4.3) perform also reasonably well at this level. When going from level 1 to 4, the bow shock becomes more spatially defined, and subsequently, the error senors merely concentrate on this area as well. The ETE-based error senors slightly lag behind the others in terms of how crisp the error senor distribution appears, but these differences vanish quickly with increasing levels. What however remains is a slight difference in the regions, with the ETE having the thickest "error band" at level 4. One could explain this behavior from the origin of the ETE terms, which stems from the advection term (see e.g., Eq. 3.11). Thus the ETE terms, as their name says, show an error transport. Another behavior only to be found in the Richardson- and ETE-type sensors are two aliases of the bow shock located slightly downstream. The importance of these two additional error regions is not easy to judge and should be further investigated.

Shock reflections: The shock reflections originating from the top and bottom walls are captured similarly by the error sensors. Again the structures are washed-out at level 0 and become more and more refined with a finer mesh. Important for the effectiveness of the error sensors is not only that the regions of interest are captured but also that regions unimportant for the simulation are not captured. For example, inspecting Figure 4.4d one can see how there is virtually no error indicated between the shocks. This does not hold for Figure 4.5 and 4.6 which show traces of error emanating from the shocks. From the qualitative observations, one can also here conclude that all error sensors perform similarly.

Step corner and entropy layer: While every error sensor predicts an error around the step corner,

the highest errors are usually placed at the prominent regions of the bow shock and the shock reflection point on the lower wall. The sharp corner, introduces an error into the simulation, leading to the creation of a spurious entropy layer [2]. Due to the pressure gradient, this layer separates and leads to the separation bubble found at the shock reflection point on the lower wall. With this in mind, the behavior of the error sensors can be considered to be somewhat disappointing since the erroneous effects in the separation bubble are a direct consequence of the error introduction at the step corner. One could argue that an ideal error sensor should omit the erroneous separation point and create substantial errors solely at the corner so that the user gets notified that an even finer mesh is required to obtain a correct solution. The error sensors from Figure 4.3a, 4.7a and 4.9a do not only suggest the separation bubble for refinement but also large parts of the entire lower wall. The ETE-based error sensors perform in this regard qualitatively better since they only show high errors at the separation bubble itself.

Kelvin-Helmholtz instability: The Kelvin-Helmholtz instability is the last flow feature, which is qualitatively analyzed. In levels 0 and 1, this structure cannot be spotted in any error sensor. In some Figures such as in 4.4b a refinement is suggested at the top wall, which vanishes with increasing levels. Figure 4.4b can be regarded as an exception which already highlights the correct area from level 1 onward. When level 2 is reached, the Kelvin-Helmholtz instability becomes visible across all error sensors.

To conclude the qualitative analysis, looking at the results shown in Figure 4.3 to 4.8 one can say that in general, all error sensors show behavior that makes them suitable for this test case. Every error sensor highlights the areas with shock presence. Parts of little significance, e.g., in front of the initial shock, are given a low error and thus will not be suggested for adaption. Overall a surprising result considering the substantial differences in the deployed approaches. It is also pleasing to see how all error sensors work well within the four discussed regions. These have different kind of dominant physics and the fact that the error sensors perform well, highlights their diversity, especially when considering that they are as simple as the second derivative of a state variable.

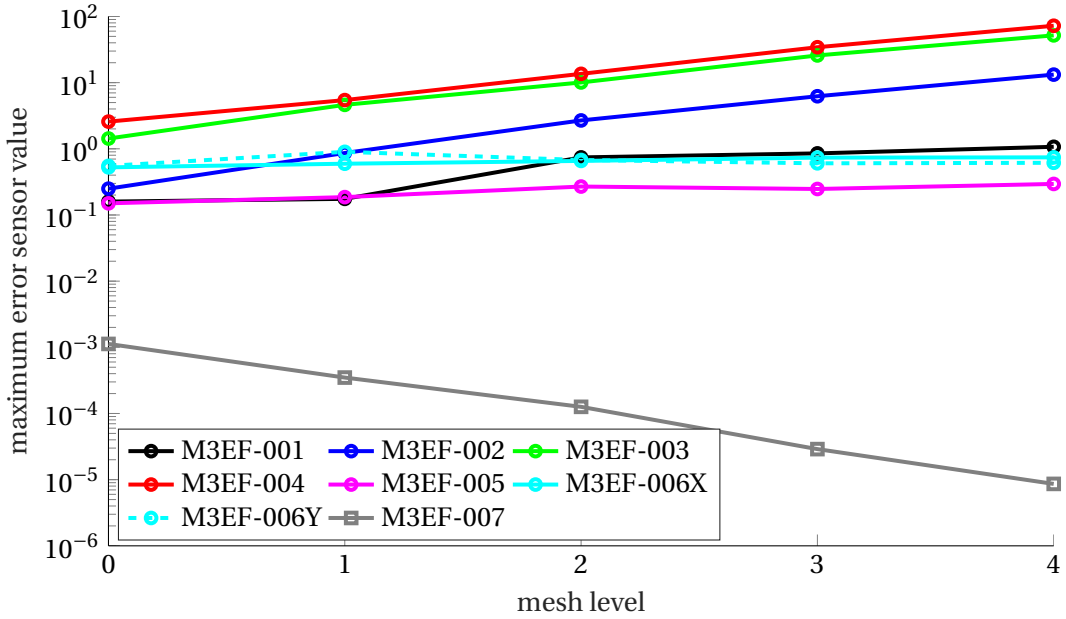
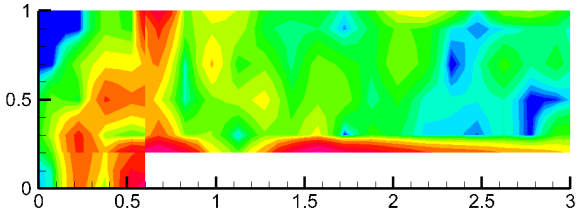


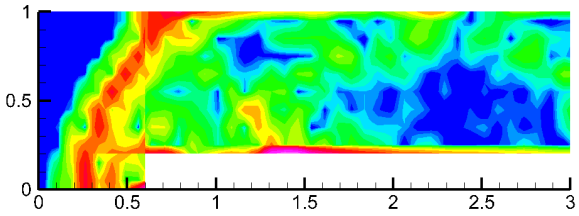
Figure 4.2: Maximum value of error sensors listed in Table 4.2. Simulations have been performed for mesh level 0 to 4.

Looking at the quantitative side of things, a careful inspection of the error sensor contour plots reveals that the areas containing high errors shrink in size with an increasing level, but the magnitude

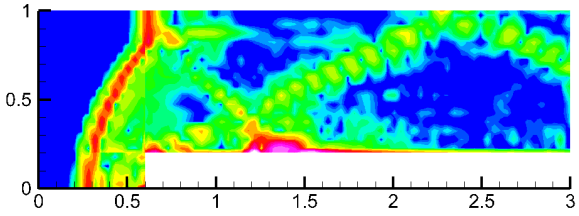
within the error regions does not reduce. Figure 4.2 shows the maximum error sensor value of the entire domain for all simulations at all levels. It becomes clear that as a matter of fact, the values not only stagnate but even increase. The curvature and error sensor from Equation 3.7 are an exception, which stems from the fact that they both contain the mesh size within their method. When considering the problem at hand, this behavior is actually not surprising. Since this type of problem is solved with the Euler equations, there are no finite shock thicknesses. With increasing mesh level, this discontinuity becomes more and more spatially defined, which leads to higher gradients and error gradients in the flow. For the ETE-based sensors this is especially true since they contain a multiplication of two factors which are becoming large at the shock locations. In the case of an Euler simulation, this is a undesired behavior since refinement would never stop. The pragmatic solution to this, which is also used throughout this thesis, is to define a maximum mesh level, thus avoiding over-refinement. Apart from, of course, developing a better error sensor, one other way to alleviate this problem would be to introduce an adaptive threshold that changes with an increasing mesh level. Yet another method could be to merely refine the worst cells until a target total cell count has been reached. However, for this strategy, mesh coarsening is crucial. As noted before, coarsening was not considered within this thesis. Otherwise, the transient setting would also lead to refinement of uninteresting areas e.g., when a shock has not completely developed yet. Another practical problem introduced by the increasing error is the definition of an appropriate error threshold. One cannot necessarily specify a target error threshold right from the beginning. It simply might be too high to get the initial adaption starting but might be entirely appropriate once it has started. On the flip side, an error threshold that might be appropriate at the beginning of the adaption cycle might simply be too low later on, leading to excessive over-refinement.



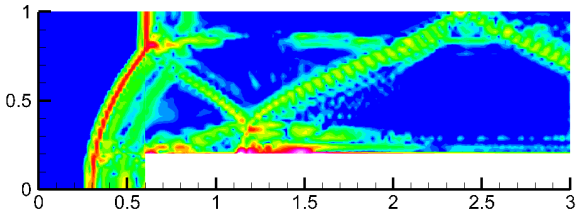
(a) Level 0 - M3EF-001



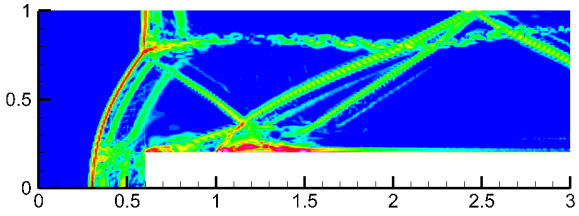
(b) Level 1 - M3EF-001



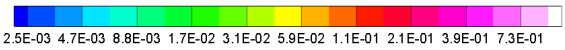
(c) Level 2 - M3EF-001



(d) Level 3 - M3EF-001

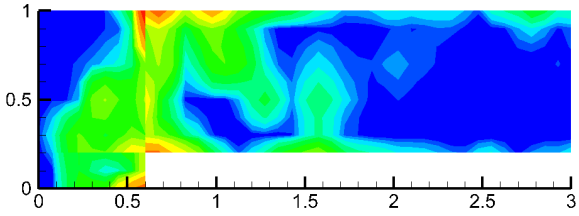


(e) Level 4 - M3EF-001

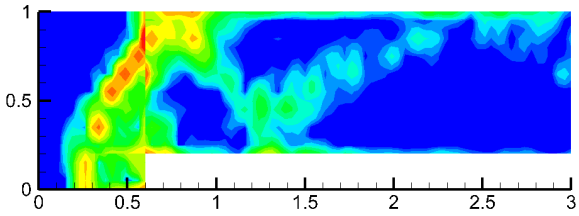


(f) Legend - M3EF-001

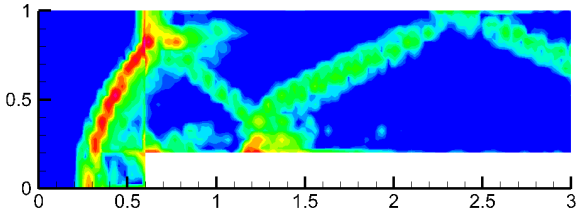
Figure 4.3: Contour plots of error sensor Equation 3.4 with velocity magnitude.



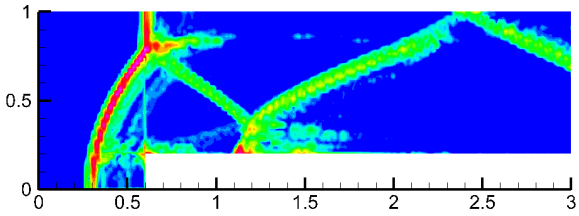
(a) Level 0 - M3EF-002



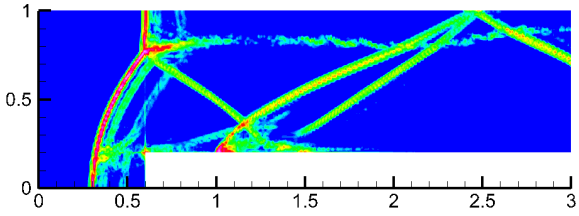
(b) Level 1 - M3EF-002



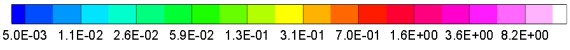
(c) Level 2 - M3EF-002



(d) Level 3 - M3EF-002

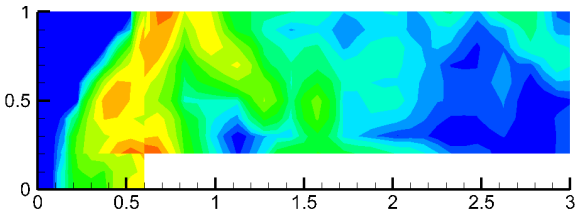


(e) Level 4 - M3EF-002

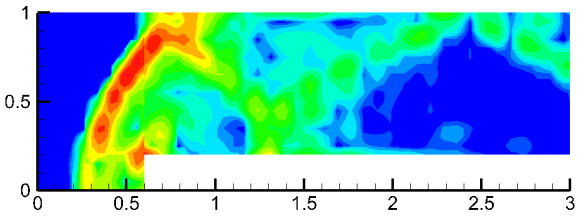


(f) Legend - M3EF-002

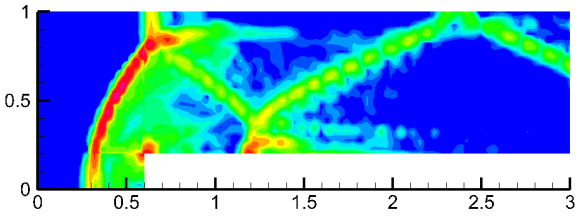
Figure 4.4: Contour plots of error sensor Equation 3.9.



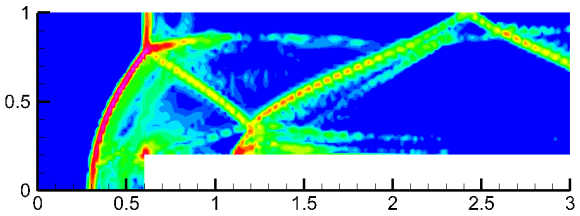
(a) Level 0 - M3EF-003



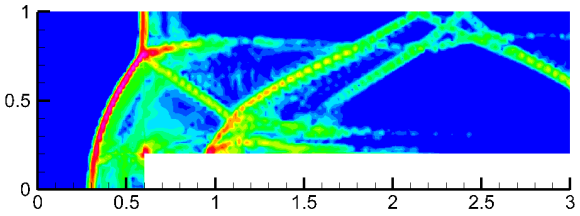
(b) Level 1 - M3EF-003



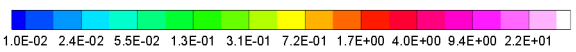
(c) Level 2 - M3EF-003



(d) Level 3 - M3EF-003

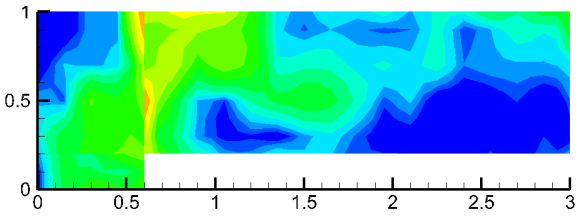


(e) Level 4 - M3EF-003

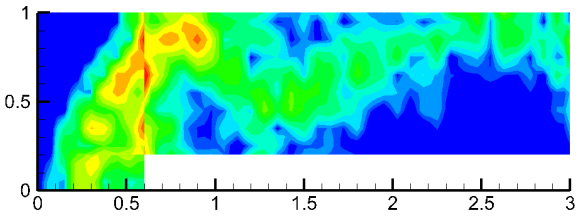


(f) Legend - M3EF-003

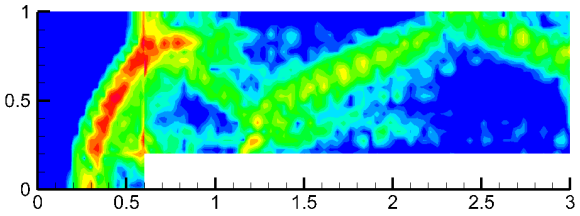
Figure 4.5: Contour plots of error sensor Equation 3.10.



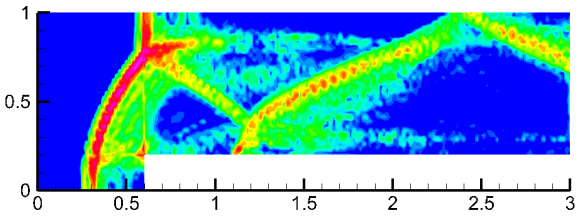
(a) Level 0 - M3EF-004



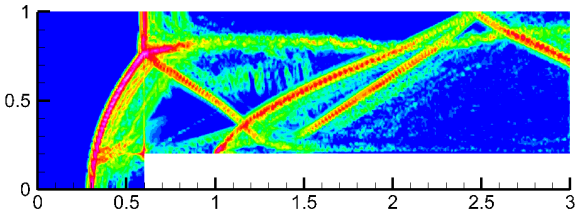
(b) Level 1 - M3EF-004



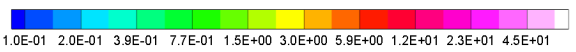
(c) Level 2 - M3EF-004



(d) Level 3 - M3EF-004

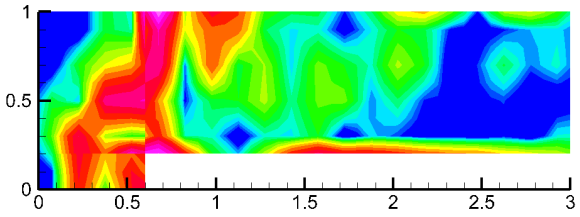


(e) Level 4 - M3EF-004

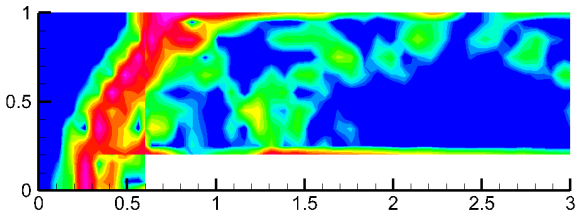


(f) Legend - M3EF-004

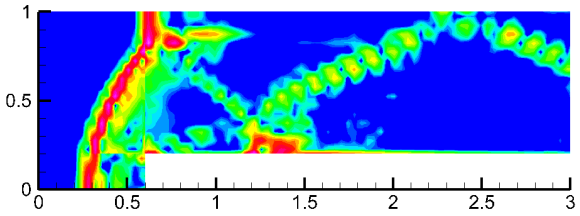
Figure 4.6: Contour plots of error sensor Equation 3.11.



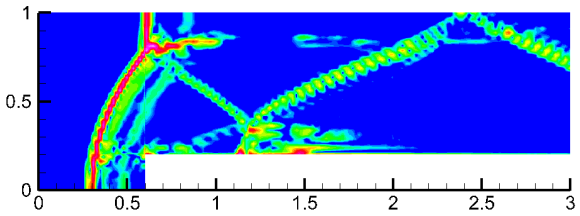
(a) Level 0 - M3EF-005



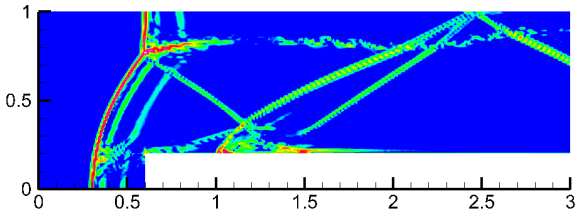
(b) Level 1 - M3EF-005



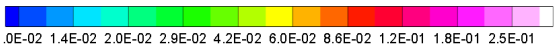
(c) Level 2 - M3EF-005



(d) Level 3 - M3EF-005

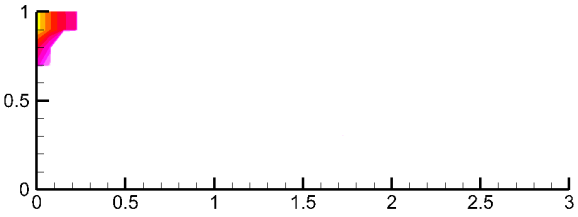


(e) Level 4 - M3EF-005

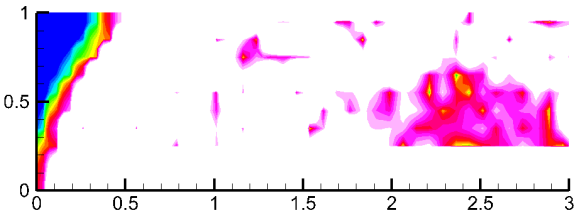


(f) Legend - M3EF-005

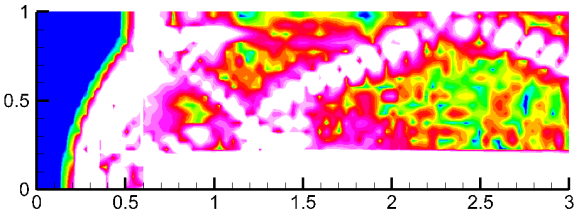
Figure 4.7: Contour plots of error sensor Equation 3.5 with velocity magnitude.



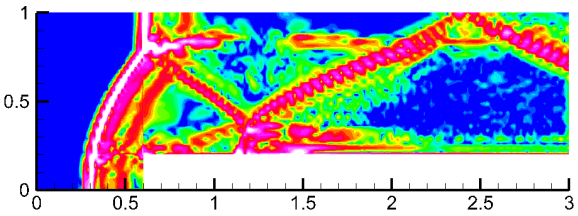
(a) Level 0 - M3EF-007



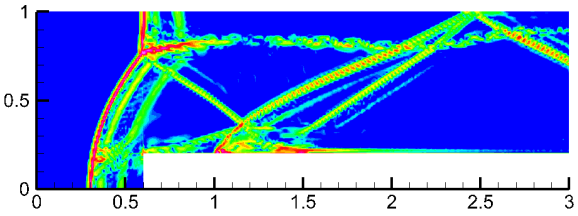
(b) Level 1 - M3EF-007



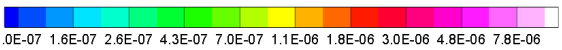
(c) Level 2 - M3EF-007



(d) Level 3 - M3EF-007

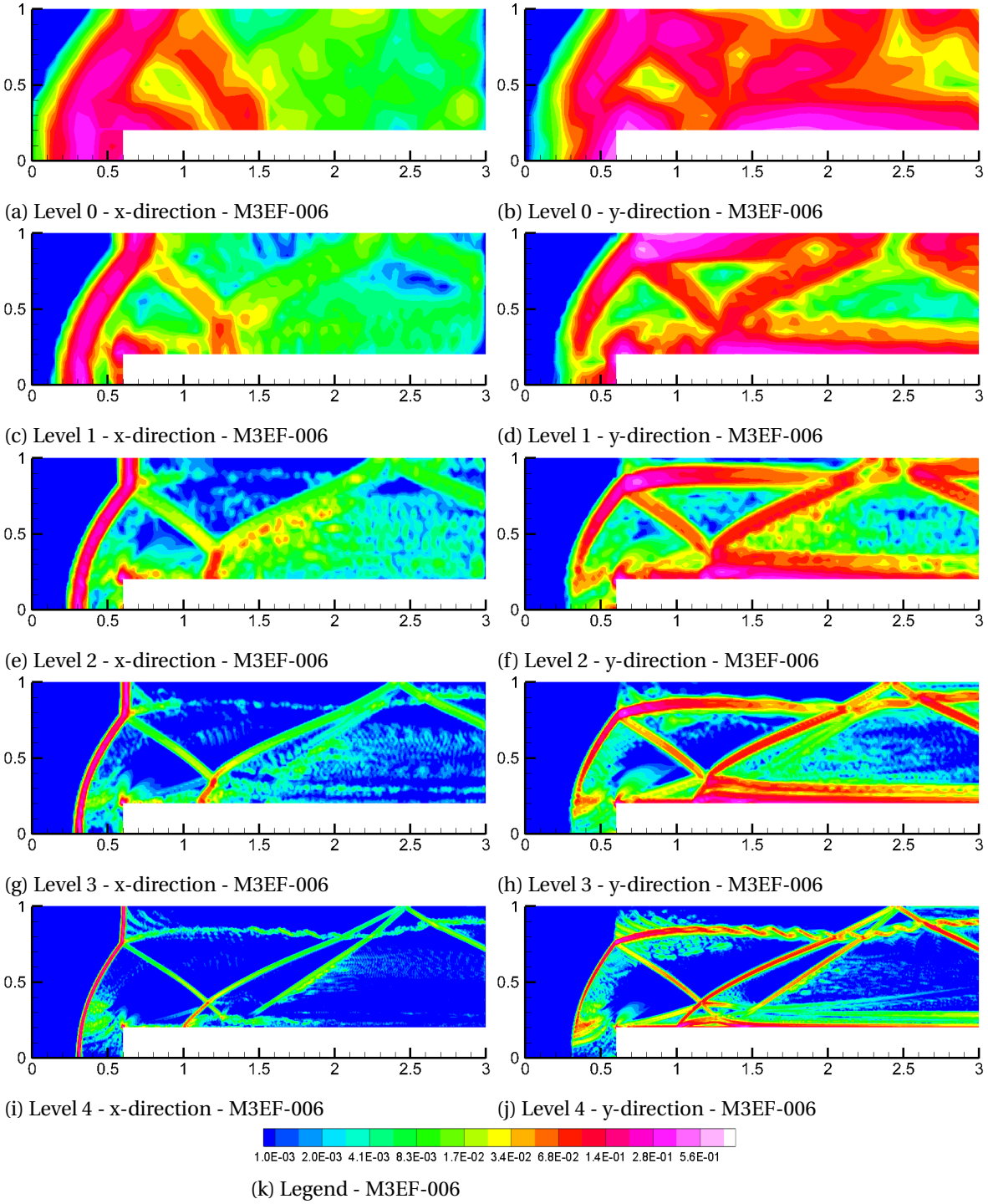


(e) Level 4 - M3EF-007



(f) Legend - M3EF-007

Figure 4.8: Contour plots of error sensor Equation 3.5 with velocity magnitude.



(a) Level 0 - x-direction - M3EF-006

(b) Level 0 - y-direction - M3EF-006

(c) Level 1 - x-direction - M3EF-006

(d) Level 1 - y-direction - M3EF-006

(e) Level 2 - x-direction - M3EF-006

(f) Level 2 - y-direction - M3EF-006

(g) Level 3 - x-direction - M3EF-006

(h) Level 3 - y-direction - M3EF-006

(i) Level 4 - x-direction - M3EF-006

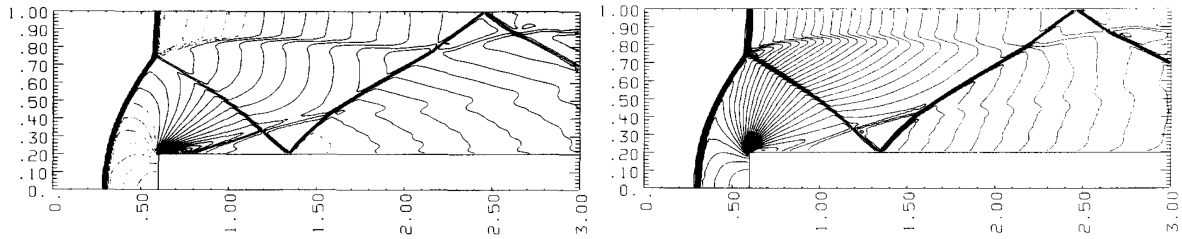
(j) Level 4 - y-direction - M3EF-006

(k) Legend - M3EF-006

Figure 4.9: Contour plots of error sensor Equation 2.3 with velocity magnitude.

4.3. Isotropic refinement

In this part, the performance of the error sensors is investigated during actual adaption. Every error sensor mentioned above will be tested with three different error thresholds. The initial threshold is the maximum error at level 0 from Figure 4.2. The threshold is thus chosen in such a way that the adaption can barely start. In every subsequent simulation, the error threshold is reduced by a factor of two (for M3AMR-007, a factor of 10 is chosen). For every test, a contour plot of the mesh levels and the error sensor is shown. In both, a Mach number line plot is given in the background as a reference to show the flow solution. The thresholds are the same as in Woodward and Colella [2]. For convenience, the reference solution of density and Mach number from literature is given in Figure 4.10. The testing matrix of all computational experiments is given in Table 4.3. It contains all relevant configuration parameters of the AMR routine. Since in this thesis, no coarsening was considered, the simulation was run until 6 seconds without adaption on a mesh level of 0. At that point, a new simulation was started with the full adaption routine activated. The reasoning behind this choice is already to preposition the formed shocks in their approximate final location. Without this initialization, the adaption routine would essentially refine all areas behind the initial bow shock, since the shocks heavily change position at the beginning of the simulated time.



(a) Mach number, 30 contours from −0.9184 to 2.856 (b) Density, 30 contours from 0.2568 to 6.067

Figure 4.10: Baseline results from Woodward and Colella [2] at $t = 4s$ (corresponds to $t = 12s$ for the non-dimensionlization used within INCA).

Table 4.3: Testing matrix for Mach 3 shock over a forward-facing step using an adapted mesh.

Identifier	max lvl	ϵ_{AMR}	FlagF	τ_{AMR}	Dim	Error sensor
M3AMR-001	4	0.1594	Alg. 2	0.25	none	Eq. 3.4 with $\|U\|$
M3AMR-002	4	0.8644	Alg. 2	0.25	none	Eq. 3.9
M3AMR-003	4	4.6150	Alg. 2	0.25	none	Eq. 3.10
M3AMR-004	4	5.4465	Alg. 2	0.25	none	Eq. 3.11
M3AMR-005	4	0.1862	Alg. 2	0.25	none	Eq. 3.5 with $\|U\|$
M3AMR-006	4	0.9000	Alg. 2	0.25	none	Eq. 2.3 with $\|U\|$
M3AMR-007	4	3.49E-4	Alg. 2	0.25	none	Eq. 3.7 with $\|U\|$

When looking at the effect of the refinement threshold for experiment 'M3AMR-007' (Figure 4.18) in comparison to e.g., 'M3AMR-002' (Figure 4.13) a clear difference in the created meshes across the levels is visible. For the former, one can rather precisely control the overall adaption of the mesh, i.e., there is an obvious difference in mesh size between the highest and the lowest threshold. The control of the latter is much more restricted since already the first refinement thresholds will lead to an adaption in large parts of the domain. In terms of control, one can say that the simulations with the mesh size included provide the greatest control (M3AMR-6 and -7) followed by the "pure" Richardson simulations (M3AMR-5 and -1) whereas the ETE-based simulations perform the worst in this regard (M3AMR-2, -3, -4). An area where all approaches work almost identical, is the treatment of the bow shock. A castellated mesh is created following the shape of the shock nicely. No

pointless, and thus spurious fine blocks can be found in front of the shock. The mesh directly after the bow shock is also very comparable across all simulations. Also here a castellated structure is formed which adheres nicely to the shape of the bow shock, the simulations in Figure 4.15e and 4.18e seem a bit more generous with the cells in this area, but this can probably be attributed to the error threshold being just a bit too low for this simulation thus leading to excessive refinement. Traversing further through the regions to the slip line and its Kelvin-Helmholtz instability, one again can speak about a very comparable situation. The entire slip line gets refined to the highest mesh level. The area encircled by the slip line and the shock reflections also sees refinement. While this area actually contains almost no error, the adaption that happens here can be attributed to the fact that the balance criterion must be enforced and that the blocks are still quite large in size, thus reaching from the shocks into the encircled area. This problem could, therefore, potentially profit from splitting block three into multiple initial blocks. In the qualitative analysis, simulations 'M3EF-001, -004, -006, -007' were found to indicate an error across the entire lower wall. Looking now at the actual adaption that took place, then indeed, the AMR routine in simulations 'M3AMR-001, -004, -007' lead to such a mesh. This was, however, not the case for the curvature criterion. A last observation can be made when considering how far the flow has developed. The simulations in Figure 4.12e, 4.16e and 4.18c have reached a solution comparable to the uniform simulations such as shown in Figure 4.3e. Examining the other simulations, then it can be seen that there is a difference in the height of the transition from the bow shock to the normal shock, and also in the position of the separation bubble. These different flow structures indeed belong to a lower simulated time. These adaption routines, therefore, create a temporal error that makes the simulations lag behind!

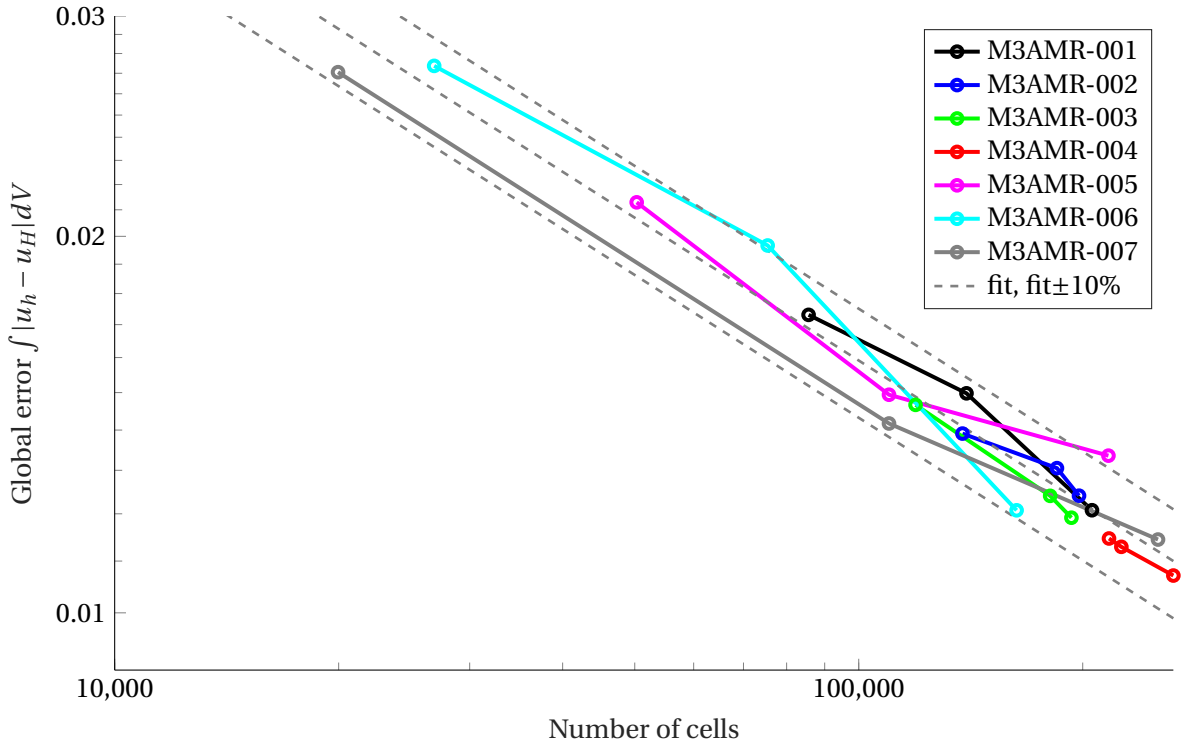
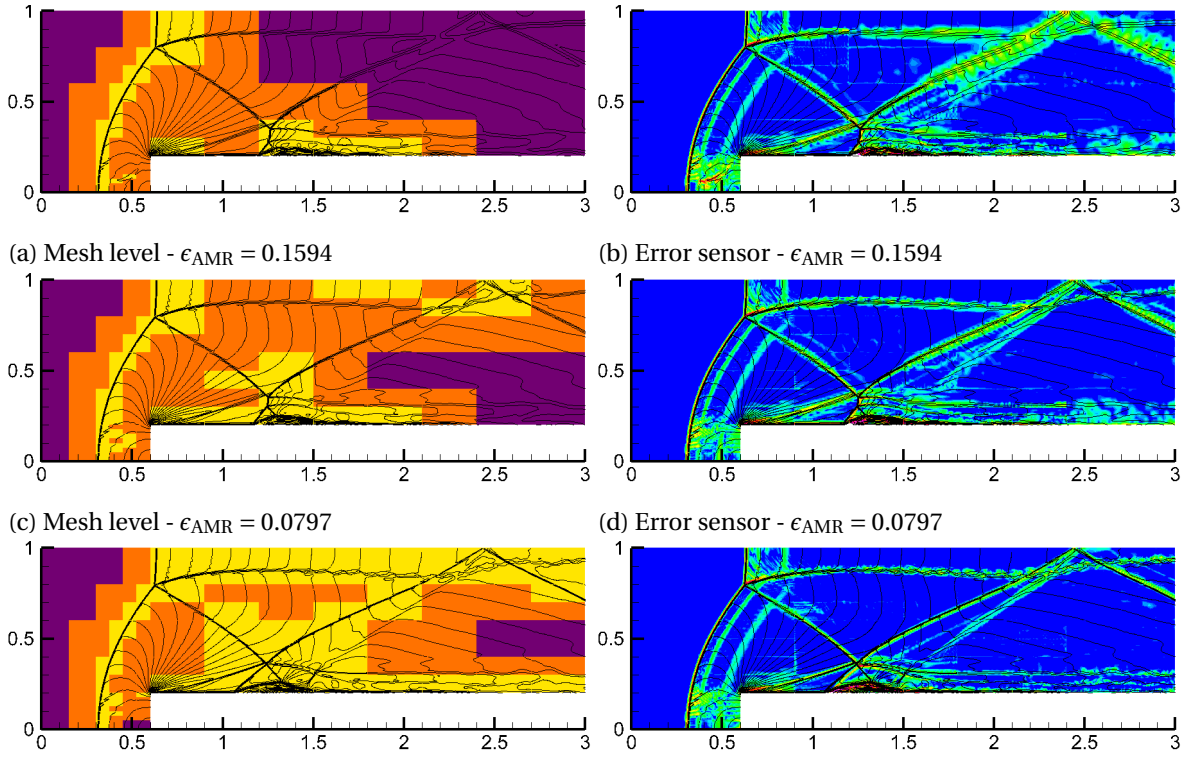


Figure 4.11: Global error vs cell count for all methods.

To obtain a quantitative measure of the effectiveness of the error sensors, a global error has been computed for all simulations as well. Equation 3.7 was used as the error measure. The results can be found in Figure 4.11. The graphs underline the results obtained so far. All error sensors did give sensible refinement suggestions in the qualitative analysis. This similarity can also be found in the behavior of the graphs. They can all considered to have the same slope, i.e., an increase in cell count

leads to the same global error reduction for all methods. The Richardson-type and curvature-based error sensors show a large span in error for the examined error thresholds. The ETE-based error sensors all operate within a narrow band, highlighting that it is difficult with them to fine-tune the exact mesh resolution of the problem since their error does increase with continuing refinement. For better visualization, a fitted curve resembling the average error versus cell count was plotted together with a curve resembling a 5% lower and higher error. It can be seen that all simulations lie within this band. Especially 'M3AMR-007' looks like a strong candidate, producing the lowest error for the lowest cell count across a wide range. Such an error sensor, therefore, allows for two approaches, either one wants to have a certain cell count and obtain the best error, or one wants to have a certain error and obtain the best cell count. In both examples 'M3AMR-007' performs the strongest across the entire range.



(a) Mesh level - $\epsilon_{\text{AMR}} = 0.1594$

(b) Error sensor - $\epsilon_{\text{AMR}} = 0.1594$

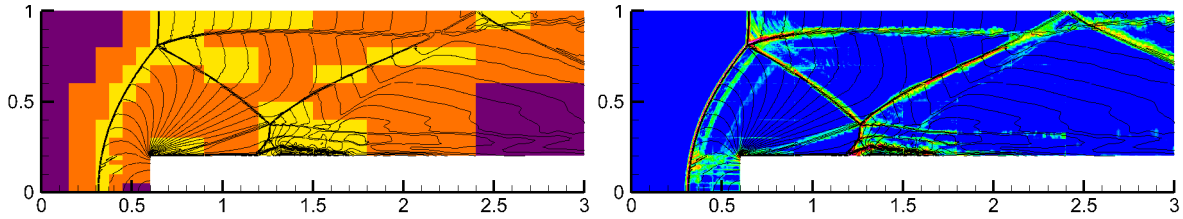
(c) Mesh level - $\epsilon_{\text{AMR}} = 0.0797$

(d) Error sensor - $\epsilon_{\text{AMR}} = 0.0797$

(e) Mesh level - $\epsilon_{\text{AMR}} = 0.0399$

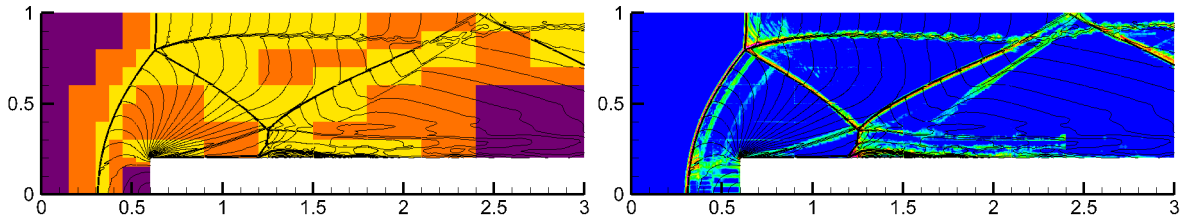
(f) Error sensor - $\epsilon_{\text{AMR}} = 0.0399$

Figure 4.12: Mesh levels (left) and error sensor values (right) for simulation M3AMR-001 with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4. Error sensor legend according to Figure 4.3.



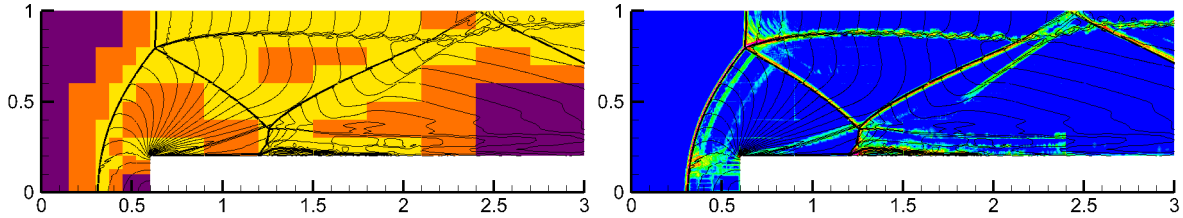
(a) Mesh level - $\epsilon_{\text{AMR}} = 0.8644$

(b) Error sensor - $\epsilon_{\text{AMR}} = 0.8644$



(c) Mesh level - $\epsilon_{\text{AMR}} = 0.4322$

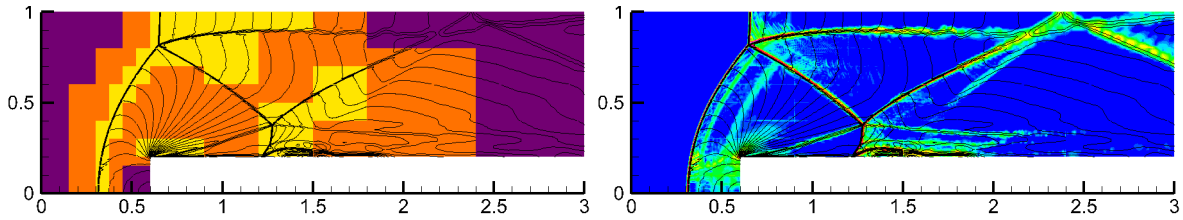
(d) Error sensor - $\epsilon_{\text{AMR}} = 0.4322$



(e) Mesh level - $\epsilon_{\text{AMR}} = 0.2161$

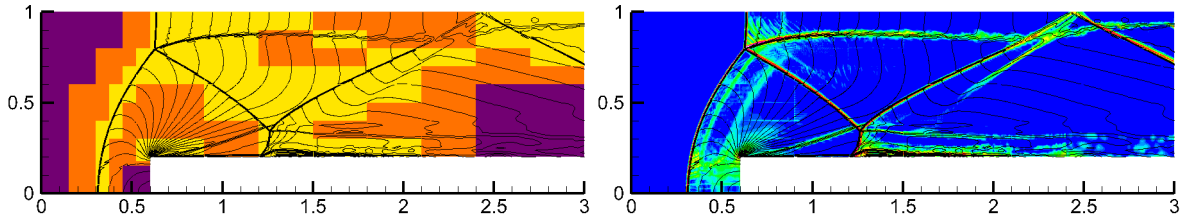
(f) Error sensor - $\epsilon_{\text{AMR}} = 0.2161$

Figure 4.13: Mesh levels (left) and error sensor values (right) for simulation M3AMR-002 with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4. Error sensor legend according to Figure 4.4.



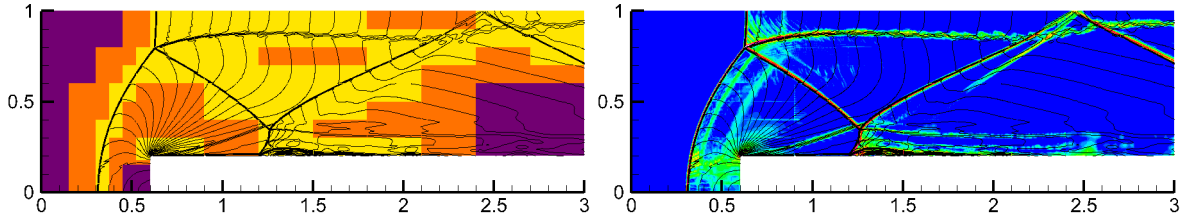
(a) Mesh level - $\epsilon_{\text{AMR}} = 4.6150$

(b) Error sensor - $\epsilon_{\text{AMR}} = 4.6150$



(c) Mesh level - $\epsilon_{\text{AMR}} = 2.3075$

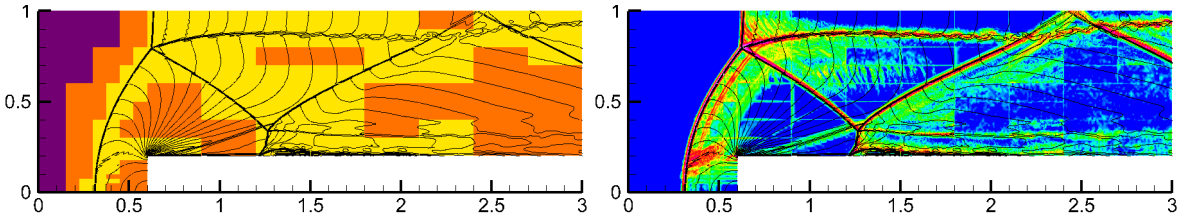
(d) Error sensor - $\epsilon_{\text{AMR}} = 2.3075$



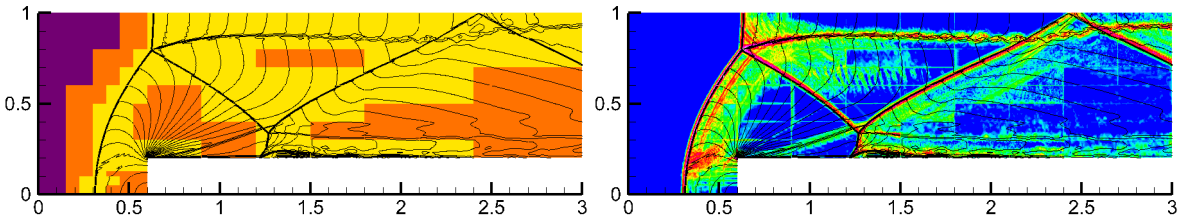
(e) Mesh level - $\epsilon_{\text{AMR}} = 1.1538$

(f) Error sensor - $\epsilon_{\text{AMR}} = 1.1538$

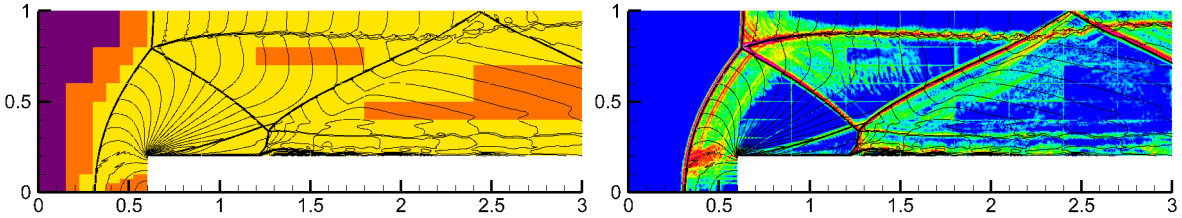
Figure 4.14: Mesh levels (left) and error sensor values (right) for simulation M3AMR-003 with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4. Error sensor legend according to Figure 4.5.



(a) Mesh level - $\epsilon_{\text{AMR}} = 5.4465$ (b) Error sensor - $\epsilon_{\text{AMR}} = 5.4465$

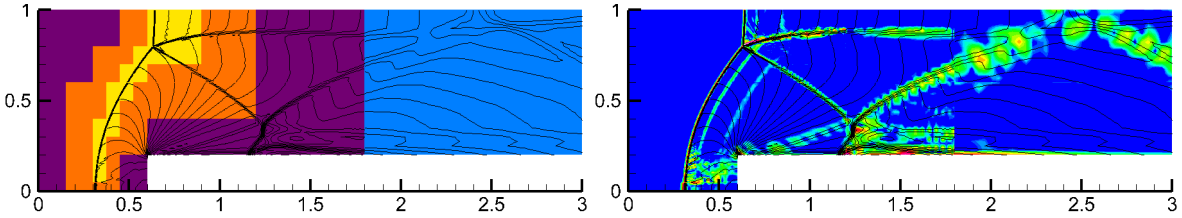


(c) Mesh level - $\epsilon_{\text{AMR}} = 2.7233$ (d) Error sensor - $\epsilon_{\text{AMR}} = 2.7233$

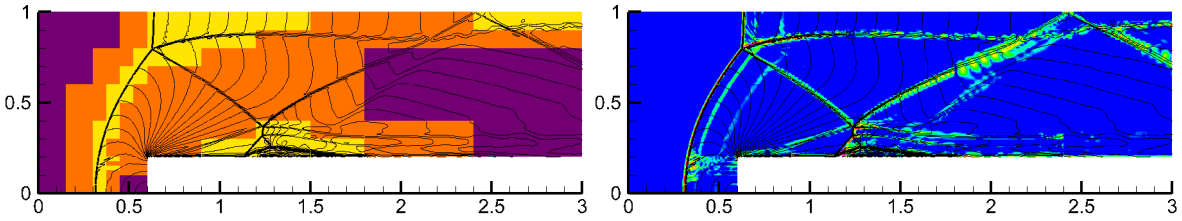


(e) Mesh level - $\epsilon_{\text{AMR}} = 1.3616$ (f) Error sensor - $\epsilon_{\text{AMR}} = 1.3616$

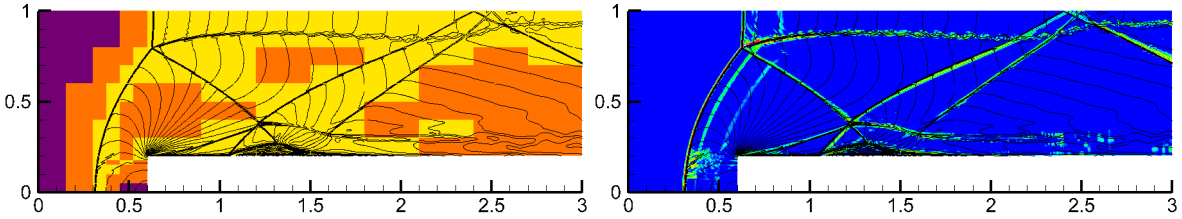
Figure 4.15: Mesh levels (left) and error sensor values (right) for simulation M3AMR-004 with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4. Error sensor legend according to Figure 4.6.



(a) Mesh level - $\epsilon_{\text{AMR}} = 0.1862$ (b) Error sensor - $\epsilon_{\text{AMR}} = 0.1862$

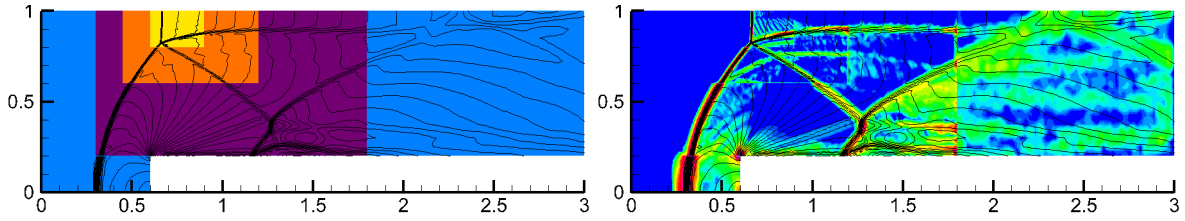


(c) Mesh level - $\epsilon_{\text{AMR}} = 0.0931$ (d) Error sensor - $\epsilon_{\text{AMR}} = 0.0931$



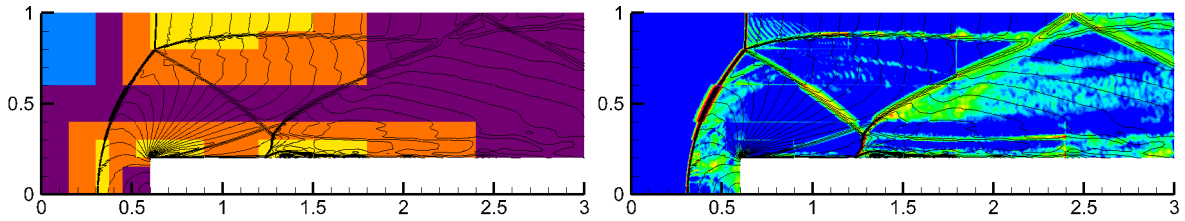
(e) Mesh level - $\epsilon_{\text{AMR}} = 0.0466$ (f) Error sensor - $\epsilon_{\text{AMR}} = 0.0466$

Figure 4.16: Mesh levels (left) and error sensor values (right) for simulation M3AMR-005 with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4. Error sensor legend according to Figure 4.7.



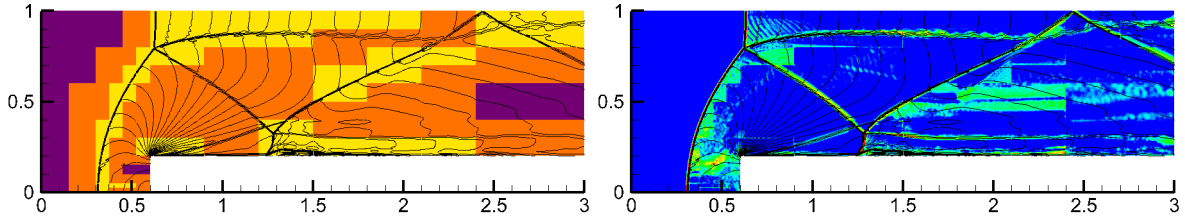
(a) Mesh level - $\epsilon_{\text{AMR}} = 0.9000$

(b) Error sensor - $\epsilon_{\text{AMR}} = 0.9000$



(c) Mesh level - $\epsilon_{\text{AMR}} = 0.4500$

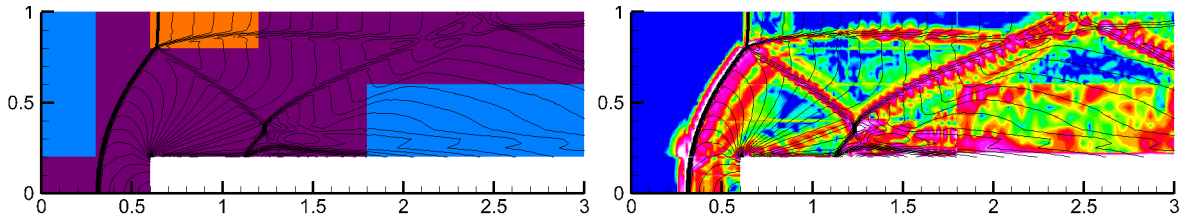
(d) Error sensor - $\epsilon_{\text{AMR}} = 0.4500$



(e) Mesh level - $\epsilon_{\text{AMR}} = 0.2250$

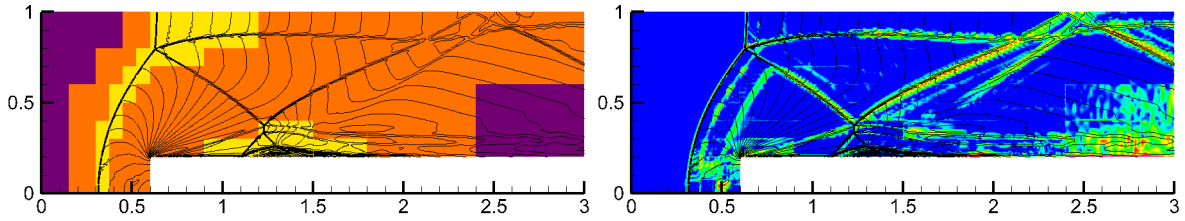
(f) Error sensor - $\epsilon_{\text{AMR}} = 0.2250$

Figure 4.17: Mesh levels (left) and error sensor values (right) for simulation M3AMR-006 with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4. Error sensor legend according to Figure 4.9.



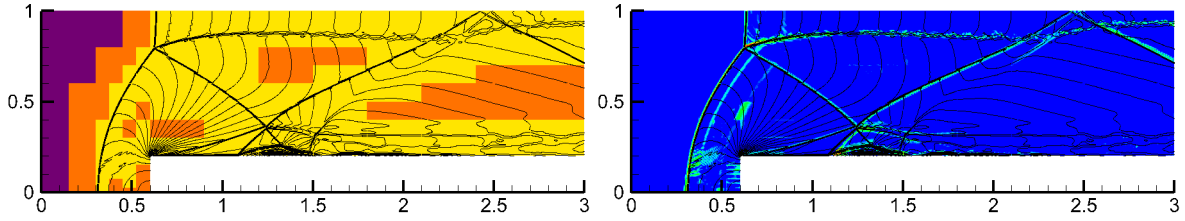
(a) Mesh level - $\epsilon_{\text{AMR}} = 3.4912E-4$

(b) Error sensor - $\epsilon_{\text{AMR}} = 3.4912E-4$



(c) Mesh level - $\epsilon_{\text{AMR}} = 3.4912E-5$

(d) Error sensor - $\epsilon_{\text{AMR}} = 3.4912E-5$



(e) Mesh level - $\epsilon_{\text{AMR}} = 3.4912E-6$

(f) Error sensor - $\epsilon_{\text{AMR}} = 3.4912E-6$

Figure 4.18: Mesh levels (left) and error sensor values (right) for simulation M3AMR-007 with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4. Error sensor legend according to Figure 4.8.

4.4. Anisotropic refinement

For the anisotropic refinement study, all simulations have been recomputed, but this time with a directionality criterion (Algorithm 4) for the error sensor. For the Richardson and ETE-based sensors, the undivided curvature was used to assess the refinement direction. An overview of all simulations is given in Table 4.4. All other settings have been kept the same as in the simulations of Table 4.3.

Table 4.4: Testing matrix for Mach 3 shock over a forward-facing step using anisotropic refinement.

Identifier	max lvl	ϵ_{AMR}	FlagF	τ_{AMR}	Error sensor
M3AMR-001-ANISO	4	0.1594	Alg. 2	0.25	Eq. 3.4 with $\|U\|$
M3AMR-002-ANISO	4	0.8644	Alg. 2	0.25	Eq. 3.9
M3AMR-003-ANISO	4	4.6150	Alg. 2	0.25	Eq. 3.10
M3AMR-004-ANISO	4	5.4465	Alg. 2	0.25	Eq. 3.11
M3AMR-005-ANISO	4	0.1862	Alg. 2	0.25	Eq. 3.5 with $\|U\|$
M3AMR-006-ANISO	4	0.9000	Alg. 2	0.25	Eq. 2.3 with $\|U\|$
M3AMR-007-ANISO	4	3.49E-4	Alg. 2	0.25	Eq. 3.7 with $\|U\|$

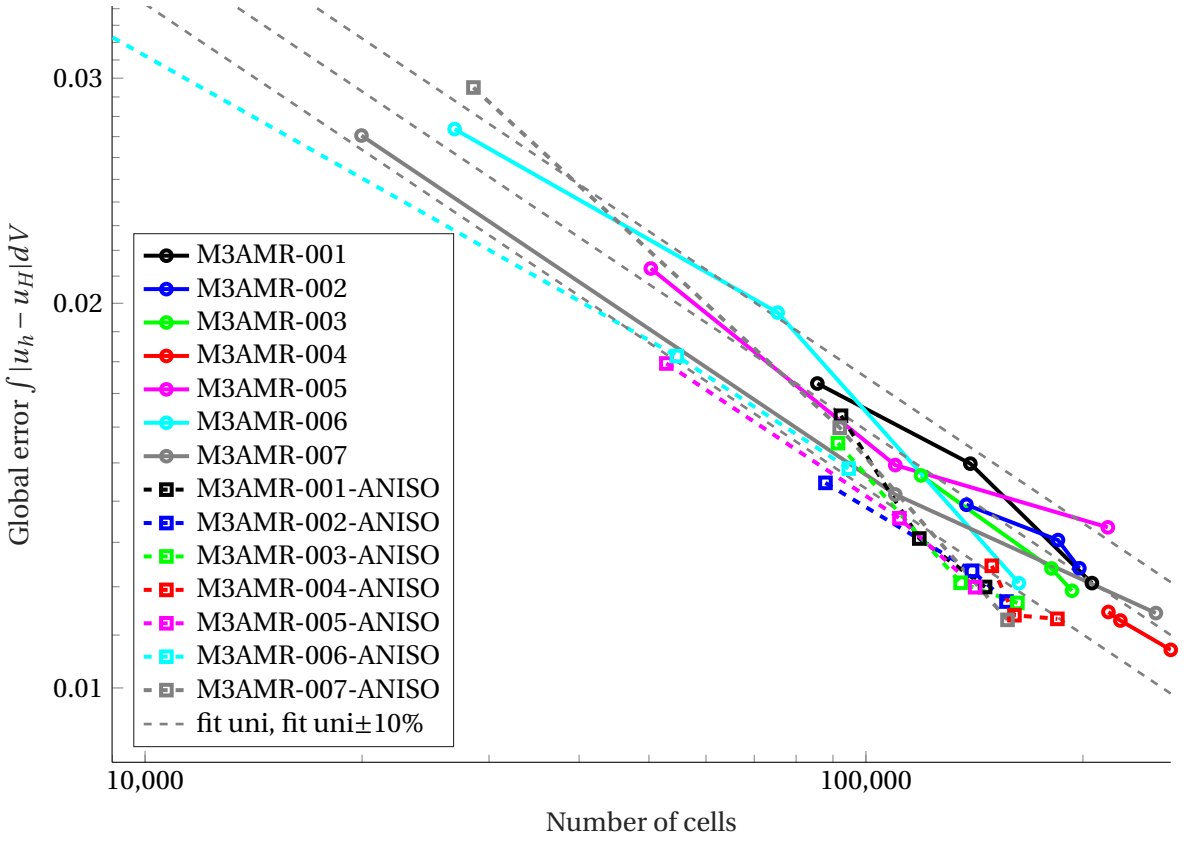
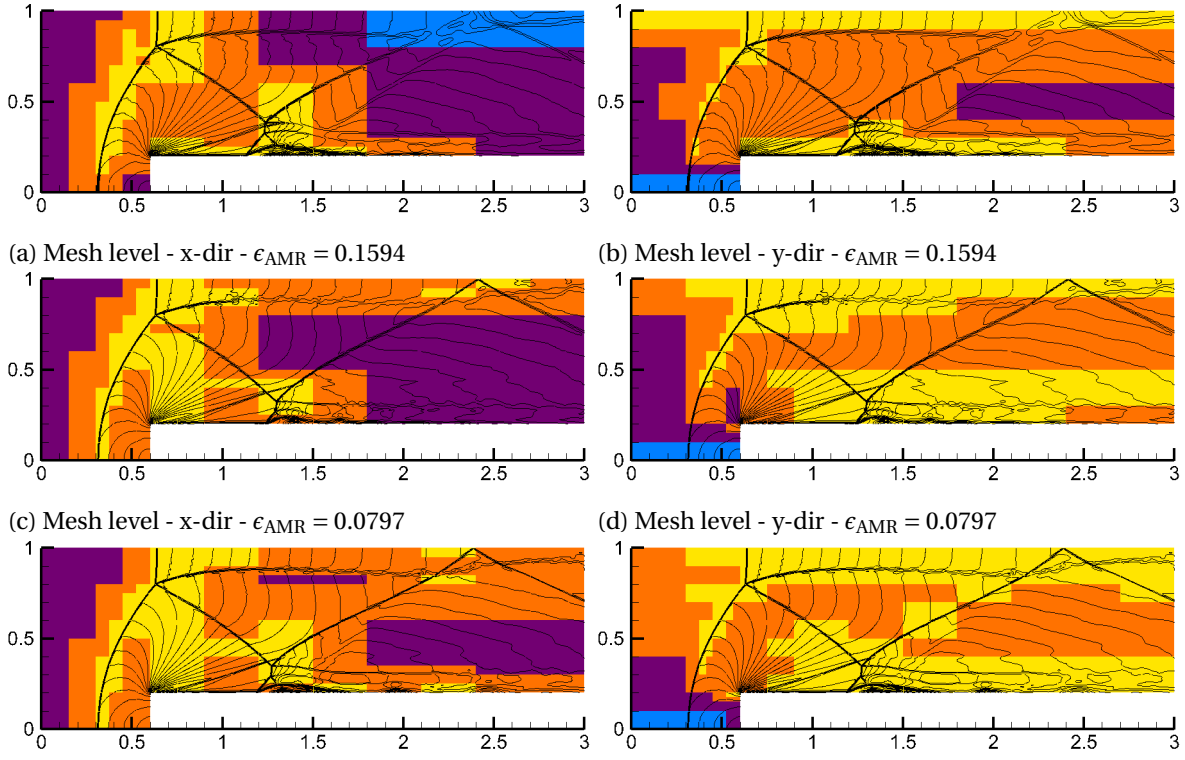


Figure 4.19: Global error vs cell count for isotropic and anisotropic refinement.

While shocks are usually a good candidate for anisotropic refinement, this case can be regarded as difficult for the block-structured mesh used in this thesis. Most of the shocks within this problem are oblique shocks, thus are not aligned with either of the two principal directions. Since H-refinement is used, adaption must happen in both directions to properly resolve an oblique shock. Figure 4.19 shows the global errors of the anisotropic simulations in comparison to the uniform ones. The regressed error lines from the uniform AMR simulations of Figure 4.11 show that anisotropic refinement leads to an error-cell count efficiency that is roughly 10 to 15% higher. This increase is

constant across all simulations. Merely the first error threshold of 'M3AMR-007-ANISO' produces an error that is above the respective one from the uniform simulation. It is important to talk here specifically about efficiency since, for some simulations, such as 'M3AMR-004-ANISO', anisotropic refinement leads to a higher error but at a significantly lower cell count. For some simulations, however, such as 'M3AMR-001-ANISO', the efficiency increase is purely created by an error reduction while keeping the overall cell count constant. Finally, the error-cell range has approximately stayed constant, meaning that the same control over the extent of the adaption is giving.

Figure 4.20 and 4.21 show the mesh refinement level in x and y-direction respectively for simulations 'M3AMR-001-ANSIO' and 'M3AMR-005-ANSIO'. In both simulations, as one can expect, the curved parts of the bow shock see uniform refinement in both directions. The straight part of the bow shock just before the step does get refined up to the maximum level in x-direction but only to level one in the y-direction. Whereas in the isotropic simulations, no spurious refinement was found in front of the bow shock, now in anisotropic refinement, there are some apparently spurious refinements in the y-direction, as shown in Figure 4.20b and 4.21b. They are, however, created by the 'Balance criterion' to satisfy mesh consistency. A situation is found at the bow shock that matches the one from Figure 3.1. Due to the 'alternating block splitting and refinement criterion', first, a refinement action happens, followed by a splitting action to resolve the conflict. The refinement of the slip line is anisotropically, with the mesh level being higher in y-direction than in x-direction. The same behavior can be seen on the lower wall, which, in both simulations, sees mainly refinement in the y-direction. In general, the refinement of the y-direction resembles the refinement of the uniform case. Therefore the main savings in cell count stem from savings in the x-direction. This is also in line with the results from Figure 4.9i and 4.9j where it is visible that the undivided curvature is much higher in y-direction than in x-direction.



(a) Mesh level - x-dir - $\epsilon_{\text{AMR}} = 0.1594$

(b) Mesh level - y-dir - $\epsilon_{\text{AMR}} = 0.1594$

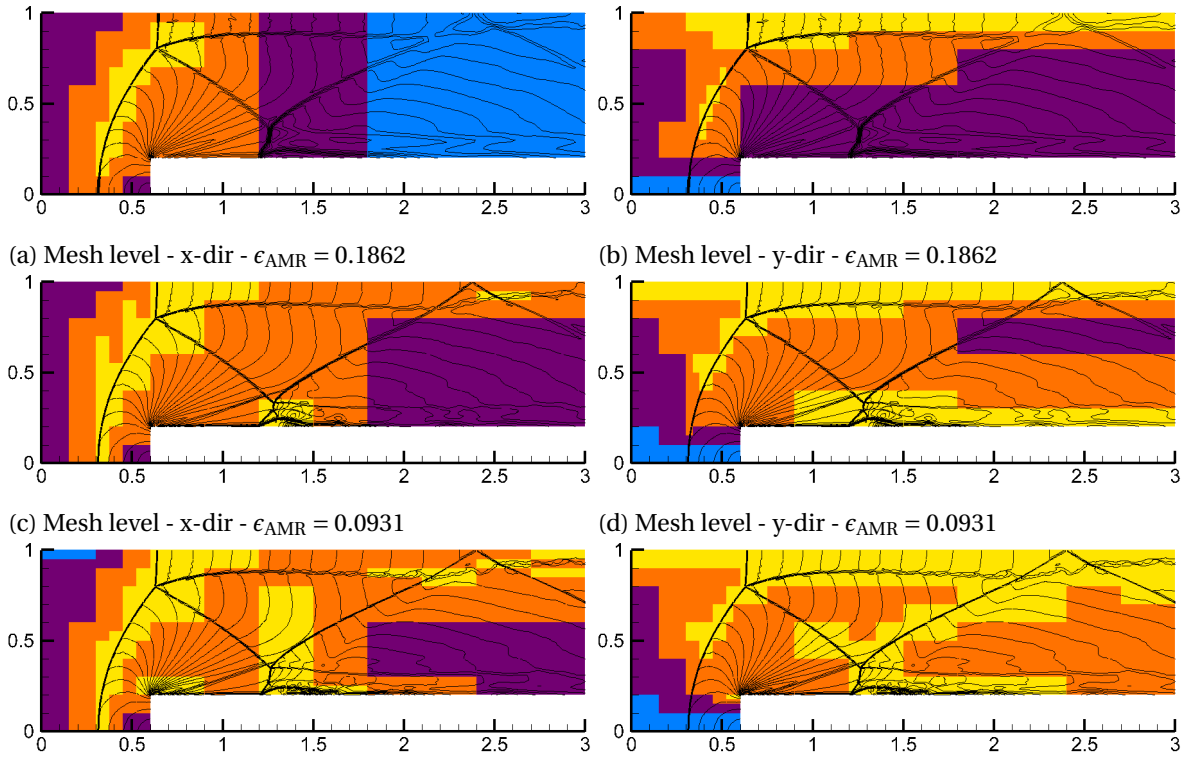
(c) Mesh level - x-dir - $\epsilon_{\text{AMR}} = 0.0797$

(d) Mesh level - y-dir - $\epsilon_{\text{AMR}} = 0.0797$

(e) Mesh level - x-dir - $\epsilon_{\text{AMR}} = 0.0399$

(f) Mesh level - y-dir - $\epsilon_{\text{AMR}} = 0.0399$

Figure 4.20: Mesh levels for simulation M3AMR-001-ANISO with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4.



(a) Mesh level - x-dir - $\epsilon_{\text{AMR}} = 0.1862$

(b) Mesh level - y-dir - $\epsilon_{\text{AMR}} = 0.1862$

(c) Mesh level - x-dir - $\epsilon_{\text{AMR}} = 0.0931$

(d) Mesh level - y-dir - $\epsilon_{\text{AMR}} = 0.0931$

(e) Mesh level - x-dir - $\epsilon_{\text{AMR}} = 0.0466$

(f) Mesh level - y-dir - $\epsilon_{\text{AMR}} = 0.0466$

Figure 4.21: Mesh levels for simulation M3AMR-005-ANISO with Mach contours of -0.9184 to 2.856 in 30 increments. Mesh legend: Level 1, Level 2, Level 3, Level 4.

4.5. Conclusion

To conclude, all the investigated refinement criteria perform very comparably, the most significant differences among them can be found in the error magnitude evolution during adaption. A rising error leads to difficulties in choosing the correct threshold and automatically leads to poor controllability regarding the mesh size. Due to the discontinuities created by solving the Euler equations, there will be no reduction in error at these points when looking at the difference between fine and coarse-grained solution. Using the relative error is, therefore, less suitable. Methods using the mesh size were able to solve this problem and led to the most favorable results. Nevertheless, and probably especially due to the discontinuities, this case is very usable for adaption since the discontinuities resemble the areas of interest, and all investigated methods did not have problems catching them. One could also argue that this is the reason why AMR has been used with such success for compressible flows so far (see Löhner [12] for numerous applications). The potential mesh savings are significant, and capturing the areas of interest proves to be easy. Even though this computational case is not ideal for anisotropic refinement when using a block-structured mesh, a measurable and consistent reduction in global error level was recorded in comparison to isotropic refinement. Within either isotropic or anisotropic refinement, the quantitative differences in error are small among the error sensors. However, it appears that when looking at the entire cell count range, the Richardson and ETE-based error sensors perform slightly better than the classic curvature error sensor. Nevertheless, all results lie within a range of $\pm 10\%$ in error and therefore any quantitative results should be taken merely as an indication and could potentially change from case to case.

5

Flow around a two-dimensional circular cylinder at Re=100

The second test-case that is used to benchmark the various error sensors is the flow over a two-dimensional cylinder at Re=100. The two-dimensional and laminar setting makes it a good introductory problem for incompressible flow. Further, the presence of an immersed boundary to model the cylinder wall gives this problem more relevance concerning industrial flows. The test case allows to investigate the effect of the AMR routine on a quantity of interest, i.e., lift coefficient and Strouhal number, as well as examining the behavior close to a wall and in the Kármán-vortex street.

The chapter starts by outlining the case setup and a reference solution that is used to rate the results of the different AMR simulations. Next, simulations with the AMR reference error sensors are presented. Subsequently, the newly-developed error sensors are tested in isotropic and anisotropic refinement and compared to the results from the reference error sensor. Two specific aspects are discussed at the end of the chapter. The influence of the adaption time step on the performance of the error sensors and the overhead that is created by using AMR within the simulation.

5.1. Case setup

Table 5.1 shows a selected summary of publications that have studied this case. It becomes obvious that even though the test case is simple, the spread of reported values for drag and lift is large. Therefore the decision was made to compute a reference solution on a completely uniform mesh, which will be used for assessing the performance of the AMR routines. This approach is deemed sufficient since, for this assessment, merely the grid converged solution of INCA is important rather than the true solution.

Table 5.1: Summary of results for flow around a two-dimensional cylinder at Reynolds number of 100. Taken from Meyer *et al.* [5].

Study	C_d	$C_{l,\max}$	St	Type
Fey *et al.* [75]	-	-	0.165	Experiment
Kim *et al.* [14]	1.33	0.32	0.165	Simulation
Dröge and Verstappen [76]	1.24	0.3	0.165	Simulation
Meyer *et al.* [5]	1.26	0.34	0.165	Simulation

For all cases, the reference solution and the AMR calculations the same domain size is chosen. It is shown in Figure 5.1. The domain size for the two-dimensional case is 20 cylinder diameters in height and 40 diameters in length. The center of the cylinder is placed at 10 diameter height and length. The boundaries on the top and bottom of the domain are set to free-slip, the front and rear boundary to periodic. The inlet boundary has a velocity inflow, and the outlet boundary a pressure outlet condition.

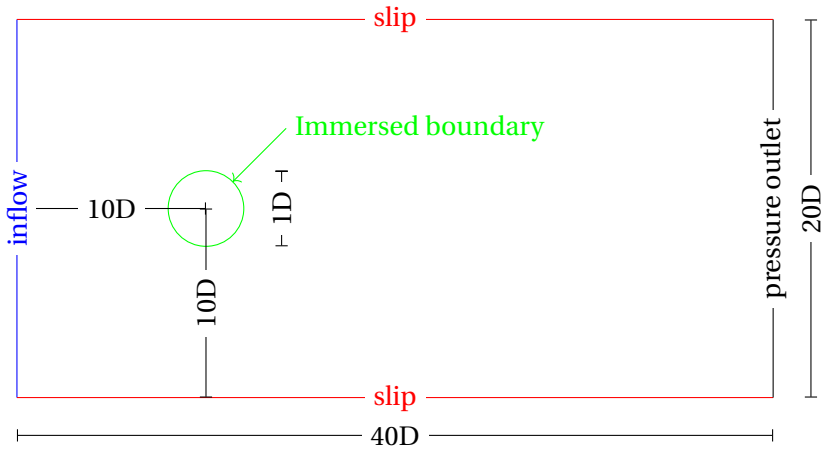


Figure 5.1: Case setup for flow over a circular cylinder at $Re = 100$.

Table 5.2 shows the computed values for the uniform reference solution. In total, four simulations were performed from level two to five. No computation below level two could be performed due to divergence caused by the rough mesh resolution at the immersed boundary. Only computations up to level five were possible due to the enormous cell count of almost 10 million. For all practical calculations within this thesis, level five has been used as the reference solution. Otherwise, the required computational resources would have exceeded the scope and time frame of this thesis.

Table 5.3 shows the initial mesh parameters of the coupled AMR solution. The number of cells was chosen in such a way that the coarse simulation would just not diverge. The mesh of the master and slave simulation after the cut-cell criterion has been used is given in Figure 5.2. Therefore one finds 1.35 and 4.05 cells across the diameter for the slave and master solution, respectively. Enough so that

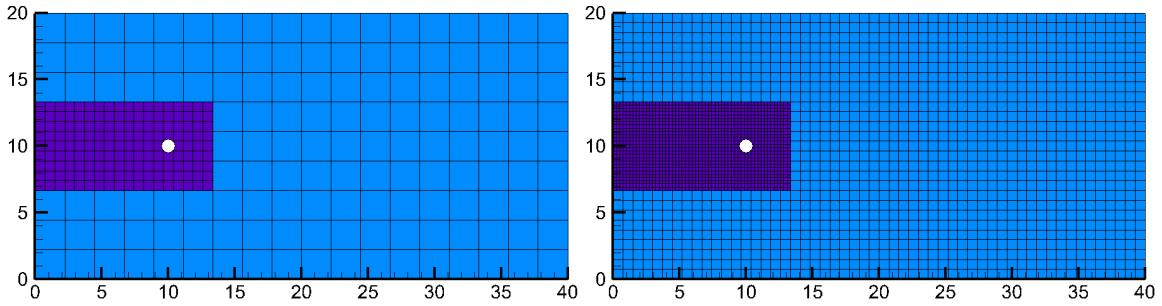
Table 5.2: Reference solution computed with uniform mesh for 2D cylinder.

Mesh level	$C_{l,\max}$	St	NX	NY	Cells
2	0.0420	0.17241	162	81	13,122
3	0.3058	0.16828	486	243	118,341
4	0.3831	0.16850	1,458	729	1,062,882
5	0.3466	0.16919	4,374	2,187	9,565,938

the geometry is just being captured on the slave simulation by the immersed boundary approach.

Table 5.3: Initial mesh for flow over two-dimensional cylinder at Re=100.

	Master	Slave
Cells x-dir	18	6
Cells y-dir	9	3
Cells z-dir	1	1
Cut-cell level	2	2
Number of initial blocks	1	1



(a) Slave simulation (b) Master simulation

Figure 5.2: Mesh for flow over two-dimensional cylinder at Re=100 after cut-cell criterion. Mesh levels: Level 1, Level 2, Level 3, Level 4, Level 5.

5.2. Reference curvature refinement

The undivided difference of curvature of velocity was chosen as the baseline error sensor. For all simulations, the maximum cell level was incrementally increased from 2 to 5. Every computation was initialized with the results from the previous level to save computational cost. No meaningful differences in mesh, and results were observed when the computations were performed from a clean slate. A total of five series of computations were performed to obtain a suitable Pareto front. The case setup of these simulations is summarized in Table 5.4. No results are shown for the level 2 computation since, for that mesh, no vortex street could be observed. It should, however, be noticed that the absence had no influence on the adaption routine, and the error sensor was still able to suggest refinement so that the simulation traversed to higher mesh levels.

Figure 5.4 shows the error in the maximum lift coefficient, and Strouhal number for the curvature adapted simulation. Two sets of graphs have been created. First, the errors were computed by using the uniform level 5 simulation as the exact solution. Secondly, the relative error was computed with the uniform solution of the maximum respective AMR mesh level as the reference. This was based on the argument that the 'correct' result of an AMR simulation fixed to a maximum mesh level is the

Table 5.4: Simulation settings for curvature refinement of 2D-cylinder.

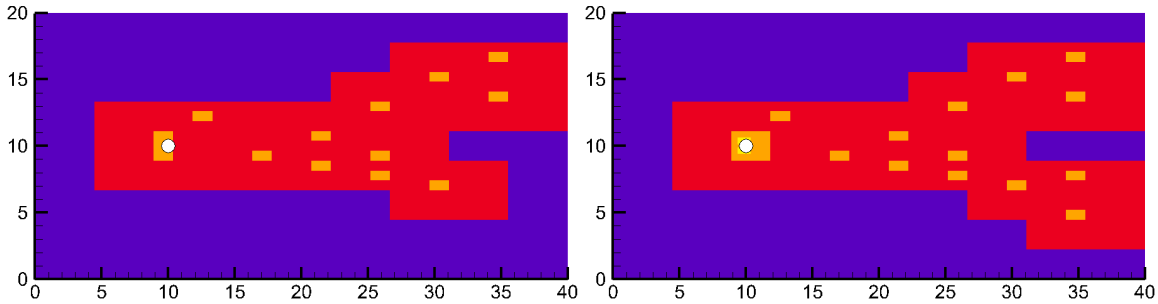
Legend	Identifier	Levels	ϵ_{AMR}	τ_{AMR} (s)	t_{sim} (s)	Error sensor
×	REF-0001	3-4-5	0.01	0.25	70	Eq. 2.3 with $\|U\|$
+	REF-0002	3-4-5	0.02	0.25	70	Eq. 2.3 with $\|U\|$
○	REF-0004	3-4-5	0.04	0.25	70	Eq. 2.3 with $\|U\|$
□	REF-0010	3-4-5	0.10	0.25	70	Eq. 2.3 with $\|U\|$
△	REF-0025	3-4-5	0.25	0.25	70	Eq. 2.3 with $\|U\|$

one from the uniform mesh of the same level. Or said differently, when an adapted mesh leads to a solution that is closer to the true solution (e.g., obtained from a DNS, or a higher mesh level) than the uniform mesh, then this is the result of erroneous error cancellation within the domain. With the assumption that no error sensor studied in this thesis has any obvious properties that lead to beneficial error cancellation, the above-described methodology is justified.

Figure 5.4a and 5.4b show the error in lift coefficient in comparison to the results from the uniform level 5 computation. The error is plotted against the cell count and computational cost. A new Poisson-timestep-cell (PTC) metric was introduced in Appendix B to represent computational cost. The metric is based on the argument that the number of Poisson solver iterations, the cell count, and the number of time steps have the largest impact on the wall-clock time of an incompressible simulation in INCA.

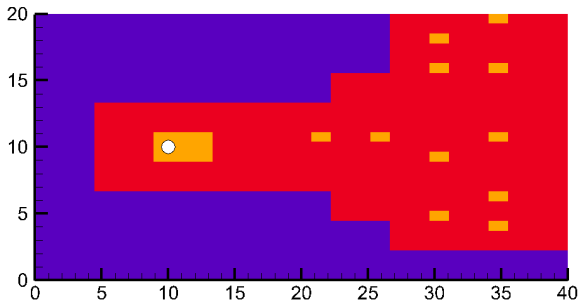
At levels 3 and 5, the results are as one would expect, the lower the error threshold, the lower the actual error in the lift. This pattern is only broken at level 4, where the errors of the simulation with the highest threshold are the lowest. When, however, looking at Figure 5.4c and 5.4d, this anomaly does not take place, and a lower threshold leads to a lower intra-level error. Interestingly, there is an increase in inter-level error when using the relative comparison. This is a good example of highlighting the problems of defining a fair reference when performing AMR. While the simulations from level 3 to 5 span a cell count of approximately one order of magnitude, they span up to four orders of magnitude for the PTC count. This nicely highlights that the surge in computational cost from increasing the cell level does not necessarily stem from the higher cell count, but rather by the need to compute more iterations to not exceed a particular CFL number and to cope with a worse conditioned Poisson matrix. The error in Strouhal number behaves differently. In both comparisons, the error decreases with increasing mesh level. At level 3, the errors across all refinement thresholds are almost identical. This changes slightly at level 4, with the simulation 'REF-0025' falling behind slightly. When reaching level 5, there is a clear difference across all simulations, with expected results: A lower error threshold leads to a lower error.

Figure 5.3 shows the resulting mesh of all five error thresholds at levels 4 and 5. The curvature error sensor leads to sensible refinement in the vicinity of the cylinder. At level 5, in simulations 'REF-0025' to 'REF-0004', only the first half of the cylinder gets refined, whereas the backface does not see any refinement. It is also very apparent that substantial refinement of the far-field wake takes place. The fan shape of the refinement wake can be explained by the fact that the vortex street diffuses in y-direction when traveling downstream. Small spikes in curvature lead to speckles of refinement zones in the far-field wake. Since a block was refined even if only a single cell violated the error threshold, there is some inherent vulnerability to these speckles. Judging the result so far, then one can be quite pleased with the results. The curvature criterion has shown that it can create a mesh that leads to sensible results on the one hand but also reduces the overall cell count significantly on the other. The ease of implementation adds on top.

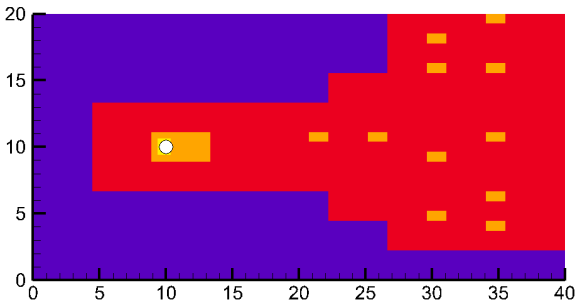


(a) REF-0025 - Level 4

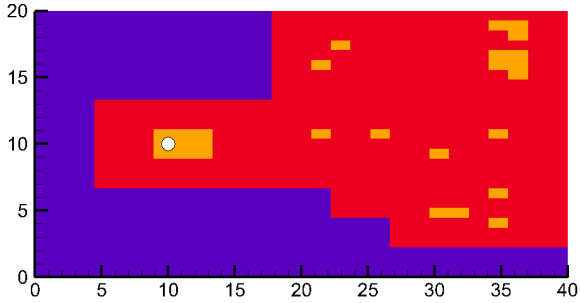
(b) REF-0025 - Level 5



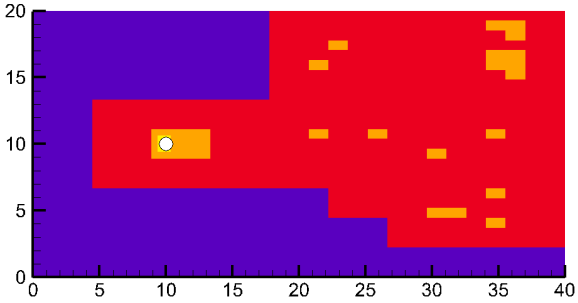
(c) REF-0010 - Level 4



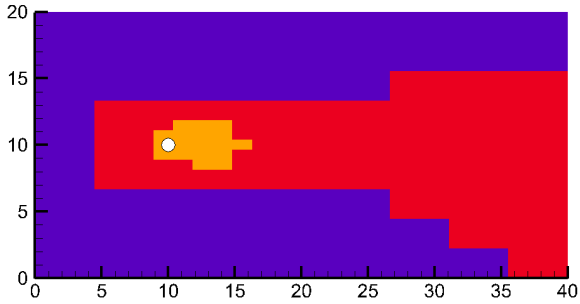
(d) REF-0010 - Level 5



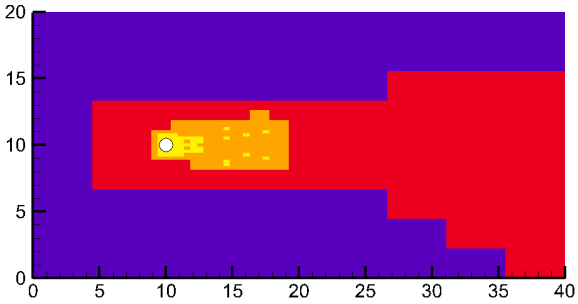
(e) REF-0004 - Level 4



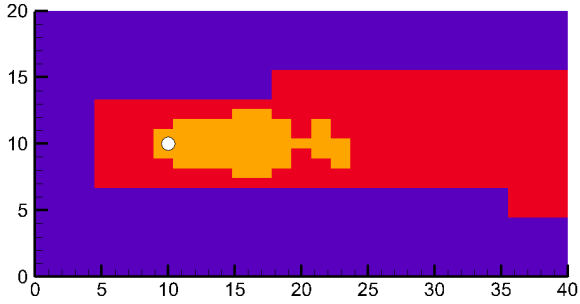
(f) REF-0004 - Level 5



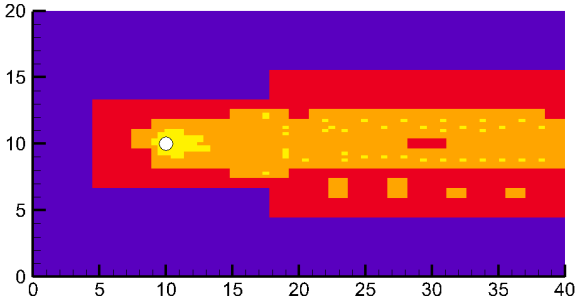
(g) REF-0002 - Level 4



(h) REF-0002 - Level 5

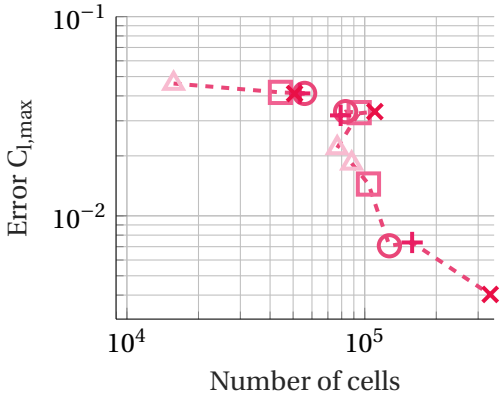


(i) REF-0001 - Level 4

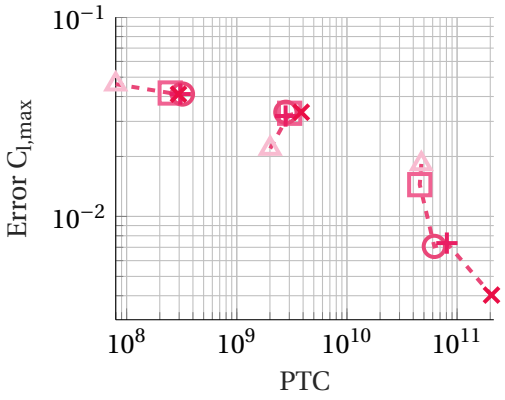


(j) REF-0001 - Level 5

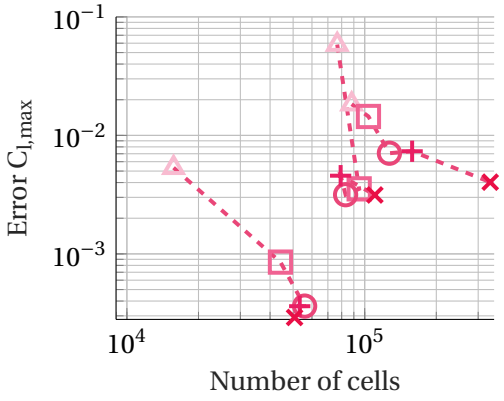
Figure 5.3: Mesh levels created by the reference curvature error sensor. Mesh levels: Level 1, Level 2, Level 3, Level 4, Level 5.



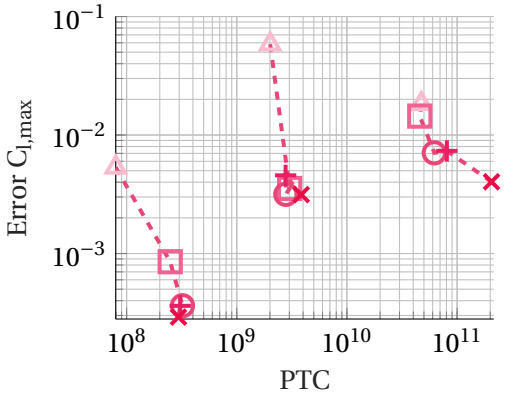
(a) w.r.t uniform level 5 mesh



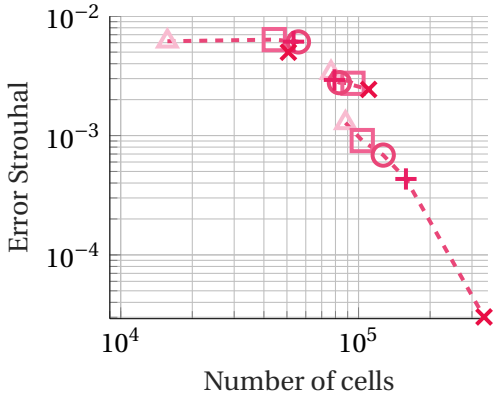
(b) w.r.t uniform level 5 mesh



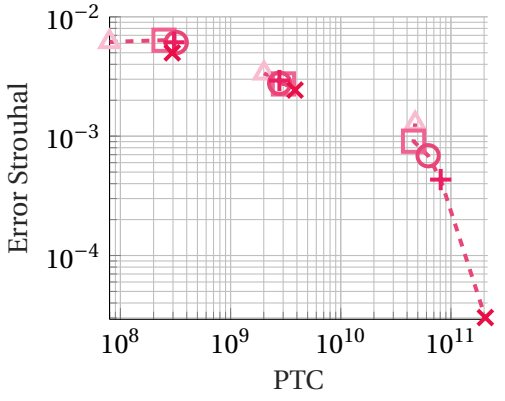
(c) w.r.t uniform mesh of corresponding level



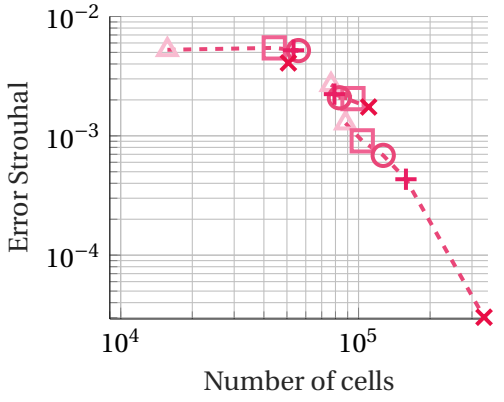
(d) w.r.t uniform mesh of corresponding level



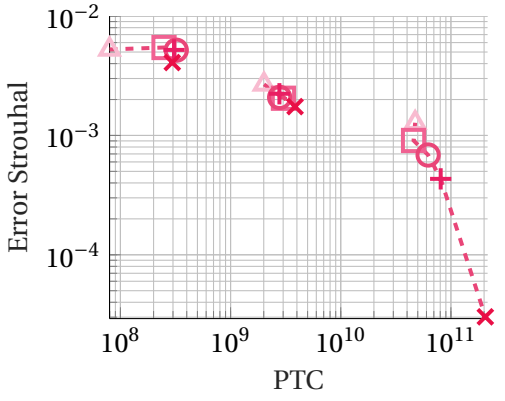
(e) w.r.t uniform level 5 mesh



(f) w.r.t uniform level 5 mesh



(g) w.r.t uniform mesh of corresponding level



(h) w.r.t uniform mesh of corresponding level

Figure 5.4: Errors in maximum lift coefficient and Strouhal number for curvature error sensor. REF-0004 (○), REF-0010 (□), REF-0025 (△), REF-0002 (+), REF-0001 (×).

5.3. Error sensors in isotropic adaption

First, the performance of the Richardson and ETE-type error sensors is mapped out for an isotropically refined mesh. The simulation parameters are shown in Table 5.5. The same error sensors were investigated as for the Mach 3 shock case. For every error sensor, the goal was to obtain three series of computations, all traversing from levels 2 to 5. Every series was performed with one particular error threshold. The thresholds were determined, such that approximately the same cell range was spanned as for the curvature error sensor discussed in the previous section. This required a considerable amount of trial and error. This aspect will be discussed further in the conclusion of this chapter. For simulation IS-004, it was only possible to perform one simulation. This was because the error would increase with progressing refinement i.e., the same behavior as shown in Figure 4.2. This led to excessively fine meshes and thus unfeasible long wall clock time.

Table 5.5: Testing matrix for two-dimensional circular cylinder at $Re = 100$ using isotropic refinement.

Legend	Identifier	Levels	ϵ_{AMR}	τ_{AMR}	$t_{\text{sim}}(s)$	Error sensor	Figure
	IS-001-0004	3-4-5	0.04	0.25	70	Eq. 3.4 with $\|U\|$	5.5
	IS-001-0010	3-4-5	0.10	0.25	70	Eq. 3.4 with $\|U\|$	5.5
	IS-001-0025	3-4-5	0.25	0.25	70	Eq. 3.4 with $\|U\|$	5.5
	IS-002-0001	3-4-5	0.01	0.25	70	Eq. 3.9	5.6
	IS-002-0003	3-4-5	0.03	0.25	70	Eq. 3.9	5.6
	IS-002-0004	3-4-5	0.04	0.25	70	Eq. 3.9	5.6
	IS-003-0004	3-4-5	0.04	0.25	70	Eq. 3.10	5.6
	IS-003-0006	3-4-5	0.06	0.25	70	Eq. 3.10	5.6
	IS-003-0010	3-4-5	0.10	0.25	70	Eq. 3.10	5.6
	IS-004-0075	3-4-5	0.75	0.25	70	Eq. 3.11	5.7
	IS-005-0002	3-4-5	0.02	0.25	70	Eq. 3.5 with $\|U\|$	5.5
	IS-005-0004	3-4-5	0.04	0.25	70	Eq. 3.5 with $\|U\|$	5.5
	IS-005-0005	3-4-5	0.05	0.25	70	Eq. 3.5 with $\|U\|$	5.5
	IS-007-000001	3-4-5	0.0001	0.25	70	Eq. 3.7 with $\|U\|$	5.7
	IS-007-000002	3-4-5	0.0002	0.25	70	Eq. 3.7 with $\|U\|$	5.7
	IS-007-000004	3-4-5	0.0004	0.25	70	Eq. 3.7 with $\|U\|$	5.7

The results, i.e., the errors in maximum lift coefficient and Strouhal number can be found in Figure 5.5 to 5.7. The results were split up into three sets of graphs to keep them readable. The resulting meshes for level 5 are shown in Figure 5.8. For comparison, the Pareto front of the reference solution was included in all figures. From a first glance, the results are very different across the error sensors, as opposed to the Mach 3 forward-facing step flow, where the results were very similar. However, all error sensors lead to a consistent reduction in absolute error for the lift coefficient and the Strouhal number with increasing mesh level. Continuing with the analysis, for levels 3 and 4, the main differences in the solution are either the cell count or the PTC, while the actual error stays fairly constant across every level. Thus, both quantity of interests do not seem to be affected considerably by the actual mesh. This can be attributed to the fact that the errors created at the cylinder are dominating while any differences in wake refinement are only of minor importance. The situation changes at the subsequent level. The general error has dropped in both metrics drops by up to one order of magnitude, and now distinct performance differences can be spotted. At this level, some simulations reach an error of approximately 1% for the lift coefficient and approximately 0.5% for the Strouhal number. At this stage, the influence of the refinement of the wake becomes more prominent on the lift coefficient and Strouhal number and leads to measurable differences. When looking at the relative error in the lift coefficient and Strouhal number, then the effects of the dominant

immersed boundary get filtered out, and differences in error magnitude can be spotted for levels 3 and 4. Nevertheless, they remain much closer to each other than for the level 5 computations. The relative error also has two other interesting properties. While there is a steady reduction in error for the Strouhal number with increasing mesh level, the lift coefficient only shows this reduction when going from level 3 to 4. From level 4 to 5 the error actually grows, so far that it actually exceeds the ones from level 3.

Figure 5.5 contains the results of simulations 'IS-001' and 'IS-005'. 'IS-001' uses the relative error in velocity magnitude as the error sensor, while 'IS-005' uses the absolute error of the same. All 'IS-001' simulations show a better computational efficiency in relative and absolute error for the lift coefficient than the reference solution. Except for $\epsilon_{\text{AMR}} = 0.25$, the same also holds for the Strouhal number. For the same error threshold, one should also note the very low error in the lift coefficient for the level 5 simulation. This can probably be attributed as a lucky circumstance due to favorable error cancellation. The quantitative results look promising when comparing to the reference error sensor. At level 5, 'IS-001-0010' has approximately four times higher computational efficiency for the lift coefficient than the reference. This number only drops slightly to 3.3 for 'IS-001-0004'. The savings are more moderate for the Strouhal number. Here savings in cells and PTC of around 1.2 to 2.1 can be achieved. Interestingly, for $\epsilon_{\text{AMR}} = 0.25$, one can observe a drastic influence of an ill-conditioned Poisson matrix when looking at Figure 5.5f. This stems most likely from the fragmented mesh, as shown in Figure 5.8a. No consistent refinement around the cylinder takes place. Speckles of level 5 refinement zones are being placed randomly around the cylinder and in the near wake. Leading to a very unsymmetrical and chaotic mesh. However, when the error threshold is lowered to $\epsilon_{\text{AMR}} = 0.10$, an almost symmetric mesh is formed, which merely refines the immersed boundary and the near wake up to $x = 12$ with the highest refinement level. Lowering the threshold even further to $\epsilon_{\text{AMR}} = 0.04$ extends the near wake refinement until approximately $x = 13$. Larger changes happen in the level 3 zone which gets extended from $x = 17$ to $x = 27$. The mesh also loses its near-perfect symmetry.

The IS-005 simulations cannot keep up with the excellent performance of the IS-001 ones. Concerning the cell count, the IS-005 series can still produce higher computational efficiency in the lift coefficient and Strouhal number than the reference indicator. The exception being 'IS-005-0002' at level 5, which requires about double the cell count than the reference simulation. The picture changes when the PTC count is used. For levels 3 and 4, the computational efficiency is still higher, but when going down to level 5, the performance deteriorates to just about equal or worse than the reference error sensor. In contrast to 'IS-001', 'IS-005' puts more emphasis on the refinement of the wake. At $\epsilon_{\text{AMR}} = 0.05$, one can already start to see that the refinement zone around the cylinder is wider. At this threshold, still, a reasonably symmetric mesh is generated. Further, the only area which is getting refined with a level 5 mesh is the front half of the cylinder. With a decreasing error threshold, the wake is getting refined more and more, until at $\epsilon_{\text{AMR}} = 0.02$ two level-4-trains are extending until almost the end of the domain. Both trains are covered with level 5 refinement speckles. The level 5 refinement of the immersed boundary covers the entire cylinder but can be considered as very uneven. In general, the empirical quality of the mesh cannot be considered as good, as for 'IS-001'. The results of 'IS-001' and 'IS-005' are very interesting since they, at least for this test case, can answer the question on whether one should refine areas of large absolute error, as the IS-005 simulations do, or refine areas of large relative error. Since the latter is more successful, one can deduce that errors in 'weak features' within a solution can have a more significant impact on the global result than the large absolute errors of the strong features. An argument also supported by Löhner [12] for compressible cases.

The results of error sensors 'IS-002' and 'IS-003' are depicted in Figure 5.6. At level 3, all 'IS-002' simulations pretty much lie on the reference Pareto front. For the Strouhal number, the front is even

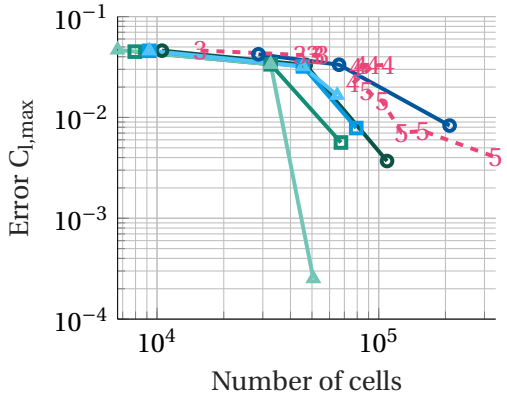
getting exceeded. Simulation '-0003' and '-0004' suddenly start to perform much better at level 4. Here we can observe meshes with half the cell count for the same error in comparison to the reference. The same performance gains can also be found when looking at the PTC count. At level 5, '-0003' and '-0004' can still lead to cell savings in the order of 30 to 40%. Unfortunately, when looking at the PTC count, the performance of '-0004' deteriorates significantly and now lies on the Pareto front. The computational efficiency of '-0001' for the lift coefficient error is comparable to the reference and sometimes even slightly worse. When looking at the Strouhal number, then the performance is slightly better at level 4 but completely crumbles at level 5. The mesh generated by 'IS-002' is again fairly wake-dominant. At $\epsilon_{\mathrm{AMR}} = 0.04$, only the front half of the cylinder is refined with the maximum level. Stepping to the next threshold leads to a mesh covering the entire immersed boundary with a level 5 mesh. At this point again, two level-4-refinement-zone-trains start to form. At the lowest threshold, also the near wake is refined with a level 5 mesh. Unfortunately, the mesh is again not very symmetric and contains many speckles. Both wake trains now also reach the end of the domain.

In comparison to 'IS-002' 'IS-003' shows a weaker performance. At level 3, all simulations have passed the Pareto front of the reference for the lift coefficient. The situation is similar to the Strouhal number. The exception being '-0006', which can stay ahead of the reference performance. The computational efficiency of the error sensors increases slightly at level 4. Simulation '-0010' can even save around 50% of cells in comparison to the reference when looking at the lift coefficient and even slightly more when looking at the Strouhal number. When using the PTC metric, these gains increase to about 60%. The two other computed error levels show roughly identical performance for the lift coefficient in comparison to the reference computation. However, for the Strouhal number, they can perform up to 25% better. At level 5, the overall performance worsens again. While '-0010' is still a good contender, leading to savings in cells of about 35% and 20% in PTC, '-0006' and '-0004' perform either worse or equal to the reference throughout all metric. A tiny exception being '-0004', which can save about 25% cells for the lift coefficient. Unfortunately, these gains do not translate to PTC savings also. At the highest threshold, only the front face is refined with a level 5 mesh, and the immersed boundary already sees a substantial amount of lateral refinement at the bottom and top of the cylinder. At the lowest threshold, the mesh shares some resemblance with the one from 'IS-001-0004'. The near wake sees equal level 5 refinement up to approximately the same x-location, even though the mesh can not be considered as being as uniform. Further, the entire wake is also refined, but only with a level 3 mesh.

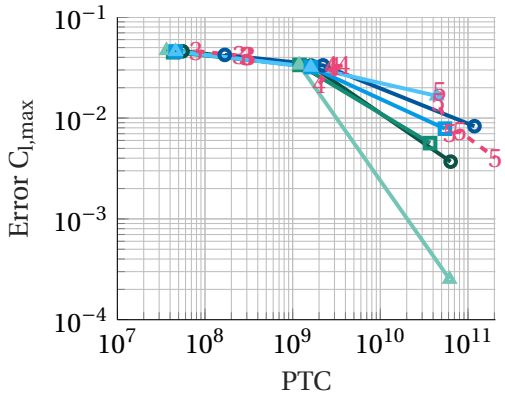
Figure 5.7 shows the last two error sensors. Unfortunately for 'IS-004', only one simulation series was performed. The error sensor was so sensitive to the exact refinement threshold that either almost no refinement took place or that so much refinement happened that a mesh of at least 1 million cells was obtained. Nevertheless, some interesting observations can be made. At level 3, the performance is about equal to the reference error sensor. At level 4, however, the lowest error in the lift coefficient of all simulations is obtained, while maintaining a cell count of only 50,000. The error is so low that the reference error sensor is only able to achieve it with a level 5 mesh. When looking at the relative error, one can spot that it is surprisingly large. The conclusion is thus that the low error stems from favorable error cancellation. It is also interesting to see that the error in the lift coefficient stays constant when going from level 4 to 5. At the latter level, the error sensor penetrated the Pareto front for both metrics. The resulting mesh is also noteworthy. The front face of the cylinder sees level 5 refinement, while the back face stays at level 4. However, further downstream, two large blobs of level 5 refinement are formed. Additionally, some level 5 refinement speckles can also be found in the far-wake.

At level 3, the simulations of 'IS-007' perform equal or worse than the reference error sensor. Stepping up to level 4, some savings can be found. Simulations '-000002' and '-000004' give around 10 to

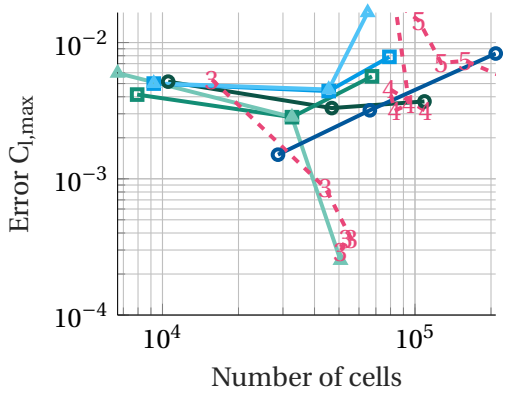
15% savings in cells and PTC for the lift coefficient error. The same error in the Strouhal number can be realized for about 40% fewer cells and PTC in simulation '-000004'. Simulation '-000002' can still deliver 15% savings in cell count but performs as good as the reference when using PTC count. At level 5, all error sensors again perform equal or worse than the reference. Noteworthy is simulation '-000002', which performs as good as the reference for the cell count, but when using PTC as the benchmark, it takes about twice the time to compute. The effects of an ill-conditioned Poisson matrix are thus clearly visible. When looking at the mesh, one can see that the error sensor puts, when compared to all other error sensors, the most emphasis on wake refinement. Refinement at the immersed boundary is only taking place at the front face and the sides. At the highest threshold, there is no level 5 refinement, whereas large parts of the domain after the cylinder are already refined with level 3. At $\epsilon_{\mathrm{AMR}} = 0.0001$ still, only the front and sides of the cylinder see a level 5 refinement, but almost the entire domain is already refined with a level 3 mesh, and also the extent of the level 4 mesh wake refinement is considerable.



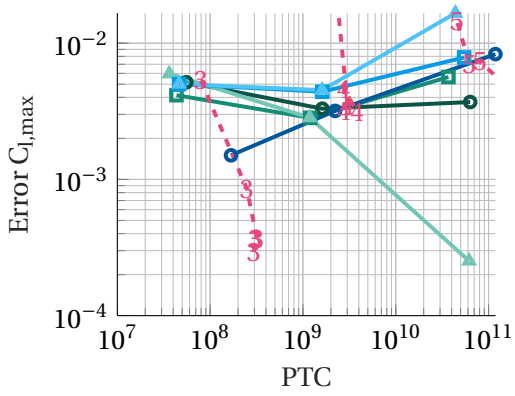
(a) w.r.t uniform level 5 mesh



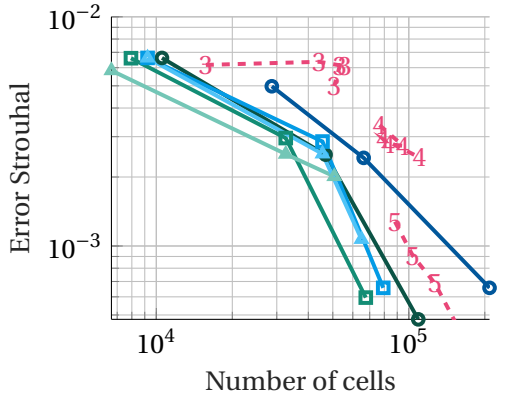
(b) w.r.t uniform level 5 mesh



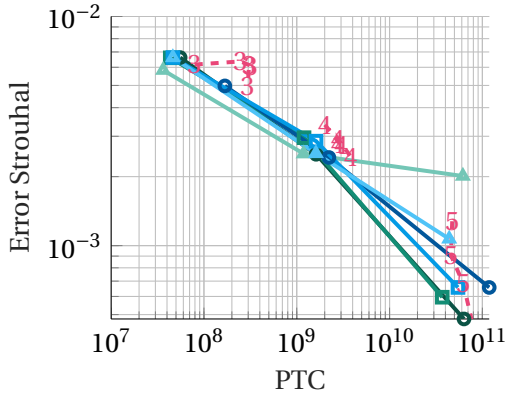
(c) w.r.t uniform mesh of corresponding level



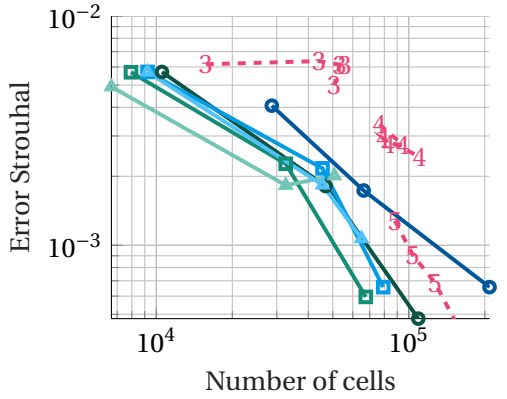
(d) w.r.t uniform mesh of corresponding level



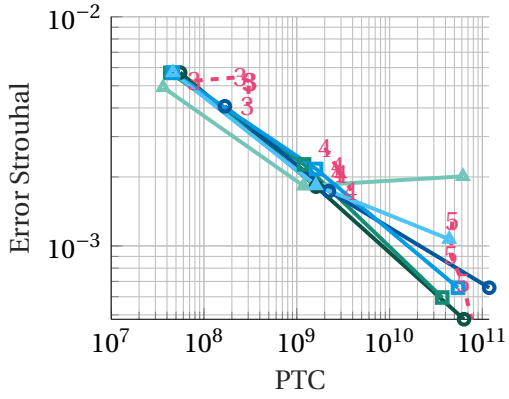
(e) w.r.t uniform level 5 mesh



(f) w.r.t uniform level 5 mesh

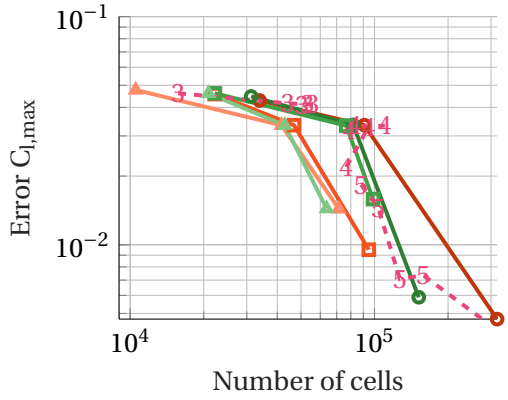


(g) w.r.t uniform mesh of corresponding level

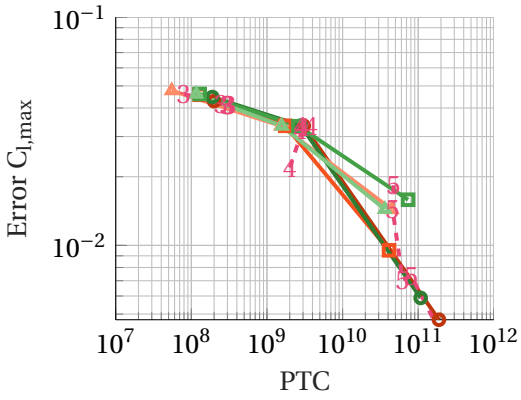


(h) w.r.t uniform mesh of corresponding level

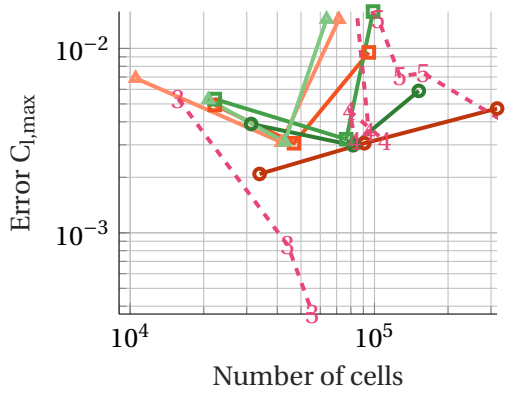
Figure 5.5: Errors in maximum lift coefficient and Strouhal number for isotropic refinement. REF (- - -), IS-001-0004 (—o—), IS-001-0010 (—□—), IS-001-0025 (—△—), IS-005-0002 (—o—), IS-005-0004 (—□—), IS-005-0005 (—△—).



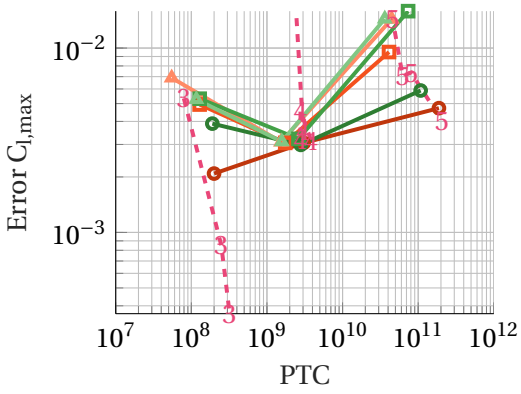
(a) w.r.t uniform level 5 mesh



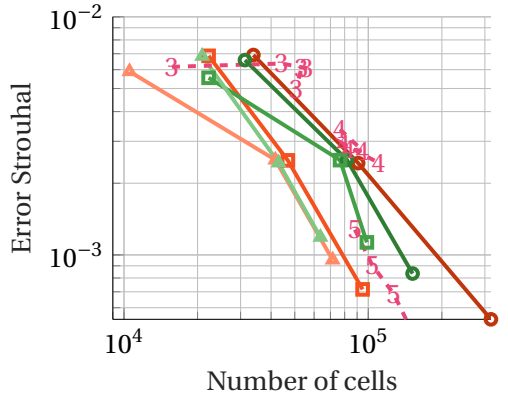
(b) w.r.t uniform level 5 mesh



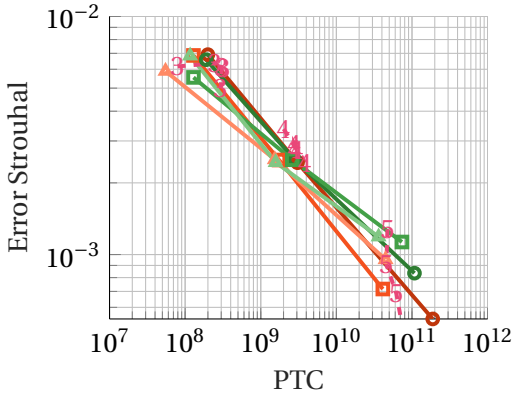
(c) w.r.t uniform mesh of corresponding level



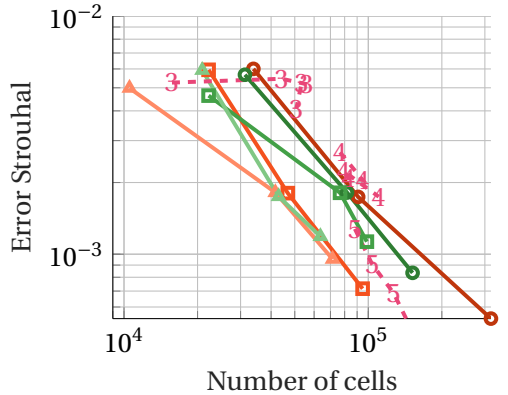
(d) w.r.t uniform mesh of corresponding level



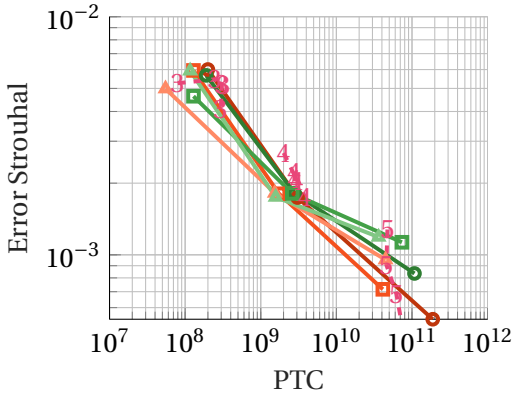
(e) w.r.t uniform level 5 mesh



(f) w.r.t uniform level 5 mesh

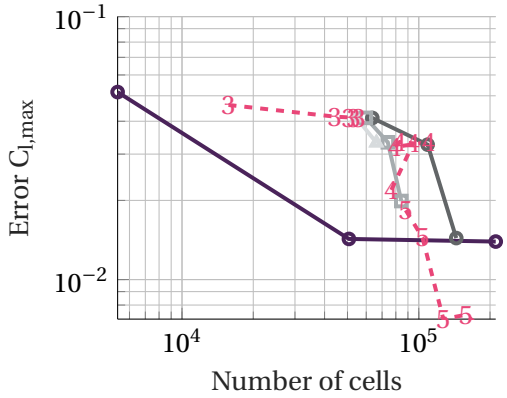


(g) w.r.t uniform mesh of corresponding level

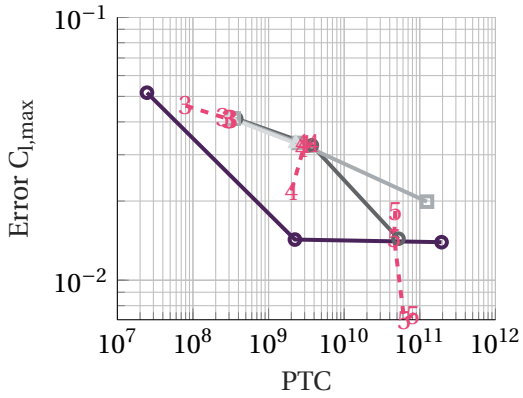


(h) w.r.t uniform mesh of corresponding level

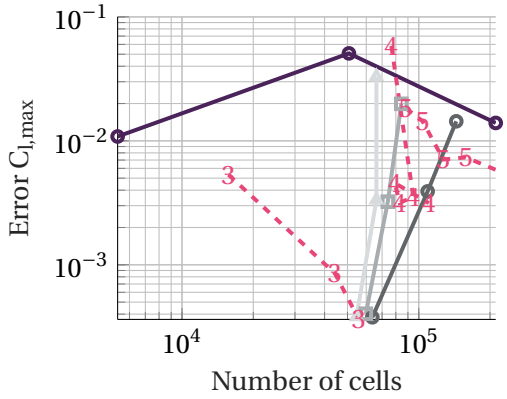
Figure 5.6: Errors in maximum lift coefficient and Strouhal number for isotropic refinement. REF (- - -), IS-002-0001 (—o—), IS-002-0003 (—□—), IS-002-0004 (—△—), IS-003-0004 (—o—), IS-003-0006 (—□—), IS-003-0010 (—△—).



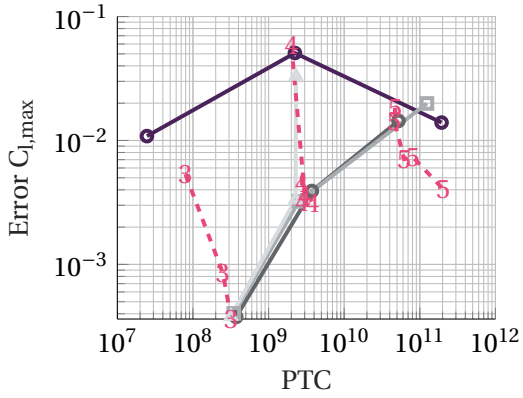
(a) w.r.t uniform level 5 mesh



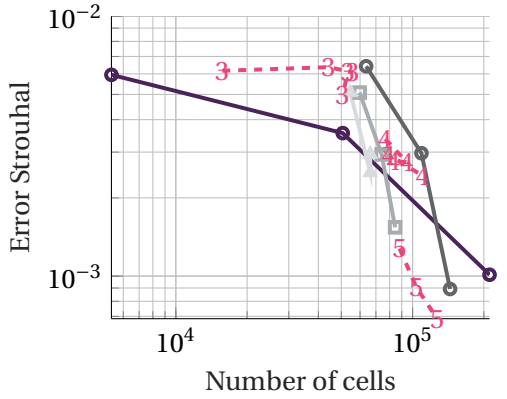
(b) w.r.t uniform level 5 mesh



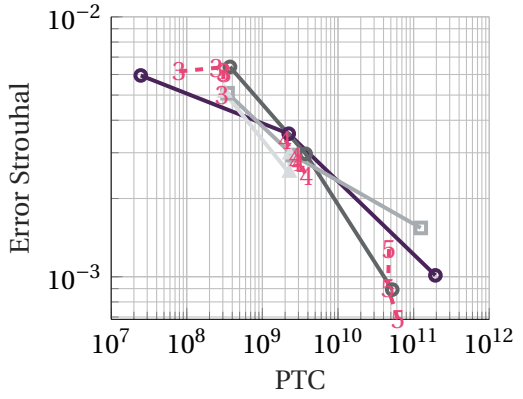
(c) w.r.t uniform mesh of corresponding level



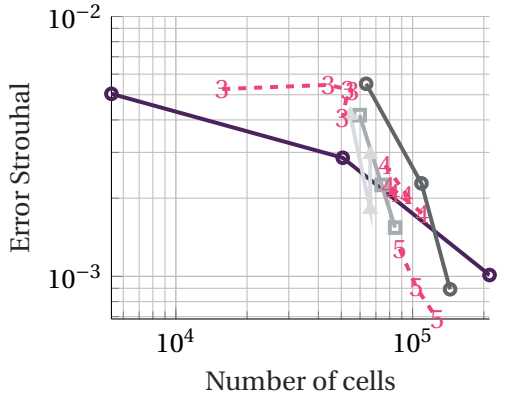
(d) w.r.t uniform mesh of corresponding level



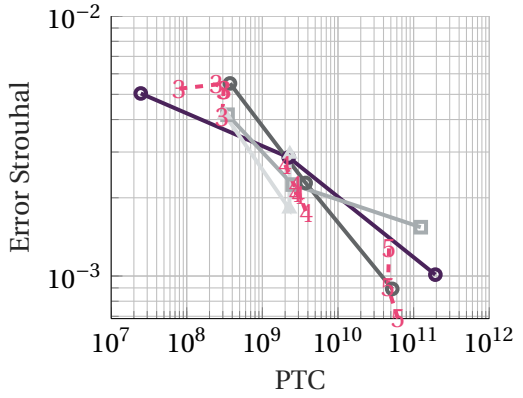
(e) w.r.t uniform level 5 mesh



(f) w.r.t uniform level 5 mesh

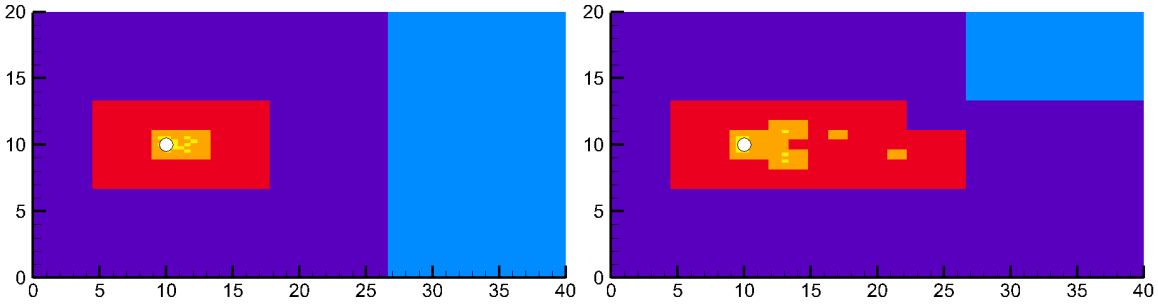


(g) w.r.t uniform mesh of corresponding level



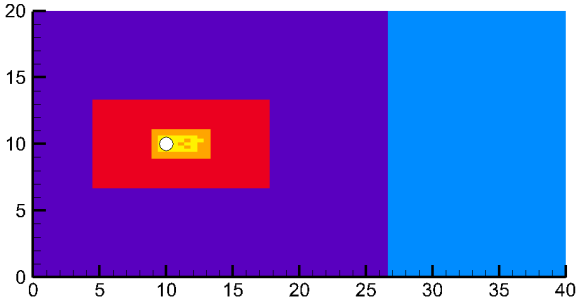
(h) w.r.t uniform mesh of corresponding level

Figure 5.7: Errors in maximum lift coefficient and Strouhal number for isotropic refinement. REF (- - -), IS-004-0075 (—●—), IS-007-000001 (—●—), IS-007-000002 (—□—), IS-007-000004 (—△—).

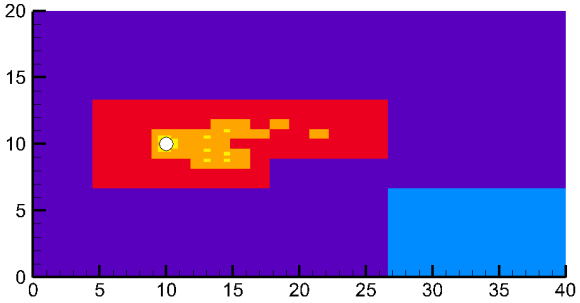


(a) IS-001-0025 - Level 5

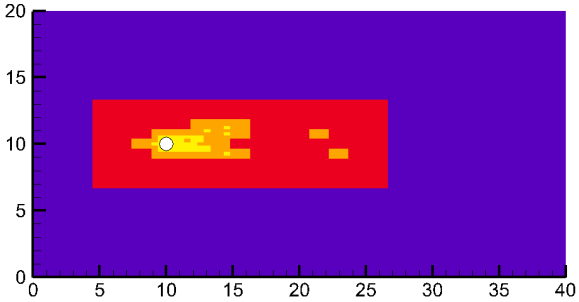
(b) IS-005-0005 - Level 5



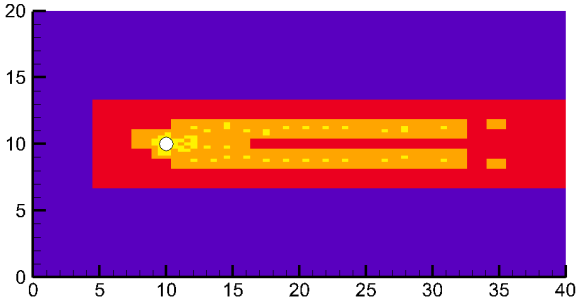
(c) IS-001-0010 - Level 5



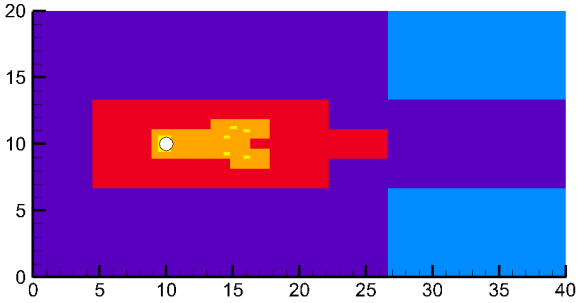
(d) IS-005-0004 - Level 5



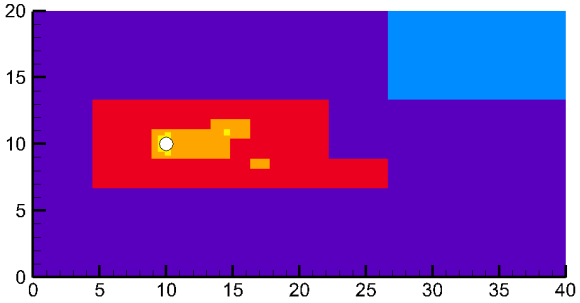
(e) IS-001-0004 - Level 5



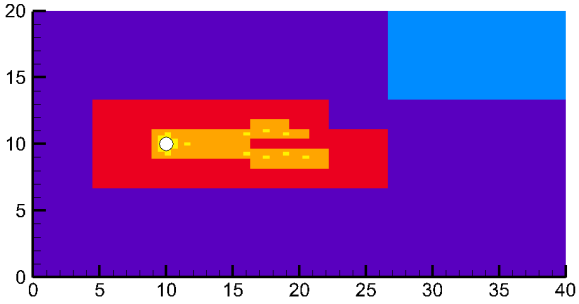
(f) IS-005-0002 - Level 5



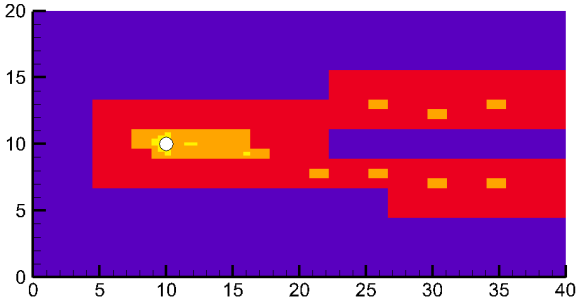
(g) IS-002-0004 - Level 5



(h) IS-003-0010 - Level 5

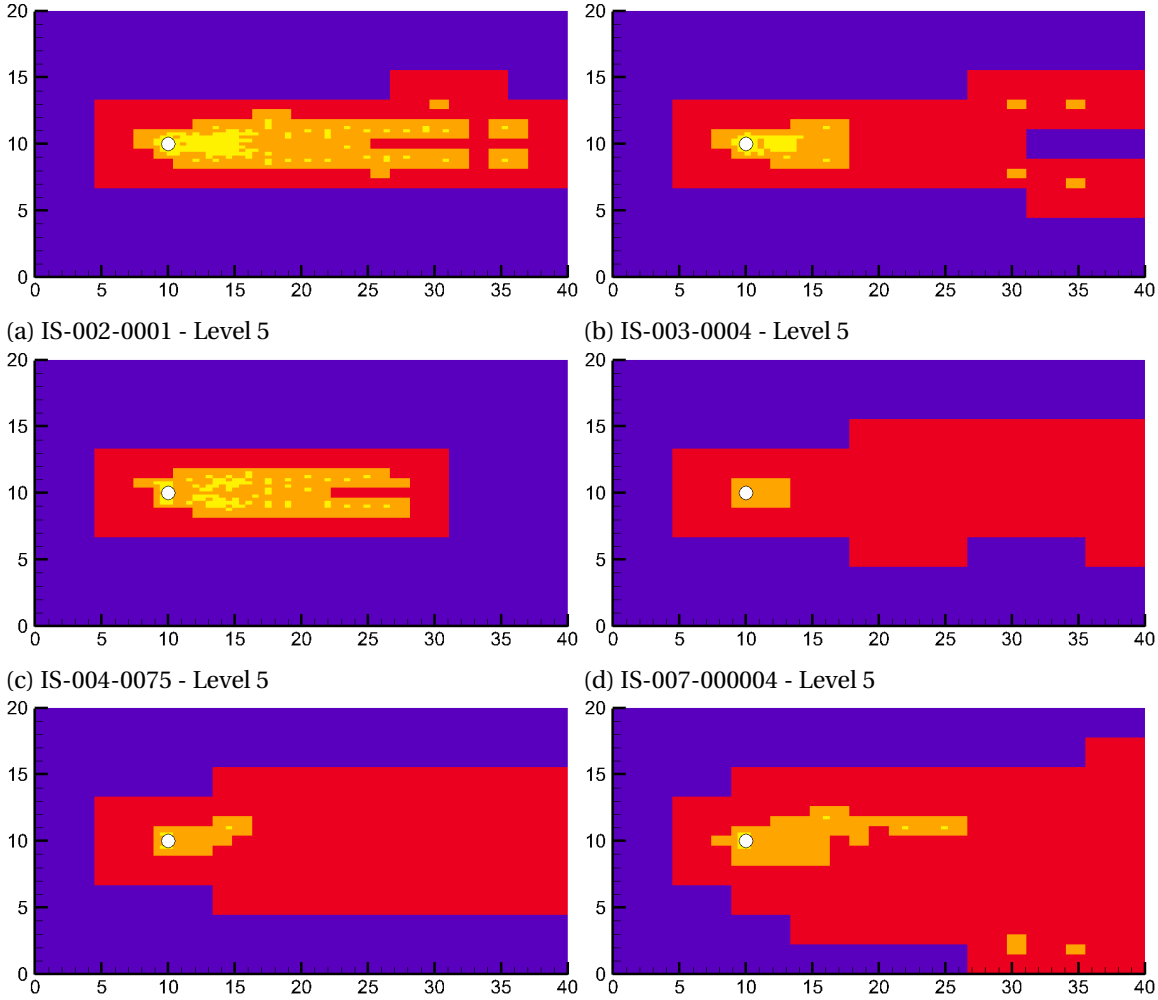


(i) IS-002-0003 - Level 5



(j) IS-003-0006 - Level 5

Figure 5.8: Mesh levels created by isotropic refinement part 1. Mesh levels: Level 1, Level 2, Level 3, Level 4, Level 5.



(a) IS-002-0001 - Level 5

(b) IS-003-0004 - Level 5

(c) IS-004-0075 - Level 5

(d) IS-007-000004 - Level 5

(e) IS-007-000002 - Level 5

(f) IS-007-000001 - Level 5

Figure 5.9: Mesh levels created by isotropic refinement part 2. Mesh levels: Level 1, Level 2, Level 3, Level 4, Level 5.

5.4. Error sensors in anisotropic adaption

The 2D cylinder case is potentially a good candidate for anisotropic refinement. The wake of the cylinder, when statistically averaged, contains flow structures mainly in the lateral direction. Refinement focused on y-direction could, therefore, lead to meaningful savings in computational resources. Table 5.6 shows the list of computational experiments used for this investigation. Two error sensors have been recomputed with anisotropic refinement (Algorithm 4). This includes all simulations of the reference error sensor with and the two lowest error thresholds of the relative error Richardson sensor. The latter was chosen since it showed the best overall performance for isotropic refinement. The errors in Strouhal and maximum lift coefficient are shown in Figure 5.11. The mesh for the maximum refinement level 5 is depicted for the x and y-direction in Figure 5.10.

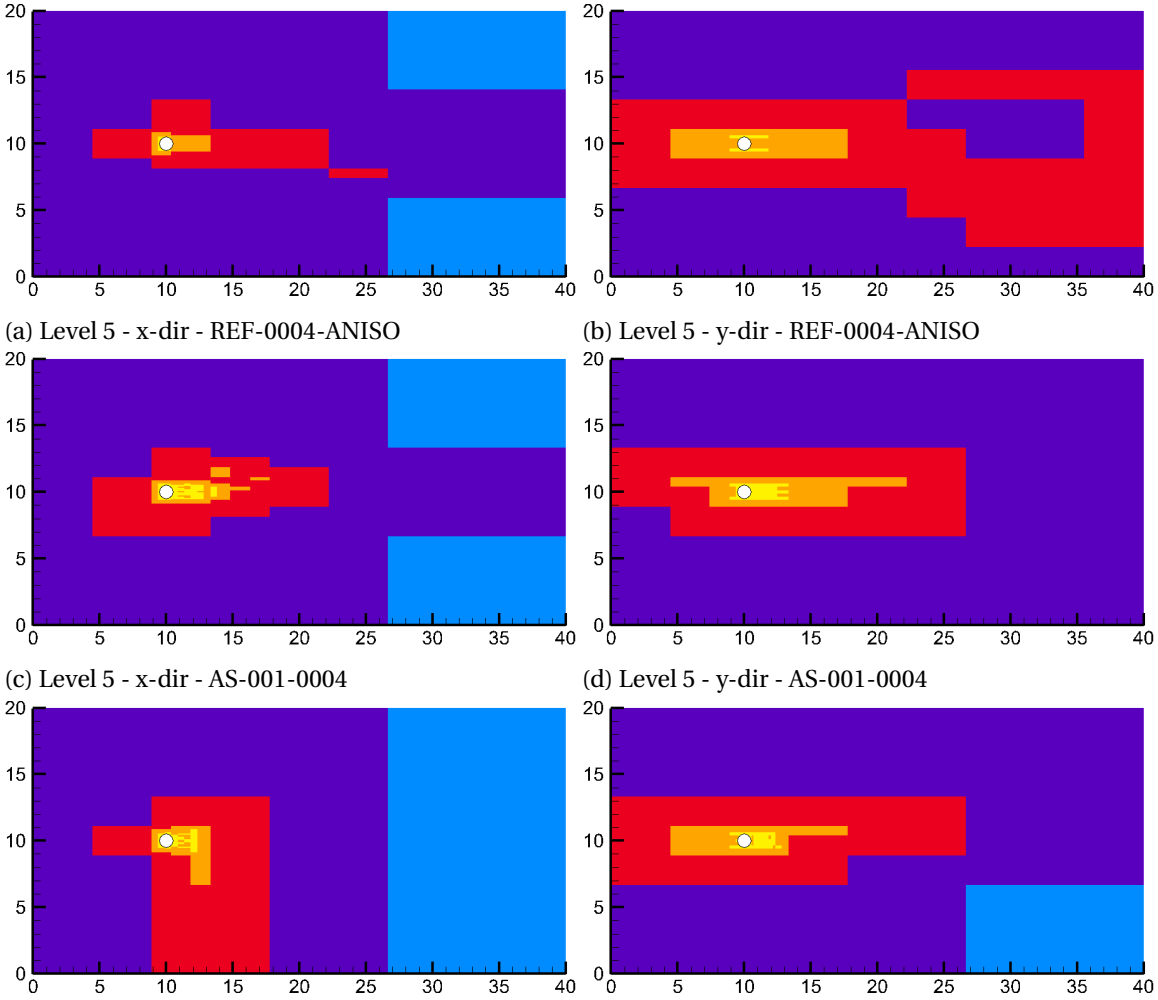
Table 5.6: Testing matrix for two-dimensional circular cylinder at $Re = 100$ using anisotropic refinement.

Legend	Identifier	Levels	ϵ_{AMR}	τ_{AMR}	$t_{\mathrm{sim}}(s)$	Error sensor
- ● -	AS-001-0004	3-4-5	0.04	0.25	70	Eq. 3.4 with $\|U\|$
- ■ -	AS-001-0010	3-4-5	0.10	0.25	70	Eq. 3.4 with $\|U\|$
- - -	REF-ANSIO	3-4-5	0.01,0.02,0.04,0.10,0.25	0.25	70	Eq. 2.3 with $\|U\|$

The isotropic reference simulations led to a rather monotonic Pareto front, the results of the anisotropic refinement are, however, more erratic. Often different levels overlap, or error reduction stagnates on either level 4 of 5. This means that unlike the isotropic refinement where a lower threshold leads to more computational cost but also a lower error, the anisotropic refinement is much more sensitive on the particular error threshold. Considering merely the cells as the performance metric, one can find about equal performance between the isotropic and anisotropic error sensor at level 3. Large cell savings can be observed at levels 4 and 5. At level 4, the cell count is cut in half while still maintaining the same error in the lift coefficient and Strouhal number. At level 5, one can also find savings of up to 50% for the Strouhal number for the higher error thresholds. At the lowest threshold, savings of only up to 15% can be found. The result at level 5 for the lift coefficient can be regarded as somewhat peculiar. On the highest threshold, an error is obtained, which lies in the region of a level 3 simulation. When going to the next lower threshold, one finds that the cell count has actually decreased, which is also counter-intuitive. Reducing the error threshold even further, the actual error in the lift coefficient moves sideways with only a very slight error reduction with increasing mesh size. The error is now on the level of the isotropic error sensor. Unfortunately, when one uses the PTC count to determine the computational cost, the situation changes, and all gains are largely negated. On level 4, we can find in both the quantities of interest savings of up to 25 to 50%. However, there are also error thresholds where the anisotropic error sensor performs equal or even worse than the isotropic one. At level 5, the tide turns completely against the anisotropic error sensor, which then, in turn, requires up to 50% more PTC than its isotropic brother. The results for the Richardson error sensor (IS-001) are even less convincing. For threshold '-0010', one finds savings in cells for the Strouhal number, but an increase in cell counts for the lift coefficient. '-0004', on the other hand, can lead to savings for both quantities of interest. Here one can find savings of about 30% at level 5. Yet again, when looking at the PTC count, the results become much worse. At levels 4 and 5, the anisotropic simulations require more PTC than their isotropic counterpart. This is especially the case for the level 4 simulation, e.g., the anisotropic simulation of '-0010' requires about five times more PTC.

Figure 5.10 shows the anisotropic mesh. The isotropic counterparts can be found in Figure 5.3 and 5.8 respectively. Scanning the resulting figures shows that the y-direction meshes share the largest similarity with the isotropic ones. Thus, the error sensors indeed suggest mainly refinement in the y-direction, as predicted in the introduction to this section. While in the y-direction, the meshes

can be regarded as rather symmetric; this is clearly not the case for the x-direction. The anisotropic meshes are much more irregular than their isotropic counterparts. Looking at the simulations separately: The reference error sensor refines the front face of the cylinder in x-direction while the lateral parts of the cylinder are refined in the y-direction with a level 5 mesh. Even though there is x-refinement in the wake also, one can say that the refinement in the y-direction is significantly more prominent. One would expect that at least in one direction, the wake refinement zones have the same extent as their isotropic sibling. This is, however, not the case, in both principal directions, the wake is refined to a much lower degree. The exception here is the lateral level 5 refinement mentioned before. Additionally, one can observe that the 'refinement speckles' have vanished as well. The situation is exactly the opposite in the region in front of the immersed boundary. Here the mesh in y-direction extends forward up to the inlet face of the flow domain, something that does not happen for the isotropic simulation. The latter is also an observation that can be made for Richardson-type simulation. The situation is different for the wake, e.g., in case '*-001-0010', the wake extends in y-direction up to $x \approx 17$ for the isotropic simulation but up to $x \approx 25$ for the anisotropic one. The y-direction wake thus grows, as opposed to the shrinking as it was happening for the reference case. With respect to the near field wake and immersed boundary refinement, the anisotropy at the maximum mesh level is much lower for the Richardson simulations. Especially the wake directly behind the cylinder gets refined almost isotropically. The far-field wake is again dominated by y-refinement. Last but not least, as in the reference case, the mesh is rather irregular in x-direction.



(a) Level 5 - x-dir - REF-0004-ANISO

(b) Level 5 - y-dir - REF-0004-ANISO

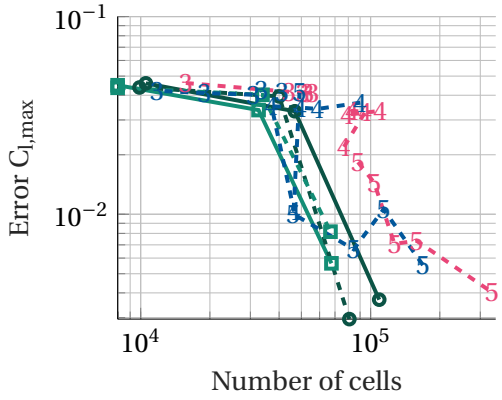
(c) Level 5 - x-dir - AS-001-0004

(d) Level 5 - y-dir - AS-001-0004

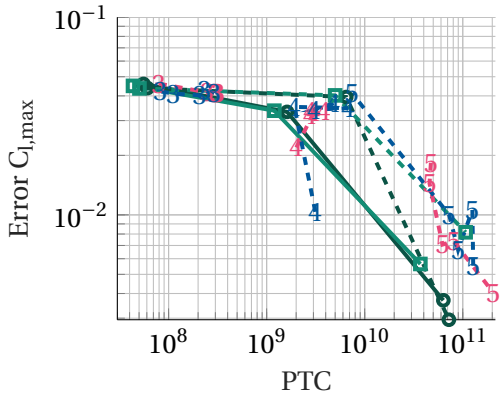
(e) Level 5 - x-dir - AS-001-0010

(f) Level 5 - y-dir - AS-001-0010

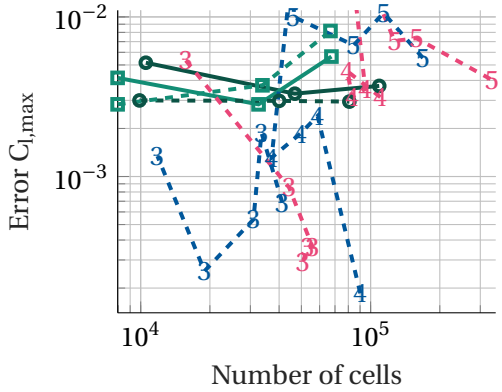
Figure 5.10: Mesh levels created by error sensors using anisotropic refinement. Mesh levels: Level 1, Level 2, Level 3, Level 4, Level 5.



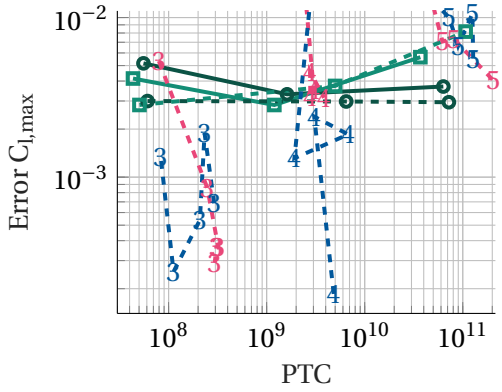
(a) w.r.t uniform level 5 mesh



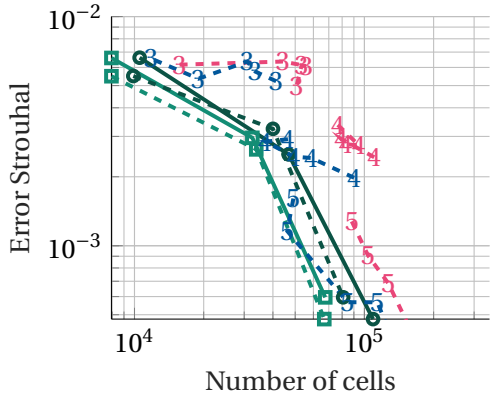
(b) w.r.t uniform level 5 mesh



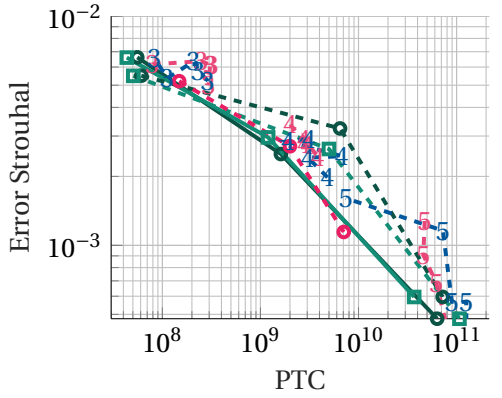
(c) w.r.t uniform mesh of corresponding level



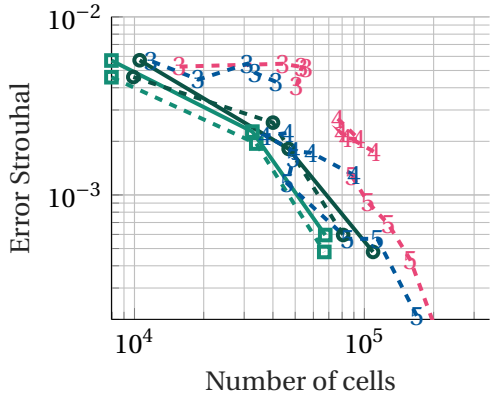
(d) w.r.t uniform mesh of corresponding level



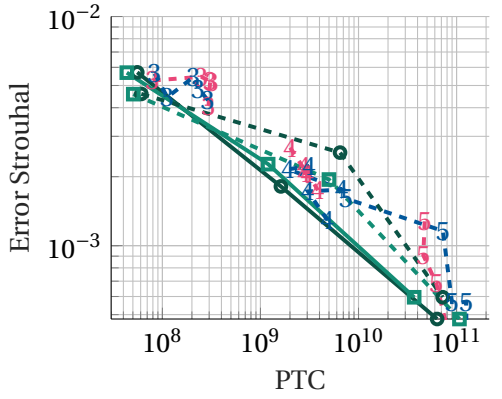
(e) w.r.t uniform level 5 mesh



(f) w.r.t uniform level 5 mesh



(g) w.r.t uniform mesh of corresponding level



(h) w.r.t uniform mesh of corresponding level

Figure 5.11: Errors in maximum lift coefficient and Strouhal number for anisotropic refinement of 2D-cylinder. REF (- - -), IS-001-0004 (—o—), IS-001-0010 (—□—), REF-ANISO (- - -), AS-001-0004 (- -o- -), AS-001-0010 (- -□- -).

5.5. Effect of adaption time step

The mesh adaption time step (τ_{AMR}) used for the algorithm has a potentially big impact on the mesh, and thus error that is going to be obtained. If the time step is too large, then the master and slave simulation might 'run apart' so that in the limit case, most of the domain will be refined, resulting in poor computational efficiency. On the flip side, if the time step is too small, then there might not be enough difference in both simulations to start adaption. A series of tests have been performed to investigate this effect. The standard Richardson relative-error from Equation 3.4 was used for this investigation. Two variables were varied, the time step with values of 0.01, 0.08, 0.25, and 0.75 seconds and two error threshold of 0.04 and 0.1 respectively. Table 5.7 summarizes all simulation parameters. The results of all simulations are shown in Figure 5.12, the corresponding mesh of level 5 is shown in Figure 5.13.

Table 5.7: Simulation settings for time step investigation of 2D-cylinder.

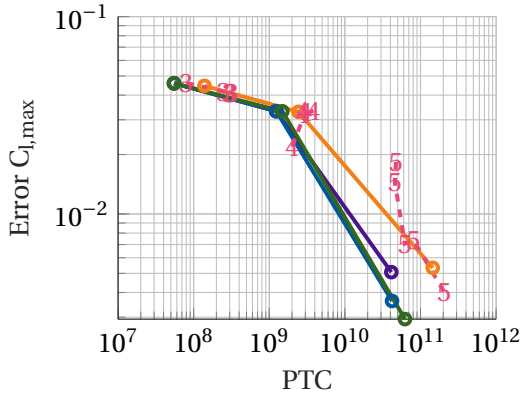
Legend	Identifier	Levels	ϵ_{AMR}	τ_{AMR}	$t_{\text{sim}}(s)$	Error sensor
	TIME-001-0004-01	3-4-5	0.04	0.01	70	Eq. 3.4 with $\|U\|$
	TIME-001-0004-08	3-4-5	0.04	0.08	70	Eq. 3.4 with $\|U\|$
	TIME-001-0004-25	3-4-5	0.04	0.25	70	Eq. 3.4 with $\|U\|$
	TIME-001-0004-75	3-4-5	0.04	0.75	70	Eq. 3.4 with $\|U\|$
	TIME-001-0010-01	3-4-5	0.10	0.01	70	Eq. 3.4 with $\|U\|$
	TIME-001-0010-08	3-4-5	0.10	0.08	70	Eq. 3.4 with $\|U\|$
	TIME-001-0010-25	3-4-5	0.10	0.25	70	Eq. 3.4 with $\|U\|$
	TIME-001-0010-75	3-4-5	0.10	0.75	70	Eq. 3.4 with $\|U\|$

For the simulations of $\epsilon_{\text{AMR}} = 0.04$, there is very little difference in error for both Strouhal number and lift coefficient at levels 3 and 4. The exception is simulation $\tau_{\text{AMR}} = 0.75$, which requires about double the PTC count for achieving the same error level. Nevertheless, even then, the performance of the reference error sensor can be matched or exceeded. For simulations $\tau_{\text{AMR}} = 0.01$ to $\tau_{\text{AMR}} = 0.25$, at level 5, the error in Strouhal number and lift coefficient is gradually decreasing, but at $\tau_{\text{AMR}} = 0.75$ the error suddenly spikes and even exceeds the one from $\tau_{\text{AMR}} = 0.01$ for the lift coefficient. The Pareto front of the reference error sensor gets also exceeded for the error in Strouhal number. The error spike is a peculiar behavior since a finer mesh should lead to a more accurate result. A careful inspection of the time series of the results showed that the maximum lift coefficient was constant across the shedding cycles. Thus poor post-processing can not be the reason for this behavior. One has to conclude that either advantageous error canceling takes place for the lower τ_{AMR} or that a more refined mesh can indeed lead to higher errors. In terms of error at level 5, the first three AMR time steps lead to savings of up to 5 times the PTC count for the lift coefficient and 3.5 times for the Strouhal number. The measurable differences in lift coefficient error and Strouhal number for the lowest two τ_{AMR} are also interesting considering the very similar meshes as shown, e.g., in Figure 5.13a and 5.13c. One could also argue that the results are already within some noise band. There is only a difference of one block between both simulations, but this already has a measurable influence on the obtained error.

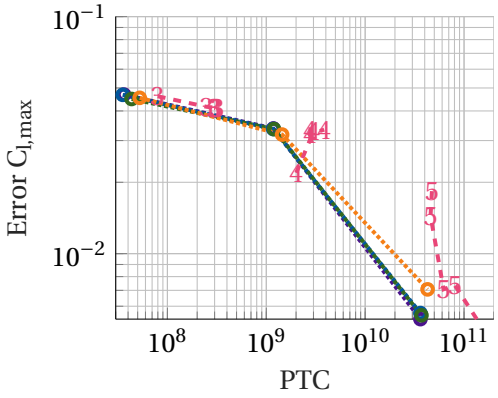
Considering the simulations of $\epsilon_{\text{AMR}} = 0.10$, again, very similar results are obtained at level 3 and 4 for the error in lift coefficient and Strouhal number. At level 5 $\tau_{\text{AMR}} = 0.75$ performs, as previously, erratically. While the error in the lift coefficient is again higher, now suddenly, the by far lowest error is obtained for the Strouhal number. The results for $\tau_{\text{AMR}} = 0.01$ to $\tau_{\text{AMR}} = 0.25$ are this time even closer to each other. The reasoning for this can be seen in the virtually identical meshes. Only simulation $\tau_{\text{AMR}} = 0.25$ has one extra refined block. In terms of computational efficiency savings of about four times can be achieved for the lift coefficient while savings of about two times in PTC

count can be achieved for the Strouhal number when comparing to the reference error sensor.

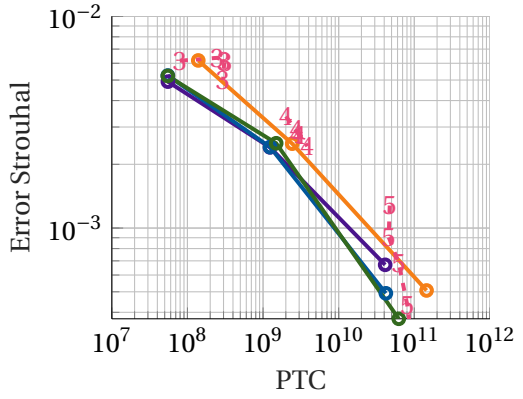
It is apparent that the influence of τ_{AMR} grows with a decreasing error threshold of ϵ_{AMR}. While at $\epsilon_{\mathrm{AMR}} = 0.10$ the results for $\tau_{\mathrm{AMR}} = 0.01$ to $\tau_{\mathrm{AMR}} = 0.25$ are close to each other, the differences grow with $\epsilon_{\mathrm{AMR}} = 0.04$. In general, these three τ_{AMR} perform very similar in terms of computational efficiency. $\tau_{\mathrm{AMR}} = 0.25$ can however lead to the lowest error in both metric for $\epsilon_{\mathrm{AMR}} = 0.04$. Based on the results, the decision was made to use $\tau_{\mathrm{AMR}} = 0.25$ for all (previous) investigations of the 2D cylinder problem. Obviously, this entire analysis was performed for one particular error sensor only, a full analysis for all examined error sensors is, unfortunately, not within the scope of this thesis. The main conclusion from this investigation should, however, be that the performance of the Equation 3.4 is very robust against any changes in the AMR time step. The investigated range spans almost two orders of magnitude, and only one case performed worse in the Strouhal metric in comparison to the reference error sensor. While the Richardson-type error sensor requires the extra parameter τ_{AMR}, one can conclude that it does not detrimentally affect the user-independence. At least for laminar flow, it must not be tuned very well to obtain an effective adaption.



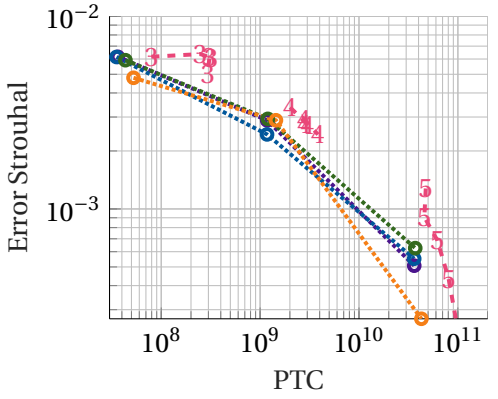
(a) absolute - $\epsilon_{AMR} = 0.04$



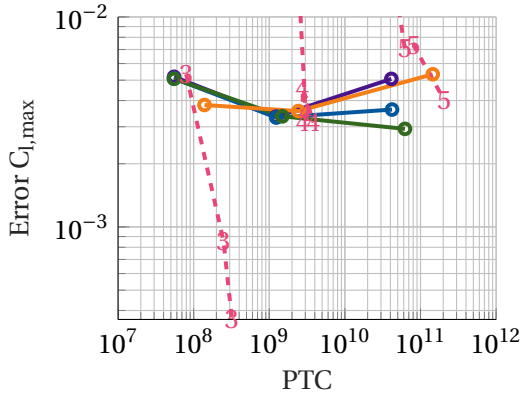
(b) absolute - $\epsilon_{AMR} = 0.1$



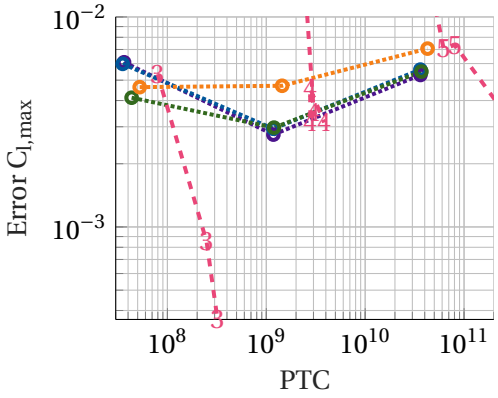
(c) absolute - $\epsilon_{AMR} = 0.04$



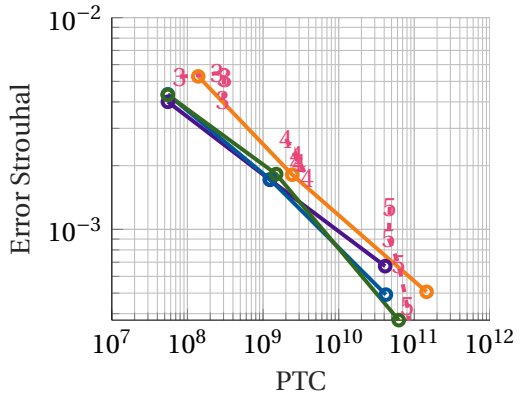
(d) absolute - $\epsilon_{AMR} = 0.1$



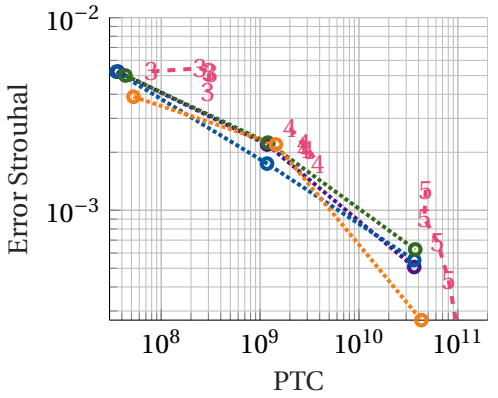
(e) relative - $\epsilon_{AMR} = 0.04$



(f) relative - $\epsilon_{AMR} = 0.1$

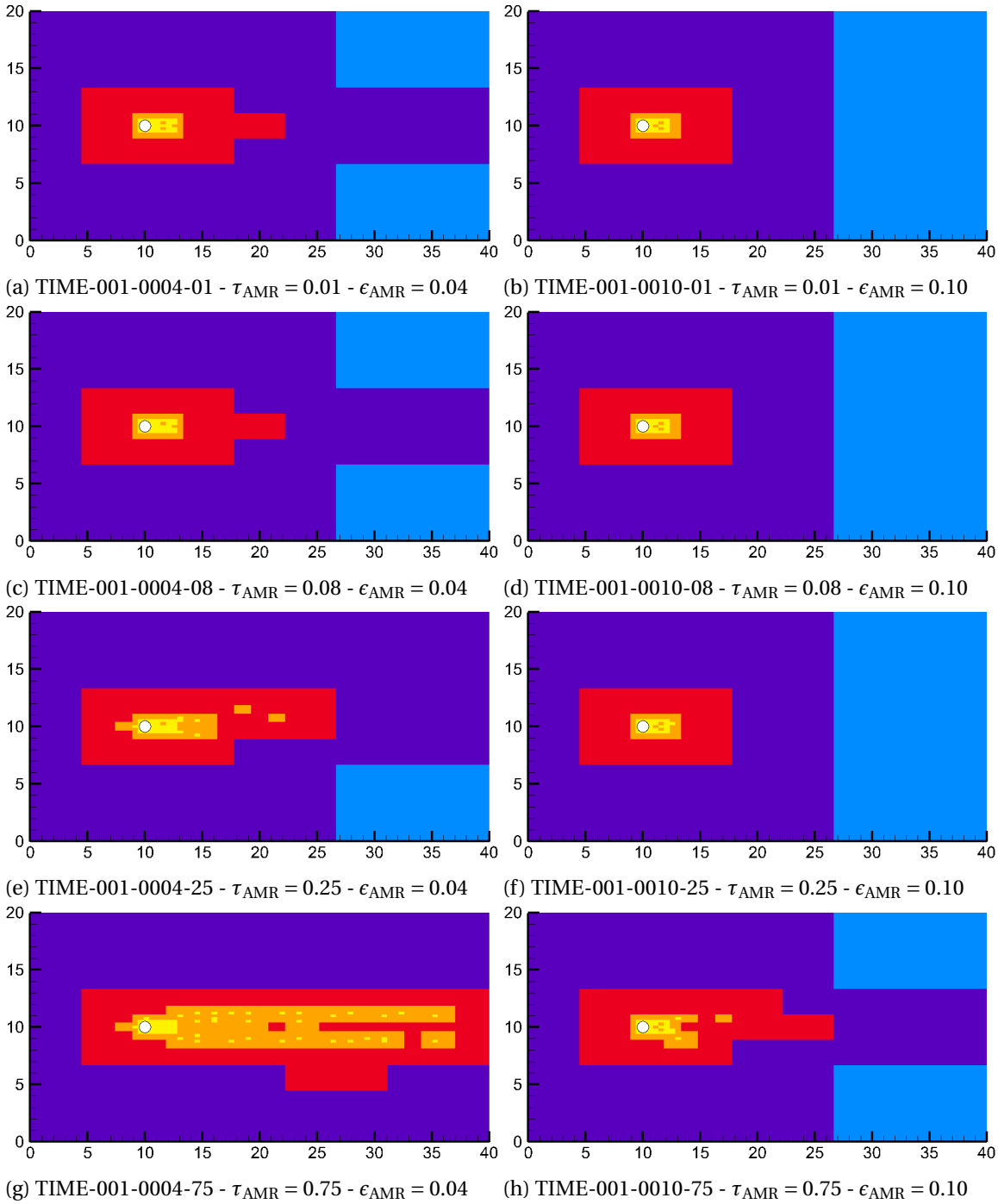


(g) relative - $\epsilon_{AMR} = 0.04$



(h) relative - $\epsilon_{AMR} = 0.1$

Figure 5.12: Errors in maximum lift coefficient and Strouhal number for simulations of τ_{AMR} investigation. TIME-001-0004-01 (—●—), TIME-001-0004-08 (—●—), TIME-001-0004-25 (—●—), TIME-001-0004-75 (—●—), TIME-001-0010-01 (···●···), TIME-001-0010-08 (···●···), TIME-001-0010-25 (···●···), TIME-001-0010-75 (···●···), REF (- - -).



(a) TIME-001-0004-01 - $\tau_{AMR} = 0.01$ - $\epsilon_{AMR} = 0.04$

(b) TIME-001-0010-01 - $\tau_{AMR} = 0.01$ - $\epsilon_{AMR} = 0.10$

(c) TIME-001-0004-08 - $\tau_{AMR} = 0.08$ - $\epsilon_{AMR} = 0.04$

(d) TIME-001-0010-08 - $\tau_{AMR} = 0.08$ - $\epsilon_{AMR} = 0.10$

(e) TIME-001-0004-25 - $\tau_{AMR} = 0.25$ - $\epsilon_{AMR} = 0.04$

(f) TIME-001-0010-25 - $\tau_{AMR} = 0.25$ - $\epsilon_{AMR} = 0.10$

(g) TIME-001-0004-75 - $\tau_{AMR} = 0.75$ - $\epsilon_{AMR} = 0.04$

(h) TIME-001-0010-75 - $\tau_{AMR} = 0.75$ - $\epsilon_{AMR} = 0.10$

Figure 5.13: Mesh created by simulations of τ_{AMR} investigation. Mesh levels: Level 1, Level 2, Level 3, Level 4, Level 5.

5.6. Overhead created by the master-slave approach

If the entire AMR routine is too expensive, then any advantages in speed up due to cell count savings will be negated. For the Richardson-type simulations, there are two potential sources for excessive computational cost, the need for a second simulation, and the routine itself. To investigate the former, the number of cells, the time steps, and average Poisson iterations for one time step have been recorded for the last shedding cycle. Table 5.8 summarizes these results. First, as one expects, the number of cells for the master simulation is nine times higher than for the slave simulation. However, every computed time step needs for the master simulation 193.3 Poisson iterations, whereas the slave simulation only requires 78.65. A difference in mesh level, thus, adds cells to not only the bill of computational cost but also results in a worse conditioned Poisson matrix. This highlights why the total computation cost should not only be measured in cell count but why rather a metric such as PTC can be advantageous. Using the PTC-metric, one obtains a total computational cost factor of 22.11 between master and slave. The extra 4.5% of overhead is measurable but can be regarded as more than acceptable especially when looking at the computational savings that AMR provides in the first place in comparison to a uniform simulation.

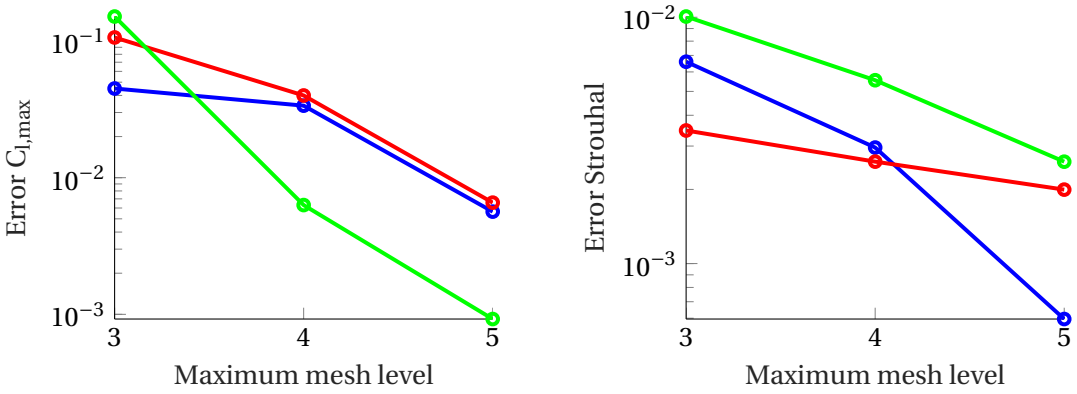
Table 5.8: Cells at last time step and overall PTC count for master and slave simulation of IS-001-0010 level 5.

	Time steps	Cells	avg. poisson iterations	PTC
Master	2852	67554	193.3	3.724E10
Slave	2852	7506	78.65	1.684E9
Fraction	1	9	2.46	22.11

Neither the number of cells nor a metric such as PTC take into account the overhead, which is created by the AMR criterion itself. In the case of an adjoint simulation, the overhead of the AMR criterion can be of the same order of magnitude as the primary problem [57]. This highlights that for a successful AMR adaption, not only an efficient mesh needs to be created, but the methodology used to obtain it in the first place needs to be efficient as well. Using INCA's internal profiler, the wall clock time of the AMR routine was recorded. Table 5.9 shows the measurements for one set of cases from Section 5.5 with τ_{AMR} of 0.01, 0.08, 0.25 and 0.75. Further, the average number of solver time steps between adaptions is given as well. Adaptions are defined here as: Every time the error sensor is used to query the results for possible mesh changes, including the latter. Before going into the analysis, the output in terms of wall-clock time is given as a percentage of the total computation time. The total computational time includes everything, the time required to write results to the hard drive, but also the time that was spent at MPI_WAIT commands. Further, the output comes only from the rank 0 process. Since the case was parallelized with multiple ranks and no perfect load balancing was achieved throughout the computation, the results should be taken merely as an indication of the order of magnitude of the cost that the AMR criterion requires. Since τ_{AMR} spans almost two orders of magnitude, a good overview is obtained. When the AMR criterion is called every 0.01 seconds, thus every 5th solver iteration, a computational cost of under 5% is obtained. This reduces to less than 0.4% at $\tau_{\text{AMR}} = 0.75$. Since, in this thesis, it is avoided to call the AMR criterion after every single time step, as often done in the classical Richardson-type adaptions, the computational overhead can be considered as almost negligible. Especially considering that in Section 5.5 and optimum τ_{AMR} of 0.25 seconds was found, which still produces less than 0.5% extra computational overhead. Considering that during the implementation of the adaption routine, not too much attention was paid in computational optimization, improvements in these should definitely still be possible, reducing the overhead even more. From the discussion so far, one can conclude that the significant part of the computation overhead of the Richardson-type AMR criteria is produced by the slave computation and not the error sensor computation and adaption itself.

Table 5.9: Runtime of AMR of Richardson-based adaption for 2D cylinder case.

Identifier	% of runtime	τ_{AMR} (s)	timesteps between call
TIME-001-0010-01	3.56	0.01	5
TIME-001-0010-08	0.91	0.08	39
TIME-001-0010-25	0.47	0.25	120
TIME-001-0010-75	0.36	0.75	359



(a) Lift coefficient

(b) Strouhal number

Figure 5.14: Convergence plot for IS-001-0010. Absolute error of master simulation (—o—), absolute error of slave simulation (—o—), difference between master and slave simulation (—o—).

Scientific simulations usually call for a grid convergence study. One could regard the extra coarse simulation, which is necessary for the Richardson approach, part of this. In such a case, there is no overhead, since a coarse simulation would have been computed anyway. Figure 5.14 shows two plots for the Strouhal number and the lift coefficient, respectively. This entails the error of the master and slave simulation with respect to the level 5 solution from the uniform grid as well as the difference in solution between the master and slave simulation. The hope is to deduct from the difference between the master and slave coefficients, a measure for the grid convergence of the entire problem. For the lift coefficient, this works quite well. Both, the lines for the true error of the master simulation and the estimate obtained from the master and slave computation agree closely for level 4 and 5. The results are worse for the error for the Strouhal number. Here close agreement can only be found for level 4. However, the trends are still matching, meaning that the actual and estimated error becomes smaller with increasing level. This means that the results of the master and slave simulation are converging towards each other. To conclude, this simple study shows that at least for this problem, one can estimate the convergence and overall error level using this approach. Last but not least, the peculiar results of the slave simulation should also be highlighted. The true error is lower than that of the master for the lift coefficient, even though the mesh is nine times smaller. Since the slave runs operate at a lower CFL number due to the time step synchronization, a temporal error was suspected for causing the results. However, repeating the simulation with a master CFL number of 0.5 led to virtually identical results. The reason for this behavior thus remains to be unknown and requires additional attention.

5.7. Conclusion

In general, this chapter provided encouraging results showing an effective reduction in the required computational resources. Already the curvature reference error sensor leads to significant performance improvements. Considering its trivial implementation, one could say that an easy AMR is still

better than no AMR at all. In isotropic refinement, the relative error in velocity magnitude proved to be by far the most successful one, leading to savings of up to 4 times in comparison to the reference error sensor. Additionally, it performed better than adaption based on absolute error in velocity magnitude, which can be considered the second most successful criterion investigated within this chapter. The error sensors based on the error transport equation were especially at the lower thresholds able to outperform the reference error sensor. However, they also sometimes lead to just equal or even worse computational efficiency. Considering that the error sensor from 'IS-001' performed significantly better for every metric and threshold, the success of the ETE-error sensors can only be described as modest. Very disappointing was the performance of simulations 'IS-007'. Combining the absolute error in the velocity magnitude with the mesh size does not seem to be a good idea. Very little attention was given to refinement around the immersed boundary, but instead, wake dominant meshes were created. The problem is that the output of the error sensors gets artificially decreased by the mesh size. This becomes erroneous when the error does not significantly decrease after adaption.

For this problem, the so-called PTC count metric was introduced with the motivation to provide a better estimate of the actual wall-clock time of the problem. The performance of the error sensors could sometimes be described as similar for cell and PTC savings. However, there were also simulations where significant discrepancies could be spotted, especially for anisotropic adaption. When referring to the mesh quality, one should, therefore, not only talk about the error reduction in the flow field or a quantity of interest but also the conditioning of the problem's Poisson matrix. The effect should be further investigated, for example, by making some by-hand adjustments to the AMR meshes to increase homogeneity and thus remove refinement speckles. Additionally, using the PTC count revealed another aspect of an error sensor and flag function combination. Ideally, there should be some trade-off between the gain in accuracy by introducing a finer mesh level, and the increase of computational cost, e.g., due to the need for more time-steps. Instead of achieving the error reduction be a finer level, the algorithm might also decide to reduce the error by adding more cells of the current maximum level. Another aspect of this is the leveling-off of the error reduction with decreasing error threshold. E.g., from the Pareto front of the reference error sensor, one can see that at some point, the effect of reducing the error threshold saturated and subsequent improvements can only be obtained by increasing the maximum mesh level. This is yet another aspect which needs to be properly addressed to achieve user-independent mesh adaption.

Due to the lateral dominance of flow structures in the wake of the cylinder, anisotropic adaption seems to be an effective method of decreasing the computational expenditure even further. The recomputation of the reference error sensor revealed that, indeed, savings up to 50% are possible. Unfortunately, the Pareto fronts were not as monotonic in comparison to the isotropic case. Unfortunately, when also considering the PTC count, then the gains were largely negated, and performance was often only equal or worse in comparison to the isotropic reference indicator. The 'IS-001' simulations were also recomputed with anisotropic adaption. Here things looked even worse. In terms of cell count, some simulations performed worse and some better than in comparison to their isotropic counterpart. Again when using the PTC count, the performance deteriorated significantly and partially lead to inferior performance, sometimes even worse than the isotropic reference error sensor. The results can, therefore, be regarded as encouraging with regard to a reduction in cell count, but the translation into actual wall-clock time savings does not happen.

Another encouraging conclusion from this investigation includes that all error sensors were resilient to the initial under-refinement of the problem. The same observation was made for compressible Mach 3 shock flow problem in the previous chapter. The error sensors were able to successfully traverse through the level 2 mesh even though at this point, vortex shedding was present only heavily damped. In this regard, the investigated error sensors are suitable for a user-independent AMR ap-

proach were the initial mesh should be considered to be very rough e.g., when an immersed boundary is resolved with only a couple of cells.

The timestep used between adaption has a small but measurable impact on the performance, at least for the tested error sensor ('IS-001'). However, one should also keep in mind that the results are reasonably close to each other, considering that a time step sweep of almost two orders of magnitudes has been performed. The error sensor from Equation 3.4 can, therefore, be regarded as rather insensitive to the adaption time step, at least for laminar flow. Next to this, the overhead of the Richardson approach (and also the ETE approach) can be regarded as insignificant, the computation of the slave problem merely required 4.5% of the PTC count of the master simulation. Further, the overhead of the error sensor computation itself, including mesh adaption, required interpolation, etc. ranged between 0.36 and 3.56% depending on the adaption frequency. All in all the used approach can be described as very lightweight.

This case also provided valuable user-feedback on the approach taken with the flag function. The error is usually equidistributed in space when refinement and coarsening are used and merely pushed below a certain threshold when only refinement is considered, as was done in this thesis. While at first glance, this approach seems appealing since one can control the resulting error a priori, assuming that one has a 'perfect' error sensor, it comes with the problem that the final mesh size is not controlled. In Section 5.3, the aim was to select three error thresholds for every error sensor so that a mesh size of about 50,00 to 150,000 was spanned. In practice, this required multiple runs until suitable thresholds were found. Runs either had too few or too many cells. Especially, the latter creates user-problems. *Usually* an engineer or scientists have only a limited amount of computational resources available, e.g., on his personal workstation. Alternatively, he might have to select a priori how much hardware he wants to allocate when, for example, using a large computational cluster with a resource manager. Further, either wall-clock time constraints are imposed, or simply a deadline has to be met. Submitting runs to then eventually notice that they will never finish in the desired time creates significant overhead on the user side. The same holds when a run ends shortly after submission because the error threshold was chosen too large, and one could have easily obtained an even more accurate result. This becomes even more annoying when one has to wait days or perhaps even weeks until one has cleared a cluster queue to then essentially throw away the waiting time. In a more development based environment, one also often encounters the practice of starting multiple computations, which will then run over the weekend and are expected to finish on Monday. These examples highlight user-experiences that should not be encountered in any user-independent AMR routine. Since it is impossible, at least currently, to know the required computational resources and resulting error before a computation is started, the author is of the opinion that the best approach is to consider the flag function solely from a computational cost point of view. Meaning one defines a specific mesh size or PTC count at the onset of the simulation. It is not erroneous to assume that a user will be able to sufficiently estimate the wall-clock time of his case from either of these two numbers, especially when he is familiar with the hardware infrastructure at his disposal. One could argue that an inaccurate completed simulation is as useless as an accurate unfinished simulation. Perhaps one could even argue that the latter is even better since it can be continued. However, the author thinks that in practice, the engineer or scientist is resource-limited. Either because of not having enough hardware at his disposal or because the problem is still too advanced for the current hardware available on the market. Therefore a complete simulation is *as good as it gets*, and the engineer just has to work with the results even when they are not very accurate. An approach using a target mesh size requires coarsening. This is necessary since, in such a setting, one does not define an error threshold. The flag function can refine cells until the allocated cell count has been reached. For problems that evolve in time, for example, with a developing wake, as encountered in this 2D cylinder case, the approach would lead to excessive over-refinement at

the onset of the simulation. Coarsening is required to redistribute the cells throughout the domain when areas in need of refinement appear after a particular simulated time. Alternatively, one could also use the approach taken in this thesis to refine only X% of cells during every adaption cycle to limit the rate of mesh growth. During the previous arguments, it was indirectly implied that a perfect error sensor exists that can accurately predict the error within the domain. Since such a sensor does not exist yet, the situation becomes even worse. At the beginning of the simulation, one has to choose a certain threshold, but without a direct and reliable relationship between actual error and error sensor, it is challenging even to guess the numerical magnitude.

6

Flow over periodic hills at Re=10595

The periodic hill problem is a classic test case for examining the accuracy of turbulence models and RANS codes in particular. Notably, the correct prediction of the recirculation bubble poses majors challenges on the turbulence modeling. The two test problems that have been considered so far had distinct flow features such as shocks and vortex streets but did not contain any turbulence. It is reasonable to assume that the chaotic and unpredictable nature of turbulence might pose a significant challenge to the error sensors and flag function. Since a substantial part of industrial flow problems is inherently turbulent, this test case is crucial on establishing whether user-independent mesh adaption can be achieved with the, in this thesis presented, error sensors and flag functions. To limit the scope, only the reference error sensor from Equation 3.15 and the best performing Richardson-type error sensor from Equation 3.4 are investigated. Further, no analysis is performed on anisotropic refinement.

This chapter starts by describing the case setup. Subsequently, a series of tests are performed to establish how long the error sensor needs to be statistically averaged to be suitable as the driver for adaption in a turbulent flow. An extra strategy is presented to make the error sensor more robust against outliers. Next, the novel and reference error sensor are benchmarked against each other to establish which one can create the best mesh for this particular test case. Last but not least the error reduction and mesh growth during adaption is studied in more detail.

6.1. Case setup

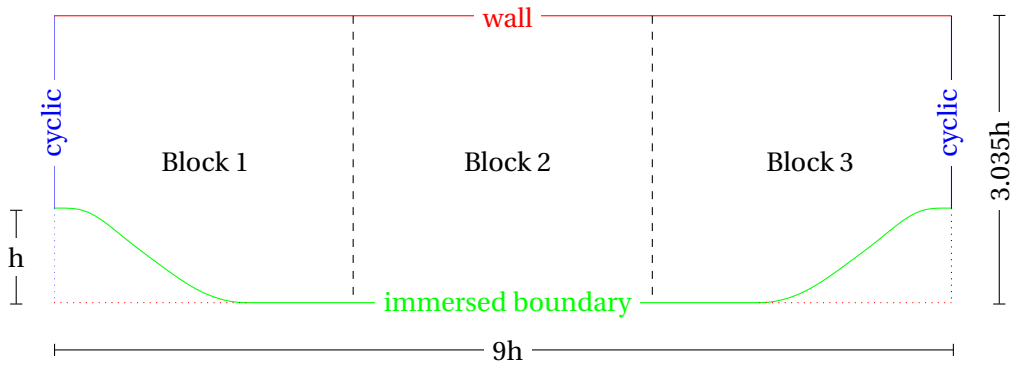
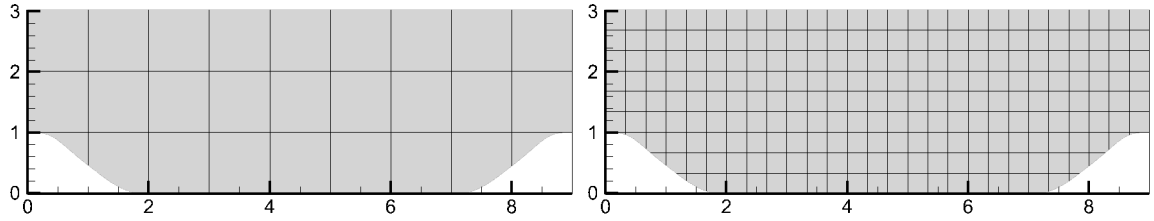


Figure 6.1: Boundary conditions and geometry for the flow over periodic hills at Re=10595. Spanwise extent is 4.5h. Periodic boundary conditions are applied in spanwise (front and rear face). Mathematical representation of immersed boundary is given in Appendix C.

The geometry and boundary conditions for this flow problem are given in Figure 6.1. A series of periodic hills are placed in a channel with the dimensions of 9 hill heights (h) long, 3.035h high, and 4.5h deep. To reduce the size of the problem, the typical approach of using cyclic boundary conditions at the inflow and outflow faces to the left and right of the domain was employed. The flow leaving the boundary on the right is used as the new inlet condition ad the left of the domain. A pressure gradient in x-direction was added to the momentum equation to drive the flow. To satisfy the correct mass flow and thus the bulk velocity of this problem, the pressure gradient was controlled by a PI-controller, which was evaluated at every time step. The controller was tuned in such a way that the fluctuations between target bulk velocity and actual bulk velocity usually stayed well below 0.1%. For the discussion of this chapter, only the statistical averaged x and y-velocity were considered. For these, the error due to the fluctuations averages out with statistical sampling and therefore, should not falsify any results. This is in contrast to the Reynolds stresses whose accuracy would be negatively affected by control errors of the PI-controller. The top and bottom of the channel were equipped with a wall boundary condition, whereas the front and rear face of the domain were connected with a periodic boundary condition. The hill geometry was modeled again with the previously used immersed boundary method. Appendix C gives the exact mathematical representation that has been used to model the hills. Data acquisition for statistical averaging started after 23 flow-through times and lasted 55 flow-through times long, which are the same values used in Fröhlich *et al.* [15]. Sampling took place every 15 solver iterations, and the results were homogenized in z-direction so that one 2D field was obtained. Krank *et al.* [3] and Gómez [77] have shown that a total of 78 flow-through times is not sufficient to obtain statistically converged solutions. Unfortunately, there were not enough computational resources to take these findings into account, and the used values for data acquisition stemming from Fröhlich *et al.* [15] represented the best compromise between accuracy and computational effort.

Figure 6.2 shows the starting mesh that has been used for all AMR runs of this test case. All simulations were started with three initial blocks, each block having 9x9x45 and 3x3x15 cells for the fine and coarse simulation, respectively. Three blocks were chosen so that every block was almost square in the x and y-direction. This meant that also all subsequently created blocks had this very same shape.

Again a reference solution was computed with INCA to establish a baseline that all adapted meshes have to compare to so that their computational efficiency can be judged. Based upon the starting mesh, a maximum refinement level of 3 was considered to be the sweet spot. A level 2 mesh would have resulted in only two different cell sizes, and anything over level 3 is too expensive to compute.



(a) Coarse simulation, 3x3x15 in 3 blocks (b) Fine simulation, 9x9x45 in 3 blocks

Figure 6.2: Starting mesh of flow over periodic hills at Re=10595. 15 cells in z-direction for coarse simulation. 45 cells in z-direction for fine simulation.

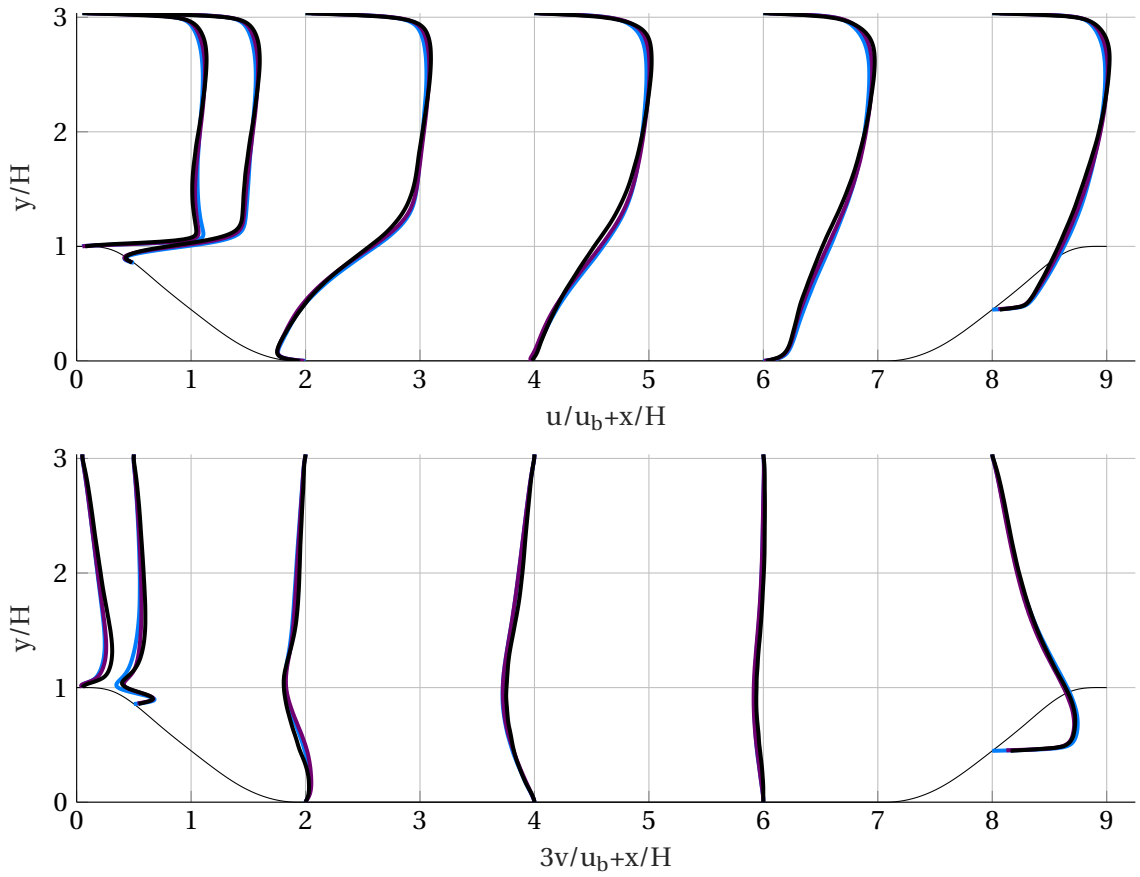


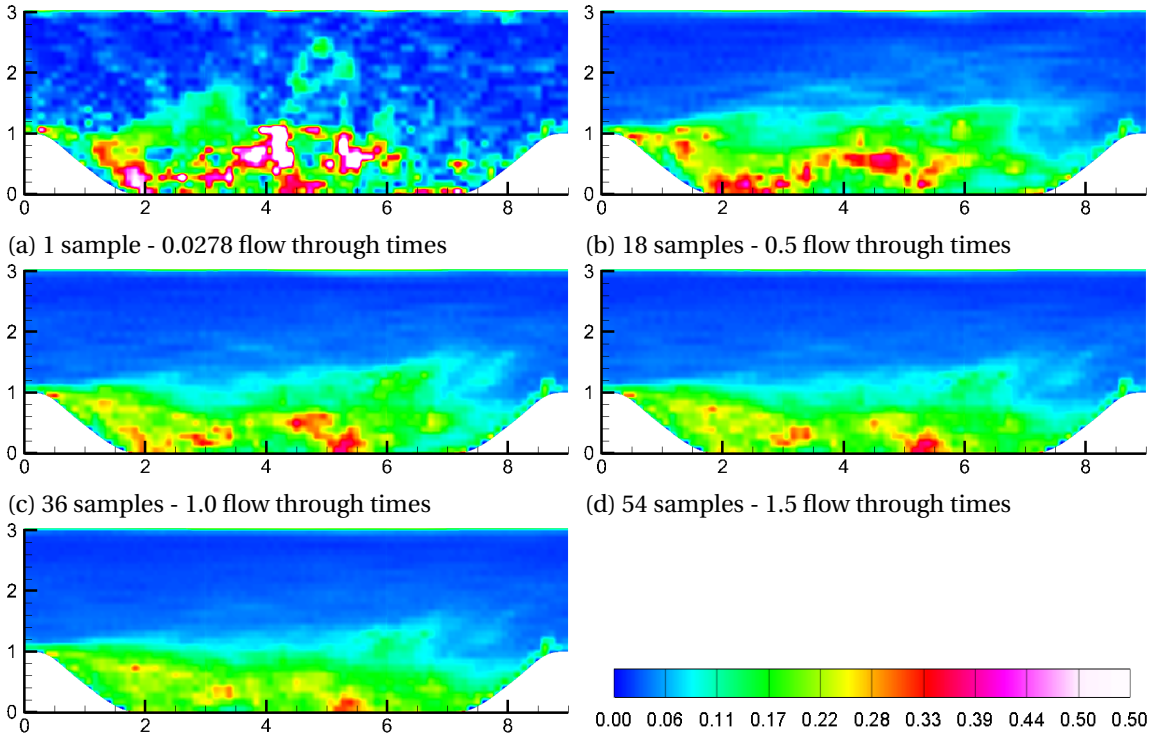
Figure 6.3: Results of uniform level 3 simulation in comparison to solutions from literature. INCA LVL3 (——), KKW (——) [3], Breuer LESOCC (——) [4].

To further reduce the complexity and necessary computational resources, the z-direction was only resolved with 45 cells in the master and 15 in the slave simulation. Adaption commenced only in x and y-direction, whereas the cell count in z-direction always stayed constant. This led to a final mesh size of 729x243x45, with a grand total of 7,971,615 cells. Figure 6.3 shows the results of this simulation in conjunction with two reference solutions from the literature. The results of the homogenized y-velocity can be described as 'line converged' with almost no differences visible across all three simulations. As expected, more differences can be spotted for the homogenized u-velocity. In the region $y/h > 2$, the velocity is consistently over-predicted in comparison to reference literature, while for $y/h < 2$ precisely the opposite statement can be made. Close to the top and bottom wall, all three results start to reach 'line convergence' again. For this thesis, the results can be considered to be acceptable. They resemble the original results close enough, so that statements

about the effectiveness of the different error sensors and flag functions can be made.

6.2. Required averaging period of error sensor

Using instantaneous error sensor outputs might lead to erroneous mesh adaption for fully turbulent flow. The inherent nature of turbulence is creating random flow structures that can either misalign during the time τ_{AMR} or cannot be resolved on the coarse mesh but are present on the fine one. 'Chaotic' error sensor output could be the result, which in turn might lead to the adaption of large or unimportant parts of the domain. To shed more light into this issue, the periodic hill simulation has been performed on a uniform level 2 mesh with the error sensor activated (81x243x45 cells). No adaption was performed, but the error sensor was recorded and averaged during the simulation, as explained in Section 3.4. Two different error sensors were considered, the so-far successful relative error from Equation 3.4 and, of course, the reference error sensor from Equation 3.15. Figure 6.4 shows five contour plots of the relative error averaged for different flow-through times, starting from an instantaneous reading. Averaging was stopped at 2 flow-through times since this amounts to already 18 seconds of simulated time. Even though it is desirable to have a perfectly averaged error sensor, it is evident that numerous adaption cycles have to be performed to develop the mesh fully. Very long required averaging times are thus clearly not practical.



(a) 1 sample - 0.0278 flow through times

(b) 18 samples - 0.5 flow through times

(c) 36 samples - 1.0 flow through times

(d) 54 samples - 1.5 flow through times

(e) 72 samples - 2.0 flow through times

(f) Error legend

Figure 6.4: Error sensor output of Equation 3.4 (relative error) for $\tau_{\mathrm{AMR}} = 0.25$ up to 2 flow through times.

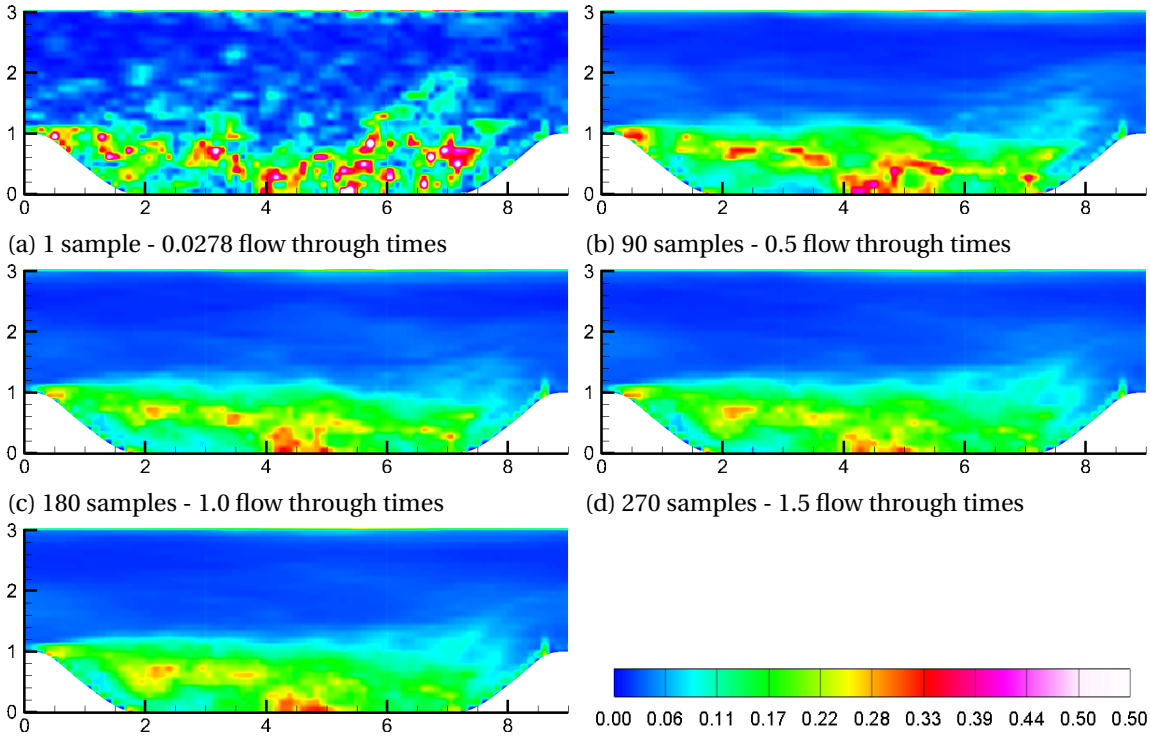
For the one instantaneous sample, the error sensor suggests refinement in sensible areas for the periodic hill problem, but it also becomes clear that random error sensor structures are formed. They manifest themselves as disconnected error blobs scattered throughout the entire domain. Already at 0.5 flow-through times, or 18 samples, it is visible that the flow above the hills ($y/h > 1$) has been significantly smoothed out, and structures in the error sensors can only be seen very faintly. With an increasing number of samples, the error distribution in this area becomes more and more uniform. Diminishing returns reveal themselves for this area when averaging past one flow-through time. In the area between the hill crests ($y/h < 1$), the effects of averaging the error sensor

are much more significant. Again the error sensor structures smooth out significantly between one sample and one flow-through time. This time, however, additional sampling noticeably adds to the quality of the average. It is apparent that two flow-through times are not enough to obtain an entirely smooth distribution, and distinct structures remain visible. In a smoother error sensor field, large error peaks vanish. For example, the maximum error that can be found in the instantaneous field is about 0.91, whereas at one flow-through time, the maximum error has reduced to about 0.41. Thus, the averaging period has, just as τ_{AMR} (shown in Section 5.5), an effect on the maximum error magnitude. Therefore, the same threshold can lead to different adaptions dependent on the chosen averaging time for the error sensor. The requirement of defining a problem-dependent averaging time for the error sensor does certainly not contribute positively to the user-independence of the methodology. The direct connection of error magnitude with the averaging time complicates this matter further. Another example of why adaption solely based on a target mesh count might be favorable.

The second variable that might influence the error sensor field is the number of samples taken during one flow-through time. Therefore, a second simulation has been performed in which the same averaging time has been used, but τ_{AMR} was reduced to 0.05 seconds. This lead to a fivefold increase in the number of samples. The results are shown in Figure 6.5. The instantaneous error field does not match the one from Figure 6.4 since a new computation has been performed. Differences between τ_{AMR} 0.25 and 0.05 can indeed be spotted. The flow in the region $(y/h > 1)$ is noticeably smoother. Whereas for τ_{AMR} 0.25 still cubical patterns can be spotted at 0.5 flow-through times, they have already vanished at that time instance for τ_{AMR} 0.05. In the region of $(y/h < 1)$, there seems to be a more substantial reduction in error sensor magnitude for τ_{AMR} 0.05 when considering the step from 0.5 to 1 FTT. For the following flow-through times, the changes in the error field due to averaging are approximately on the same level. Based on these results, it has been decided to use $\tau_{\mathrm{AMR}} = 0.05$ and a sample time of 1 FTT. Comparing the results to the ones from the laminar flow case, one can see that even though the problem is very different, τ_{AMR} still remains to be in the same ballpark. This underlines the previous findings, namely that the influence of τ_{AMR} is measurable but at the same time of only moderate importance to the entire adaption routine.

The requirement of determining a problem-dependent averaging period is not ideal, especially concerning user-independence. However, this problem is not only inherent to the Richardson error sensor but also to the regular feature-based adaption. Feature-based error sensors are computed directly from the flow field by simple relations. Thus, when their input stems from instantaneous flow data, they will also contain random error sensor structures. A third simulation has been performed to highlight this point, but this time with the curvature reference error sensor. The output is shown in Figure 6.6. The 'chaotic nature' in the error sensor is clearly visible when only one sample has been taken. Just as for the Richardson-type error sensor, averaging to 0.5 FTT smooths out the region $(y/h > 1)$ considerably. Nevertheless, checkerboard patterns can still be seen even at 2 flow-through times. The region $(y/h < 1)$ profits from averaging times above 1 FTT as well. One can conclude that the Richardson-type error sensor requires less care in choosing the right averaging strategy than the reference one and therefore comes with an actual improvement of user-independence. Nevertheless, the results should be taken with a grain of salt, since they might be the result of statistical effects.

On the last note: The current strategy of obtaining an error sensor output for turbulent flow is only applicable for statistically steady problems. How to deal with turbulence in statistically unsteady problems is a question that the current work can unfortunately not answer.



(a) 1 sample - 0.0278 flow through times

(b) 90 samples - 0.5 flow through times

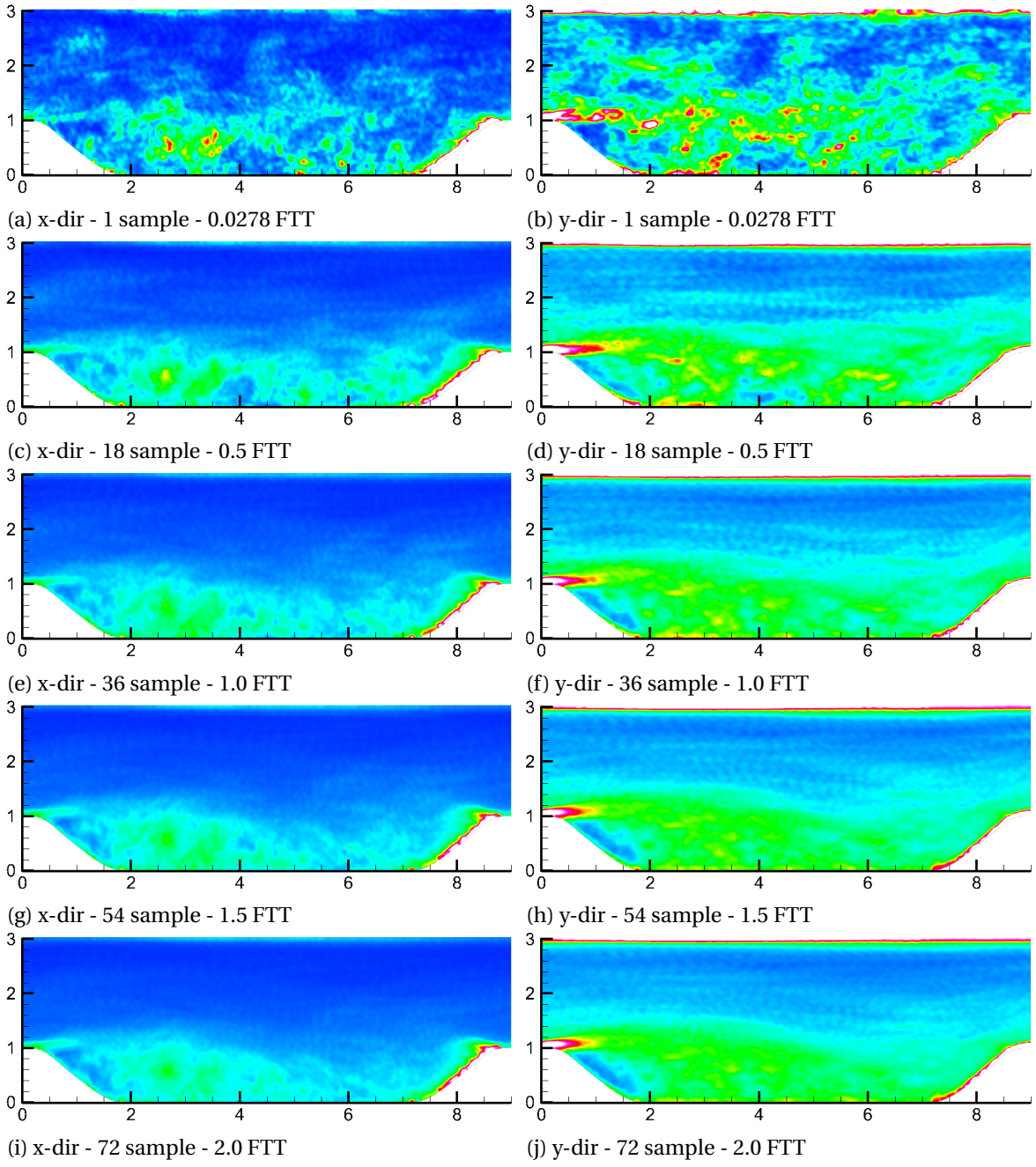
(c) 180 samples - 1.0 flow through times

(d) 270 samples - 1.5 flow through times

(e) 360 samples - 2.0 flow through times

(f) Error legend

Figure 6.5: Error sensor output of Equation 3.4 (relative error) for $\tau_{\text{AMR}} = 0.05$ up to 2 flow through times.



(a) x-dir - 1 sample - 0.0278 FTT

(b) y-dir - 1 sample - 0.0278 FTT

(c) x-dir - 18 sample - 0.5 FTT

(d) y-dir - 18 sample - 0.5 FTT

(e) x-dir - 36 sample - 1.0 FTT

(f) y-dir - 36 sample - 1.0 FTT

(g) x-dir - 54 sample - 1.5 FTT

(h) y-dir - 54 sample - 1.5 FTT

(i) x-dir - 72 sample - 2.0 FTT

(j) y-dir - 72 sample - 2.0 FTT

(k) Error legend

Figure 6.6: Error sensor output of Equation 3.15 (reference error sensor) for $\tau_{\text{AMR}} = 0.25$ up to 2 flow through times.

6.3. Isotropic refinement

Based upon the results of Chapter 5 and to limit the scope of this thesis, it has been decided to restrict the investigation to only the Richardson-type error sensor of Equation 3.4 and of course the curvature reference error sensor from Equation 3.15. Before starting with the results of the actual adaption, first, the error sensor plots from Figure 6.4 to 6.6 are analyzed in more detail. Figure 6.4 shows that the relative velocity error is the largest for $(y/h < 1)$, whereas the error is small for $(y/h > 1)$. From an adaption point of view, this indeed seems to be reasonable, since the prediction of the recirculation bubble is the main difficulty within this problem. Nevertheless, the result can also be regarded as somewhat surprising since also the channel region $(y/h > 1)$ contains significant turbulence, which in this case is not leading to a large error between the master and slave simulation. The instantaneous error sensor output shows error blobs in this region, which are, however, of significantly lower magnitude in comparison to the region between the hill crests $(y/h < 1)$. The relative error sensor also lights up the top wall of the channel. The error has a smaller magnitude than in the region $(y/h < 1)$, and the natural question arises on whether the adaption of the top wall might be shadowed by the higher error magnitudes in the region $(y/h < 1)$. The reference error sensor is shown in Figure 6.6 and consists of one output in x and y-direction. In general, one can say that the magnitude is the lowest for the x and the highest for the y-direction except for the windward side of the hill flank, where both perform fairly similar. The y-direction locates a high error at the of the inflow boundary. Further, the upper wall is also suggested for refinement. Finally, in both, x and y-direction the recirculation bubble can be seen, but it is equipped with a much lower error magnitude than the other discussed areas.

6.3.1. Additional filtering of error sensor

During initial simulation trials, it was noticed that many blocks only contained a limited amount of bad cells, usually in the range below 1%, when flagged for adaption. These threshold violations can either be genuine or the result of the limited sample period (1FFT) of the error sensor. Therefore, an investigation was started to see if it can be advantageous to exclude blocks with less than 1% of bad cells. The run list of this investigation is given in Table 6.1 and the respective results in Figure 6.7 and 6.8.

Table 6.1: Settings of simulations for periodic hill with relative error sensor (Equation 3.4).$\tau_{\mathrm{AMR}} = 0.05$, 23 FTT settling, 55 FTT statistical sampling, 1 FTT AMR sampling.

Legend	Identifier	Cells	ϵ_{AMR}	Flag function	Flag threshold
	PHILL-01-35	1002375	0.35	Algo. 3	1%
	PHILL-01-40	535815	0.40	Algo. 3	1%
	PHILL-01-45	331695	0.45	Algo. 3	1%
	PHILL-02-40	1002375	0.40	Algo. 2	-
	PHILL-02-45	739935	0.45	Algo. 2	-
	PHILL-02-55	244215	0.55	Algo. 2	-

Figure 6.7 contains 6 plots, the first two contain the in z-direction homogenized x- and y-velocities at different x-stations. Subsequently, the integrated error of the line plots is graphed against the usual metrics of cell and PTC count. Integrating the line plots gives an approximate global error with respect to the uniform level 3 simulation. It is approximate, since only the standard discrete locations reported in literature are used $(x/h = 0.05, 0.5, 2, 4, 6, 8)$ [15]. The results show that for the x and y-velocity and both performance metrics, the error is lower for simulations '01-45' and '01-40' in comparison to the simulations without extra filtering. The pattern is only broken by '01-35', which has a significantly higher error than its counterpart. The natural question arises why '01-35' performs so badly. When carefully inspecting the u-velocity line plot of Figure 6.7, one can see that

the error for '01-35' is larger close to the hillcrest at the domain entrance. This error continues to spread into the recirculation region. '02-40' does not seem to face this problem since the level 2 refinement zone in the recirculation region extends up to the second hillcrest, as can be seen in Figure 6.8a. Even though '02-40 shows superior performance, the overall results still hint at the positive effects of mild filtering. When inspecting Figure 6.8, it is, however, difficult to understand why these savings in computational efficiency occur, since the meshes between both strategies look fairly similar. Comparing the mesh of '01-40' and '02-45', it appears that the most important feature is the level 2 mesh refinement region, which is identical between both cases. The additional level 3 refinement does not lead to a better mesh and even increases the error.

'Solution artifacts' can be spotted in the statistically averaged line plots in Figure 6.7 and 6.10. Their location coincides with block boundaries of different refinement levels. This can be nicely seen for, e.g., the line plot of '01-35' at $x/H = 2, y/H = 2$, and the respective mesh shown in Figure 6.8. The root cause analysis is still in progress while completing this thesis. The interpolation method presented in Hickel [78] was used for the interpolation of the block solutions at their respective boundary in case of non-matching mesh levels. In general, this is clearly an undesirable behavior. This is an example that shows that in *practical* applications, AMR cannot only increase but also negatively affect the accuracy of the solution. The same problem might explain the findings of, e.g., Chapter 5, which has shown a larger error for a finer but irregular mesh.

One could argue that the percentage thresholds for bad blocks should be further increased to push the error threshold ϵ_{AMR} to the magnitudes of the previous two benchmark problems. The counter-argument is proposed that even more filtering might negatively influence the solution, when an area of interest is merely clipping a block, e.g., a small part of an immersed boundary, or a wall-boundary condition. A high threshold of, for example, 20% can make an adaption of these blocks impossible. Nevertheless, some adaption would still happen due to the balance criterion.

A crucial observation is that the line plots of Figure 6.7 show a significant error at the top wall. As already depicted in Figure 6.4, the Richardson error sensor outputs only a small error magnitude close to the top wall, which is completely dwarfed by the high error in the recirculation bubble. Therefore, no refinement took place close to the wall leading to an insufficiently resolved boundary layer. The over-prediction of the x-velocity close to the top wall leads to an under-prediction in the region $y/h < 1.5$. When recalling Equation 3.4, a constant of 0.1 is found within the equation to prevent singularities in regions with low velocity. One could argue that this constant is the origin of the poor refinement close to the wall. However, tests showed that varying this threshold did not lead to any meaningful change in mesh at the top wall. One must conclude that the Richardson-type error sensor performs well in capturing wakes in laminar (Chapter 5) and turbulent flow, but seems to have difficulties with under-resolved boundary layers in regions with little streamwise pressure gradient.

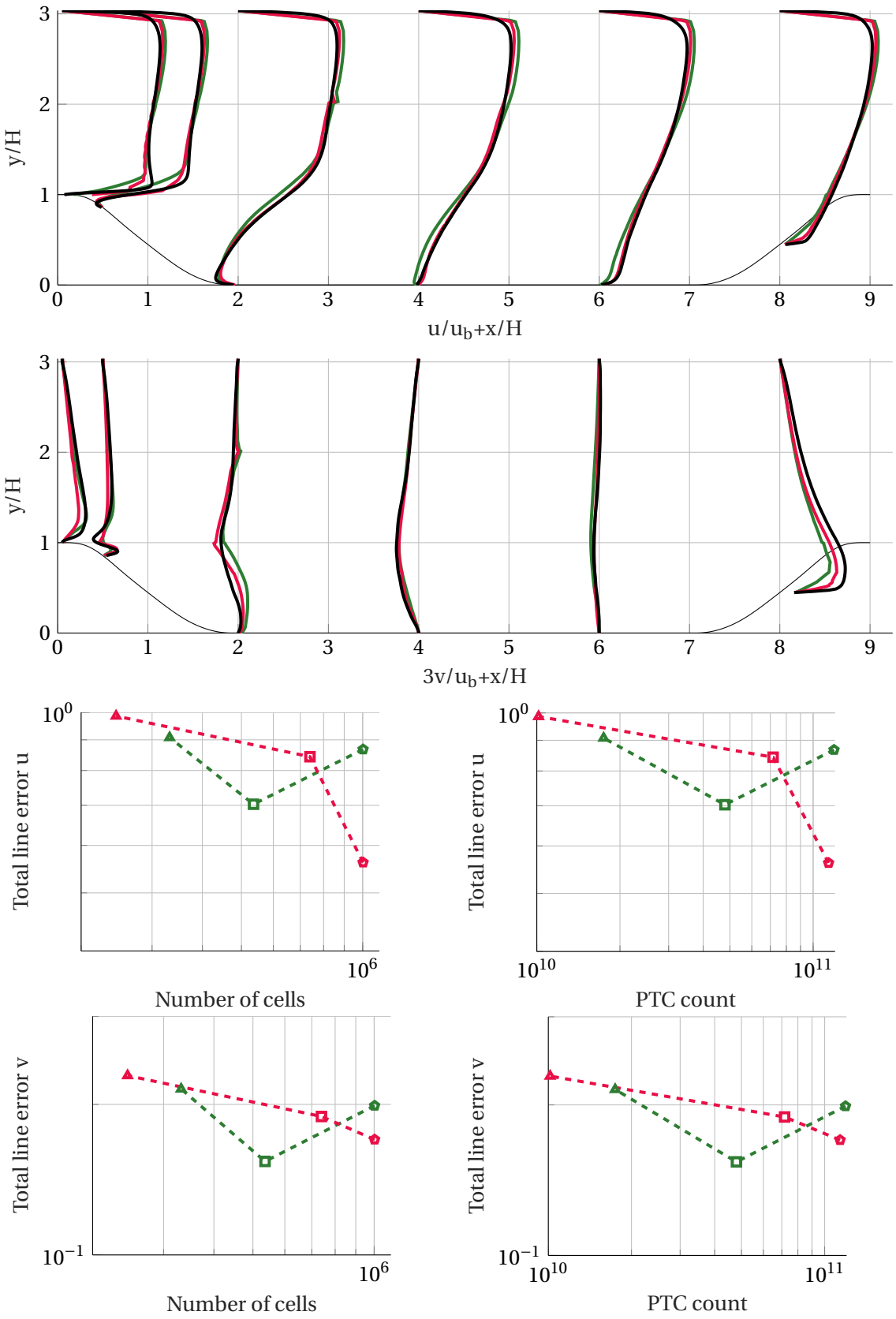
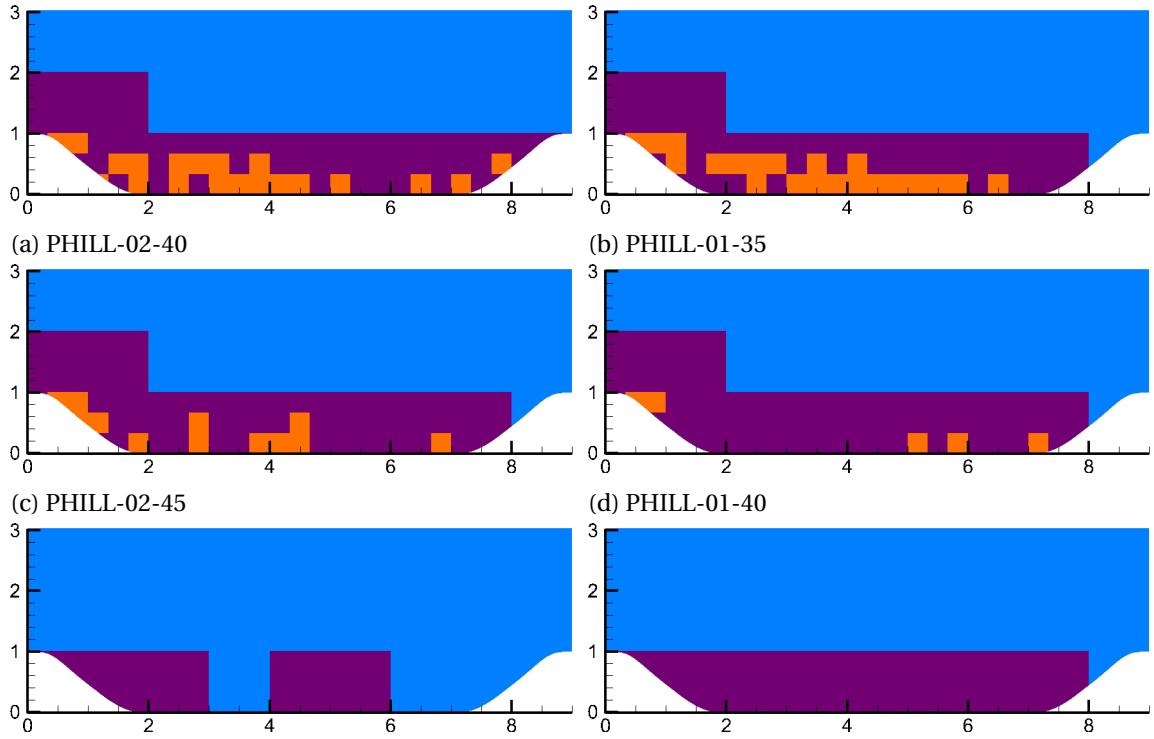


Figure 6.7: Line plots of homogeneous x- and y-velocity of periodic hill case adapted with relative error. Total error line plots of simulations from Table 6.1. Results for line plots have been scaled by a factor of two to improve readability. INCA LVL3 (—), PHILL-01-35 (—), PHILL-02-40 (—).

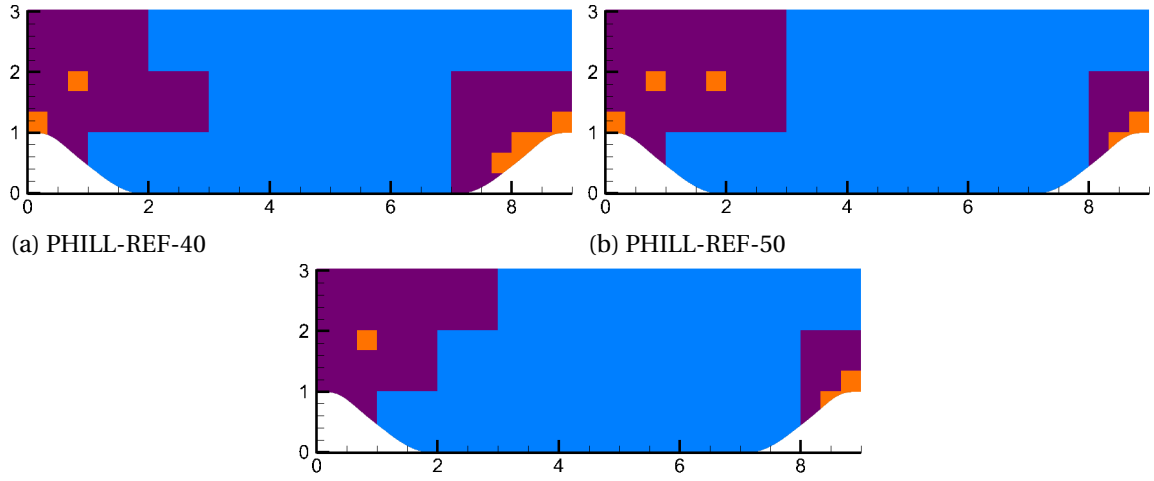


(a) PHILL-02-40 (b) PHILL-01-35

(c) PHILL-02-45 (d) PHILL-01-40

(e) PHILL-02-55 (f) PHILL-01-45

Figure 6.8: Meshes of periodic hill runs from Table 6.1. Mesh levels: Level 1, Level 2, Level 3.



(a) PHILL-REF-40 (b) PHILL-REF-50

(c) PHILL-REF-60

Figure 6.9: Meshes of reference error sensor simulations for flow over periodic hills from Table 6.2. Mesh levels: Level 1, Level 2, Level 3.

6.3.2. Benchmarking against reference error sensor

Three simulations have been performed with the reference error sensor. As was done for the Richardson-type error sensor, a 1% threshold has been chosen in Algorithm 2. All experiments are tabulated in Table 6.2. The results of the simulations in comparison to the previous ones are shown in Figure 6.10.

Table 6.2: Settings of simulations for periodic hill case with reference error sensor.τ_{AMR} = 0.05, 23 FTT settling, 55 FTT statistical sampling, 1 FTT AMR sampling.

Legend	Identifier	Cells	ϵ_{AMR}	Flag function	Flag threshold
	PHILL-REF-40	652455	0.40	Algo. 2	1%
	PHILL-REF-50	564975	0.45	Algo. 2	1%
	PHILL-REF-60	448335	0.55	Algo. 2	1%

From Figure 6.10, one can see that the global error level of the reference error sensor calculated from the line plots is significantly lower for the x-velocity in comparison to the Richardson-type error sensor. For the y-velocity, the error levels are, however, on par. The simulations adapted by the reference error sensor suffer from inefficiencies when considering the PTC count. This is especially clear to see for the x-velocity error. In terms of computational efficiency, the reference error sensor performs better for the x-velocity but shows worse performance for the y-velocity. For the latter, the same error level is reached but only at a higher cell and especially PTC count. When looking at the mesh in Figure 6.9, it is apparent that the refinement strategy of the reference error sensor is different. Adaption is mainly focused on the windward side of the hill. Further, some refinement is visible on the hillcrest which also spreads out to the top of the channel wall. Two line plots are shown for simulation P23 and P31. The reference error sensors matches the results from the uniform level 3 refinement nicely at $x/h = 0.05$ and $x/h = 0.5$. This can be seen as the result of the level 2 refinement, which extends at the inflow boundary from the hill crest up to the top wall of the channel. Additional refinement is given to the hillcrest with level 3 blocks. The results are, however, deteriorating quickly. At $x/h = 2, y/h \approx 1.5$, there is already a significantly higher error than in comparison to the Richardson-type error sensor. The same holds for the lower wall, where the influence of the low mesh level of merely 1 becomes apparent as well. Traveling further downstream at $x/h = 2$ and $x/h = 4$, also the error in the top boundary layer grows again to values that are in the same ballpark as the one from the error estimate. Also, the error below $y/h < 2$ has further increased. At $x/h = 8$ the results again match closely the ones from the uniform level 3 mesh. While for the reference error sensor, the overall error for the x-velocity error is lower than for the Richardson-type error sensor, the quality of adaption can be regarded as somewhat underwhelming. The good performance at the initial parts of the channel stems from the refinement of the hillcrest and the top wall. Especially the localized refinement of the boundary layer at the beginning of the domain helps with matching the results of the uniform level 3 mesh. Further downstream, the reference error sensor, however, fails to completely resolve either the top wall or the recirculation area, which then, in turn, leads to high error production. The consequences are apparent. The error in the boundary layer is steadily growing while moving further downstream. In contrast, the refinement of the Richardson-type error sensor in the recirculation bubble seems to be reasonable and advantageous, leading to an error, that even with the pollution from the under-resolved boundary layer, can match the results of the uniform level 3 mesh, especially further downstream, better than the reference error sensor. From the observations, one can conclude that the Richardson-type sensor actually performed solid, if it would not wholly fail to capture the top wall for refinement. The curvature error sensor is in this regard somewhat better but also not satisfactory. It would be interesting to see what the effects of a wall function might be when applied to the top wall, as was done in Fröhlich *et al.* [15]. Such an approach might alleviate the problems seen with the Richardson-type error sensor.

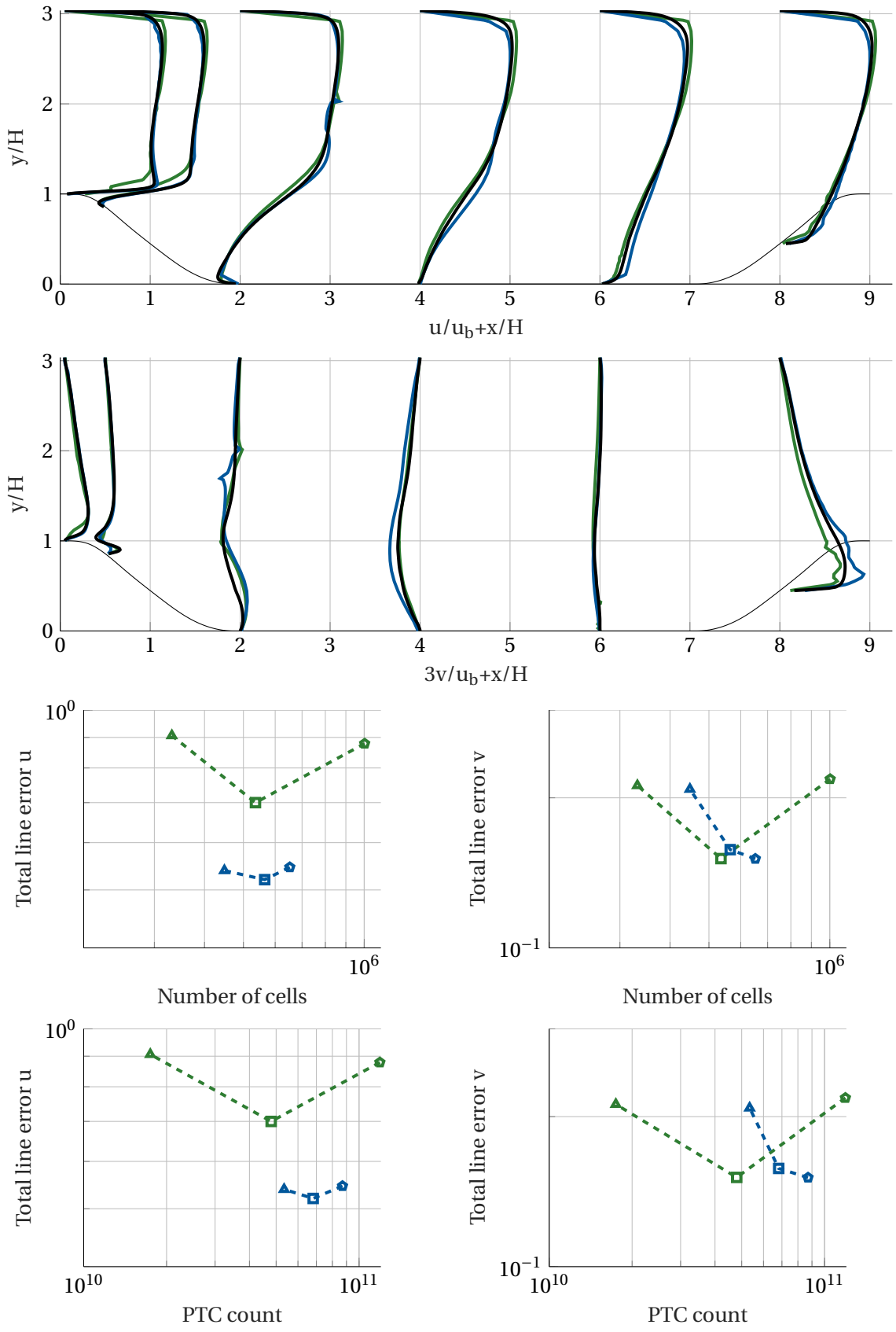


Figure 6.10: Line plots of homogeneous x- and y-velocity of periodic hill case adapted with relative error and reference error sensor. Total error line plots of reference error sensor from 6.2. INCA LVL3 (—), PHILL-01-40 (—), PHILL-REF-50 (—).

6.4. Error evolution during adaption

One aspect that has not been treated in detail is in-time behavior for the AMR-routine. For that purpose, a series of four plots (Figure 6.11) is created, showing the evolution of the mesh size and estimated global error. Results are shown for simulation 'PHILL-01-40' and 'PHILL-02-40' from Table 6.1. All data are given for the first 23 FTT of the simulation since during the statistical sampling period of 55 FTT no mesh adaption is performed.

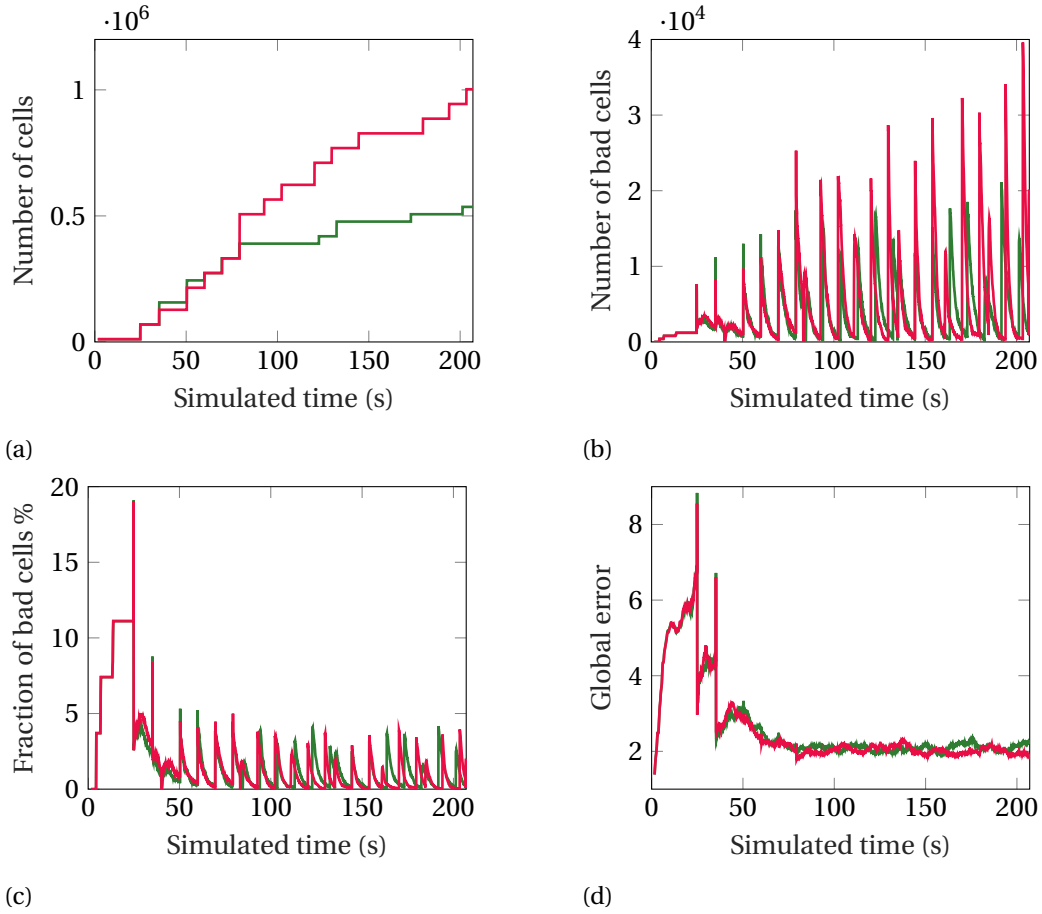


Figure 6.11: Cell and error evolution during adaption with Richardson-typ error sensor for flow over periodic hills. PHILL-01-40 (——), PHILL-02-40 (——).

Figure 6.11a shows the growth in mesh as adaption commences. The first observation that can be made is that the mesh growth can be significant per adaption cycle. The 10% threshold from Algorithm 5 merely states that 10% of bad cells should be targeted for refinement. Since every adaption action is performed on a per-block-basis, the actual quantity of refined mesh cells is much larger. Additional refinement is performed by the 'Balance Criterion', which ensures a proper grid topology. Nevertheless, it is visible how the mesh is built up gradually. While the mesh size increase for simulation 'PHILL-01-40' begins to stagnate from 75 seconds onward, 'PHILL-02-40' continues to steadily grow until the end of the 23 FTT long settling period.

Figure 6.11c shows the fraction of cells exceeding the error threshold. As the simulation starts, the fraction is steadily increasing until, after the first adaption, it drops sharply. This happens at around 25 seconds of simulated time. The mesh is still coarse at that point resulting in a time step that exceeds τ_{AMR}. The sampling time thus actually exceeds the target of 1 FTT at the very onset of the simulation. From around 50 seconds onwards, a steady saw-tooth pattern develops in the bad cell fraction. At the beginning of every AMR sampling period, many cells exceed the threshold. As

sampling continues, fewer cells violate the error threshold. While it is clear that there is no complete convergence yet after 1 FFT, it is apparent that the fraction of bad cells starts to level off sharply. This finding is in agreement with the results from Section 6.2. In the future, this metric might resemble a more mathematical thorough way of deriving the appropriate AMR sampling time than the semi-qualitative approach presented before. After 75 seconds of simulated time, one can see that the fraction of bad cells drops to almost zero after every saw-tooth cycle. Adaption is thus driven by only a tiny amount of bad cells. Also, at precisely 75 seconds of simulated time, a knee-point can be found in Figure 6.11d, which shows the estimated global error in the simulation calculated from the master and slave simulation according to Equation 3.7 integrated over the entire domain. Beyond this time point, the global error stays constant, and any commencing adaption seems to have no effect on any further error reduction. Last but not least, at 75 seconds also, the mesh growth of both simulations runs apart, as shown in Figure 6.11a. One can conclude this that any refinement past 75 seconds is not advantageous and that applying a small filter to the error sensor does indeed help to mitigate useless adaption.

These observations shed some new light into the previous analysis from Section 6.3.1. First, adaption without filtering is driven by only a tiny amount of cells leading to an evergrowing mesh. These small cells also do not contribute significantly towards the estimated global error, causing a computationally inefficient mesh, which can be seen by the fact that the global error evolution of both simulations is almost identical even though 'PHILL-02-40' contains about double the cell count. Based on the results, it seems that a higher threshold could improve the mesh by, e.g., allowing the user to reduce the error threshold while maintaining a moderate mesh size. This could lead to refinement of regions with a smaller error magnitude but larger amounts of cells violating the error threshold. An example of such a weak feature might be the top boundary layer. The time series of the global error estimate could also be used as an additional feature to control refinement. Instead of adapting until all cells are pushed below a certain threshold, adaption could be performed until the global error is converged, thus avoiding over-refinement.

6.5. Conclusion

The turbulent flow over periodic hills indeed posed the greatest challenge among all test cases highlighting that AMR in fully turbulent flows remains to be challenging. The proposed method of averaging the error sensor prior to adaption was successful but is only viable for statistically steady flows. Defining an error sensor sampling period does not add to the user-independence, but two strategies have been presented on how this parameter can be derived systematically. Future research might include using the fraction of bad cells as an automatic way of determining the appropriate sampling period. Sanitization of the error sensor output, by only refining blocks, when a certain percentage of cells exceeds the error threshold, has been successfully applied. This led to a more efficient adaption by avoiding cells that do not contribute to the actual error within the solution. Both the reference and the Richardson-type error sensor from Equation 3.4 did not perform satisfactorily. The adaption of the reference error sensor focused primarily on the wind-ward and crest of the hill, while it neglected large parts of the top wall and the entire recirculation region. The Richardson-type error sensor performed in this regard much better, by refining the hill and a sensible part of the recirculation region. While it was able to spot the top wall as a potential error source, the estimated error was not of sufficient magnitude to lead to adaption. The under-refinement of the boundary layer led to error production, which polluted the solution in the entire domain, leading to a higher global error than the reference error sensor. More research is required to find ways of making the novel error sensor more sensitive to error production for near-wall flows with little pressure gradient. A possible ansatz is a more in-depth investigation of the influence of the filtering percentage of the error sensor.

7

Conclusion and recommendations

The objective of this master's thesis has been to:

Improve the computational efficiency of LES CFD simulations by developing and testing a novel more user-independent AMR error sensor.

Based on the finding from the literature study, a series of novel error sensors have been developed that estimate the local error in the solution of a fluid problem by comparing two simultaneously computed simulations on a fine and coarse grid. Their efficient implementation in the block-structured CFD code INCA has been presented. Message Passage Interface (MPI) was used to couple INCA against itself using, allowing for the computation of the error sensors with minimum interference in the codebase. A series of three test cases have been chosen to demonstrate the workings of the novel error sensors for a wide range of flow problems. For a better performance comparison of the differently adapted simulations, the PTC metric has been introduced that aims at giving a computer system and code overhead independent measure for the simulation run-time. At the onset of this thesis, a set of research questions has been developed which are answered in the following:

The research questions from Chapter 2 are:

1. Are the current state of the art error sensors already able to provide a systematical approach to user-independent and reproducible adaptive mesh adaption?

   Answer: Based on the literature study, one has to say that it is highly problem dependent. Since some error sensors can be tailored fairly well to the underlying problem, mesh generation can indeed be user-independent if only a subset of problems is considered. For practical applications in industry, this might be sufficient when similar flow problems are solved on a repetitive basis. User-independent AMR in a broader sense concerning a wide variety of problems does not seem to be in reach yet. Output-based adaption might be a suitable candidate but, at least in their pure form, are still too expensive to compute for transient LES problems. Numerical estimators seem to be the best candidate for problem-independent adaption, especially because the performance of generic LES-specific error indicators seems to be underwhelming.

2. For all three test cases and isotropic refinement, how does the resulting solution error compare in terms of cell count and PTC for the novel and reference error sensors?

3. For all three test cases and anisotropic refinement, how does the resulting solution error com-

pare in terms of cell count and PTC for the novel and reference error sensors?

Answer to 2. and 3.: Even though the reference error sensor is primitive, it still performed decently. Especially for the compressible and laminar test case. For the compressible test case, all examined error sensors performed almost identical. The main issues were mainly in the controllability, i.e., how well suitable error thresholds can determine the size of the mesh. Nevertheless, this behavior is also in close connection to the chosen flag function. With anisotropic refinement, it was possible to increase the computational efficiency by around 10 to 15% concerning the cell size. The laminar test case showed auspicious results for the Richardson-type error sensors, which were able to lead to savings of roughly 4 to 5 times in PTC and cell count in comparison to the reference error sensor. The performance of the ETE-based error sensors was acceptable, sometimes outperforming the reference error sensor but sometimes also performing worse. A large disappointment has to be seen in the anisotropic refinement. While anisotropic lead to a consistent reduction in cell count for the same error, a less well-conditioned Poisson matrix led to a higher number of required iterations to reach the target residual, thus completely negating any gains and often leading to worse performance than isotropic refinement. Due to time constraints, no anisotropic adaption was performed for the turbulent test case. For the latter, the reference error sensor was able to outperform the Richardson-type error sensor from Equation 3.4 consistently for the error in x-velocity. Nevertheless, this is a pure consequence of the limited refinement of the top boundary layer by the reference error sensor. In contrast, the novel error sensor failed to refine the entire top boundary layer at all. The mesh created by the reference error sensor in all other regions of the domain must, however, be regarded as suboptimal, with the novel error sensor doing a much better job.

4. Should mesh adaption be performed every physical time step, or can the computational efficiency be increased by performing adaption merely every N time steps.

   Answer: Tracing the AMR routine has shown that the entire AMR routine, including adaption, is very cheap to compute. The cell savings heavily outweigh the little overhead created by the routine. When adaption was performed every five physical time steps, an overhead of about 3.5% was recorded. For the Richardson-type error sensor from Equation 3.4, it was shown that a too-small AMR time step could lead to worse computational performance. One should, therefore, conclude that it is best to perform adaption not every time step. The exact quantity is problem-dependent. However, it was also found that for both the laminar and the turbulent test case, the adaption routine was quite robust to any change in the AMR time step, adding significantly to the user-independence.

5. Will all investigated error sensors lead to a robust adaption routine?

   Answer: In terms of initial mesh size, this question can be answered with yes. It was possible to start the simulations on very coarse meshes, often merely limited by a minimum cell requirement in the block itself. All error sensors were able to deal with a heavily under-refined initial mesh and lead to sensible adaption. To the author's opinion, the immersed boundary method, in conjunction with the block-structured mesh employed within INCA, was crucial for this property. With the immersed boundary method, it was possible to resolve a geometry by as little as two initial cells, e.g., for the periodic hill problem, and still get the simulation started. In terms of error sensor reduction, the question should be answered with no. In the vicinity of discontinuities, the error sensor output will steadily grow, thus leading to never stopping adaption. However, for reasonable smooth flow fields, also close to an immersed boundary, the error would steadily reduce with increasing adaption. Nevertheless, it was still necessary to bound the maximum refinement level of every simulation with the rationale that

otherwise locally microscopic mesh cells would have been created that had led to an excessively small time step. Thus letting the PTC count skyrocket.

6. Should adaption be based upon the instantaneous or time-averaged state variables for the turbulent flow?

   Answer: It has been shown that for turbulent flow, chaotic error sensor structures exist. With increasing averaging periods, these structures started to vanish, which can be seen qualitatively be examining the error sensor field, or by, e.g., looking at the number of cells violating the error threshold. Unfortunately, time averaging is only applicable to statistically steady problems.

7. What is the computational overhead of the tested error sensors?

   Answer: The computational overhead is a compound of the time required to calculate the error sensor and to perform the adaption. For the error estimates, a second coarse simulation is required. The computational overhead for the error sensor itself is given in Answer 4. The second simulation, for the laminar test case, led to an extra 11% in extra cells and 4.5% in PTC count.

8. What kind of qualitative adaption characteristics do the investigated error sensors show for all three test cases?

   Answer: For the compressible flow case, all investigated error sensors performed very similarly. They were all able to capture the most important flow features such as shock waves, slip lines, and error introduction from the step corner singularity. The behavior was more diverse for the laminar flow case. The reference error sensor focused itself mostly on the front half of the cylinder and led to the refinement of vast parts of the wake up to the end of the domain. The Richardson-type error sensors led to a more balanced refinement of the cylinder between front and rear half while creating a fine mesh in the near wake and coarse refinement in a narrow band in the far wake. The behavior of the ETE-based error sensors was similar to the reference one but did put more focus on the near-wake refinement. For the turbulent test case, the reference error sensor was able to spot all interesting flow regions, the top-wall boundary layer, the hill, and the recirculation region. Unfortunately, the magnitudes of the output where completely shifted towards the hill leading to poor refinement of the rest of the domain. The Richardson-type error sensor from Equation 3.4, on the other hand, was able to lead to good refinement in the recirculation region but completely failed to resolve the top boundary layer.

Overall it is the author's opinion that the objective has been met. Especially the novel error sensor based on Equation 3.4 performs measurably better than the chosen reference error sensor. An aspect already criticized within the literature study is the difficulty of comparing different approaches with respect to each other. Every study uses different test cases, methodologies, and metrics. A large quantity of research has been done in the field of AMR, but it is still challenging to determine what the current 'gold standard' is. The results of this master's thesis can thus only put into relation with the rather simplistic reference error sensor.

As discussed in Section 5.7, an aspect that is, according to the author's opinion, often overlooked is the question on what to do when one has found a (near) perfect error estimate. Deciding how to adapt, i.e., refine and coarsen, is even then not straight forward. AMR is thus not only dependent on an excellent error sensor, but also on a powerful flag function. Very little literature has been found concerning the last aspect, a surprising outcome.

In the literature study, a list of twelve properties has been defined that could describe an (almost)

ideal error sensor. In Table 7.1 the attempt is made to describe the reference and error sensor from Equation 3.4 according to the properties. While one can not claim that this list is perfect, it still leads to better comparability between different error sensors.

Table 7.1: Summary of characteristics of reference error sensor and Richardson-type error sensor from Equation 3.4.

Property	Reference error sensor	Equation 3.4 error sensor
1	Very cheap	Cheap
2	Low error for compressible and laminar flow	Low error for compressible and laminar flow. Encouraging results for turbulent flow.
3	Averaging period for turbulent flow.	τ_{AMR}, averaging period for turbulent flow.[1]
4	Can only indicate errors	Estimates the error due to the spatial discretization, seems usable for grid convergence study
5	Mostly heuristic in nature	Based on mathematical arguements
6	Independent of solver, implementation is trivial	Independent of solver, implementation is straight forward but requires some work
7	Not tested	Not tested
8	Discontinuities require attention	Discontinuities require attention
9	No	No
10	Only flow field	Only flow field
11	Works well on coarse meshes	Works well on coarse meshes
12	Acceptable computational efficiency	Good computational efficiency

Based on the outcome of this thesis a couple of recommendations and directions for future research can be stated:

First, it would be interesting to test the error sensors and especially the one from Equation 3.4 in a more industrial type of problem with the aim of gaining more insights into the limitations of the error sensor. This also includes the extension to adaption in all three principal directions. Additionally, some research should be done on how to increase its sensitivity in flows with boundary layers and other strong flow features. A possible solution might lie in the constant in Algorithm 3. For the laminar test problem, anisotropic refinement has shown some encouraging results in terms of mesh size savings. Unfortunately, the conditioning of the Poisson matrix deteriorated. The introduction of coarsening is essential not only for problems that are not statistically steady but also to allow for refinement with a target mesh size instead of an error threshold. This should significantly enhance the user-independence of the presented AMR routine, as discussed in Section 5.7.

[1] Settings such as ϵ_{AMR} are a setting of the flag function.

A

Code implementation

In this chapter, the Fortran code implementation of the theoretical framework is explained. Particular focus is laid on INCA's block-structured architecture, which uses MPI to exchange information between blocks on different processors. First, the general approach is described that has been used to implement the various error sensors and flag functions within INCA. Next, the placement of the entire AMR routine within INCA is described, highlighting how it interacts with other parts of the codebase. Last but not least, a detailed description of the implemented framework is given. Essential, thesis specific, subroutines are discussed in detail.

A.1. Code implementation

For the implementation of the new error-estimate criteria, two implementation strategies were considered upfront. Within the first approach, INCA would have been extended to intrinsically support the simultaneous calculation of a problem on a fine and coarse-grained mesh. The significant drawbacks of this method are the deep interventions within the code that would have been necessary, which not only resemble a formidable programming task but also would have required the extensive study of large parts of INCA's codebase. The idea of the second approach was to couple INCA with itself and therefore run two instances of the code simultaneously, one computing the fine problem and one computing the coarse problem. This approach is much more accessible since large parts of INCA's code can stay unchanged, and only a couple of subroutines are necessary that, at the correct time, establish communication between the two launched instances of the code. Since INCA already uses MPI (Message Passing Interface), the extension was straightforward. The second approach, therefore, resembled a much slicker method on achieving the same goal which in the retrospective proved to be true since the implementation happened without any big issues.

A.2. AMR framework integration within INCA

Algorithm 6 shows a rough sketch of the main structure of INCA and the placement of the AMR routine within. Apart from the necessary steps at the beginning and end of a CFD program instance, such as memory allocation and deallocation, the main part of INCA consists of a task routine handling the time stepping of the transient simulation. Within this task routine, two important AMR specific tasks are executed every single iteration. First, the synchronization of the simulation time step. INCA uses adaptive time-stepping based upon a target CFL number. Since this number is mesh dependent, a coarse and fine-grained mesh would produce different time steps and thus, without time synchronization between the master and slave simulation, they would advance at different rates in time. While this could be used to speed up the slave computation, provided that at some point in time synchronization happens, it was decided for simplicity to perform synchronization every single time step. As a result of this, the CFL number of the slave simulation is lower than the one of the master. No direct investigation has been performed on the influence of this on time-dependent errors such as dispersion and phase lag. However, as will be explained in Section A.3 also a solution synchronization takes place, which effectively resets both of these errors after mesh adaption occurred. The algorithm shows how the time step syncing works. The time step is exchanged between the root ranks of the master and slave simulation. Subsequently, the slave root then broadcasts this time step to all other slave ranks. Secondly, the AMR call itself, which, however, only happens when a certain amount of elapsed simulated time has passed.

Algorithm 6 AMR framework integration within INCA.

```
1:  setup simulation                          ▷ Communicate block-rank assignment to other process
2:  ...
3:  while final time has not been reached yet do
4:      ...
5:      if master then
6:          calculate time step
7:          if root then
8:              send time step to slave root
9:          end if
10:     else if slave then
11:         if root then
12:             receive time step from master root
13:         end if
```

```
        broadcast timestep in slave group
    end if
    ...
    perform time stepping
    ...
    if τ_AMR seconds of simulated time elapsed then
        call AMR FRAMEWORK (see sec. A.3)
    end if
    ...
    do post-processing
end while
write final results
exit gracefully
```

A.3. Main structure and subroutines of AMR framework

Algorithm 7 shows the flow chart of the implemented AMR routine. The first time the AMR routine is called, usually at the very first iteration, the cut-cell auxiliary criterion is invoked. This happens within a while loop for all blocks and happens until all immersed boundaries are resolved to a target mesh level. The reason for this is that in INCA, every mesh adaption can happen only sequentially i.e., the subroutine that refines a block must be called multiple times when the mesh level is supposed to be increased by more than one. After the cut-cell criterion does not change the mesh anymore, it is never executed again for the remainder of the entire simulation.

Algorithm 7 Top-level adaption routine.

```
if first ever AMR call then
    for all blocks do
        while until mesh does not change anymore do
            call CUT-CELL CRITERION (see sec. 3.3.2)          ▷ Only one level at a time
            perform block splitting and refinement
        end while
    end for
end if
for i = 1:2 do
    call START SERVER (see sec. A.3.1) ▷ Communicate block-rank assignment to other process
    for all blocks do
        call MULTI CRITERION (see sec. A.3.2)          ▷ Wrapper for error sensor calculation
            retrieve and compute QOI e.g. velocity magnitude
            call RICHARDSON (see sec. A.3.3)          ▷ Routine that calculates error estimate
                send the QOI from coarse to fine grid via MPI
                restrict the h-grid solution to the H-grid
                calculate error according to methods from Section 3.5.1 and pass it to flag function
            end call
            call BLOCK FLAG FUNCTION (see sec. A.3.4)      ▷ Should a block be refined or split
        end call
    end for
    call GLOBAL FLAG FUNCTION (see sec. A.3.5)                ▷ Filter decision from 19:
    copy the fine grid state variables to the coarse grid
    call STOP SERVER (see sec. A.3.1)     ▷ Clear block-rank assignment for new adaption cycle
    perform block splitting and refinement
```

26: **end for**

The first step is to start a server on both the master and the slave process of the simulation. The master process contains the fine grid simulation, whereas the slave process contains the coarse-grained simulation. The naming stems from the fact that all mesh adaption decisions are made within the master simulation and the slave then subsequently follows them. The server provides a list on which MPI ranks the blocks of INCA's block-structured grid can be found for either the slave or master simulation. Afterward, a wrapper subroutine (MULTI CRITERION) is called sequentially for every block on all ranks. This wrapper routine provides easy access to all variables of interest, e.g., velocity magnitude, and feeds them into an error sensor pipeline, which is explained more in-depth in Section A.3.2. The error estimate, which is produced from this pipeline, is fed into the BLOCK FLAG FUNCTION. This flag function incorporates algorithms 2, 3 and 4. It is apparent that all these algorithms run independently, thus do not require any communication with other blocks. The output of all the flag functions is collected within the GLOBAL FLAG FUNCTION, which resembles the implementation of Algorithm 5. Last but not least, the master solution is given to the slave ranks, the mesh is adapted and the server stopped. At this point, the simulation commences as usual until the next τ_{AMR} seconds have passed.

A.3.1. Start server / Stop server

Every process of INCA contains a processor table that associates block location to process rank. This table is used for the exchange of block information between different processes when INCA is launched with more than one rank. This catalog is the same for all ranks within one instance of INCA. Coupling INCA with itself, however, means that this catalog has to be exchanged between all ranks of the master and slave instance so that communication can be established. This is the purpose of the server, whose workings are shown in Figure A.1. First, the processor tables are exchanged between the root ranks of the master and slave simulation. Subsequently, they are broadcast to all non-root ranks in their respective group. Every time a mesh change takes place, these tables change and have to be updated. This happens by deallocating the processor tables (stop the server) and reinitializing the table exchange (start the server).

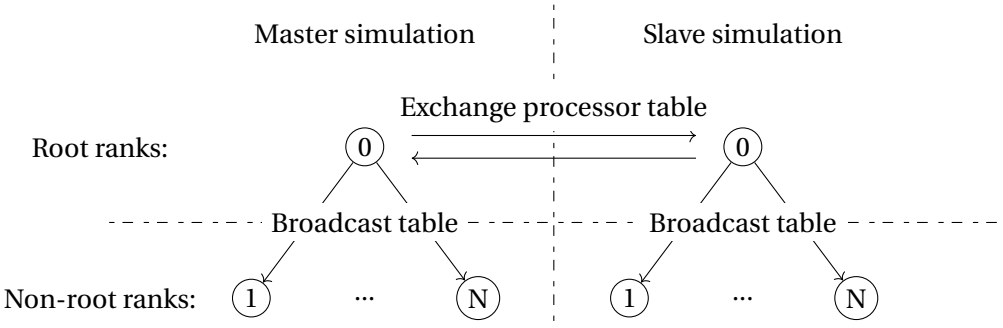


Figure A.1: Server start: Exchange of processor table via MPI.

A.3.2. Multi criterion and quantity of interest pipeline

Lots of indicators and error-estimates can be found within AMR literature, together with the fact that different error-estimates are tested within this thesis, it was desired to have a streamlined approach of implementing these. A requirement was that a new criterion should be ideally implementable without having to recompile the code, and in case of recompilation, the implementation should be very fast. To achieve this, a 'multi-criterion' was developed, which serves as a wrapper subroutine for all error-estimates and indicators used throughout this thesis. Important for the implementation is the realization that most error functions are obtained by simple operations from the state

variables, such as taking the derivative, calculating the difference, or multiplying them with the cell length. As a solution to fulfill the aforementioned needs every of these operations was put in a separate function within INCA, the following table lists all of these functions, while most of these functions are trivial, the Richardson function is discussed in more detail in Section A.3.3 due to its communication architecture between master and slave solution.

- **uvar:** This function delivers a state variable or a combination thereof. E.g. the pressure but also the magnitude can be queried. The function also translates any staggered variables into a cell-centered frame.
- **gradient:** This function calculates the gradient of its input using a central second-order finite difference scheme.
- **curvature:** This function calculates the second gradient of its input using a central second-order finite difference scheme.
- **cell lengths:** This function is used to multiply the input by the respective cell length to e.g. produce an undivided difference.
- **richardson:** Calculate the difference in a variable between a fine and coarse-grained simulation.

Using these functions the implementation of most error indicators and estimators is very easy. For e.g. calculating the undivided difference of the velocity magnitude, a popular indicator in literature, as discussed in Section A, one simply has to call 'uvar', 'gradient' and 'cell lengths' sequentially.

A.3.3. Richardson

The Richardson function (Figure A.2) is used to calculate the error between the coarse and fine-grained simulation as shown in Equation 3.5, 3.3 and 3.4. As shown in Algorithm 6, the function is called for every block separately. Using the catalog from the 'Server'-subroutine, the master and slave ranks holding the respective block can establish direct rank-to-rank communication. For the calculation of the error, the slave has to send its array of the quantity of interest, the grid size and the wall distance to the master simulation. The coarsening method, as described in Section 3.7, is used to restrict the master simulation on the slave grid. Subsequently, the error estimate of the quantity of interest can be produced. Directly at the interface of an immersed boundary, this error estimate is not well-defined, since it is not obvious how to consistently restrict the solution at this location. Due to the higher resolution of the fine mesh, the immersed boundary cuts only some cells and thus usually leaves some cells within the geometries, whereby definition e.g., the velocity is zero. Leaving this issue unattended causes erroneously high error estimates, which would always flag blocks for refinement that contain and IB. To circumvent this problem, it was decided to filter the error estimate by the wall distance, all cells which either have a negative wall distance (lie within a geometry) or zero wall distance (lie on an IB) will have zero error. As shown later in Section 5, this ad hoc fix works very well, since the error of the cells close to the IB can still grow sufficiently large to cause adaption, while also effectively decreasing when adaption commences. As a final step, the error estimate is prolonged to the fine mesh so that additional operations can be easily performed on it within the master simulation, such as the calculation of the error transport equation shown in Equation 3.8.

A.3.4. Block flag function

The block flag function contains the implementation of algorithms 2, 3 and 4. Since these blocks are calculated locally for every single block, the implementation is trivial and thus does not require any specific documentation. At the end of the block flag function, the alternating block splitting and

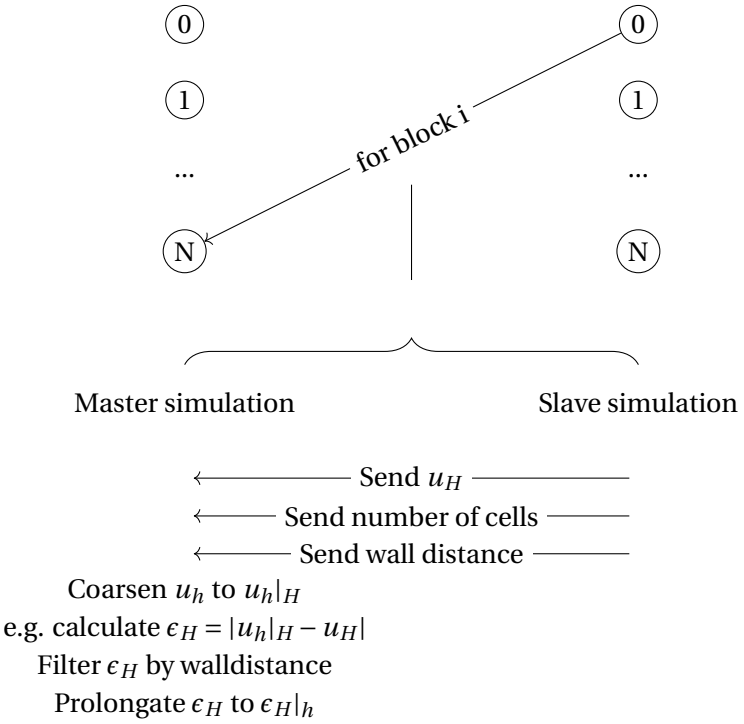


Figure A.2: Richardson function calculating the difference in a fine and coarse primary variable.

refining criterion (Section 3.3.1) is invoked, which checks if a block that violates an error threshold should either be refined or split.

A.3.5. Global flag function

While the theory of Algorithm 5 is straightforward, the actual implementation is more advanced. The reason for that is INCA's block-structured mesh, in which all operations are independently performed on every block. The communication between the blocks is realized by copying values from neighboring blocks into the ghost layers. Since the blocks are scattered throughout the different ranks of an INCA instance, the global flag function criterion of Algorithm 5 must use information that is collected from all blocks of the master INCA instance. The first step is, therefore, that the root rank of the master simulation gathers an array containing the error-estimate and number of cells of the blocks from all ranks. Further, all the mesh-actions calculated by the local flag function are collected as well. From this, all blocks can be sorted according to their maximum error. Subsequently, the total number of 'bad cells' is calculated, i.e., the number of cells contained in the blocks flagged for refinement. The next step is to sequentially select blocks, starting from the block with the highest error in descending order, until a certain target percentage of bad cells has been selected. The last step is to use the selected blocks to filter the mesh-actions. That means, all blocks that were flagged for refinement from the local flag function, but not selected from the global flag function, will not be refined. On the contrary, any splitting action will simply be passed-through and not changed by the global flag function. During the final MPI communication, the modified mesh-action will be broadcast from the root rank to all other ranks within the master simulation. The master root rank also sends the mesh-action to the root rank of the slave simulation, which then broadcasts it to all other ranks within its INCA instance.

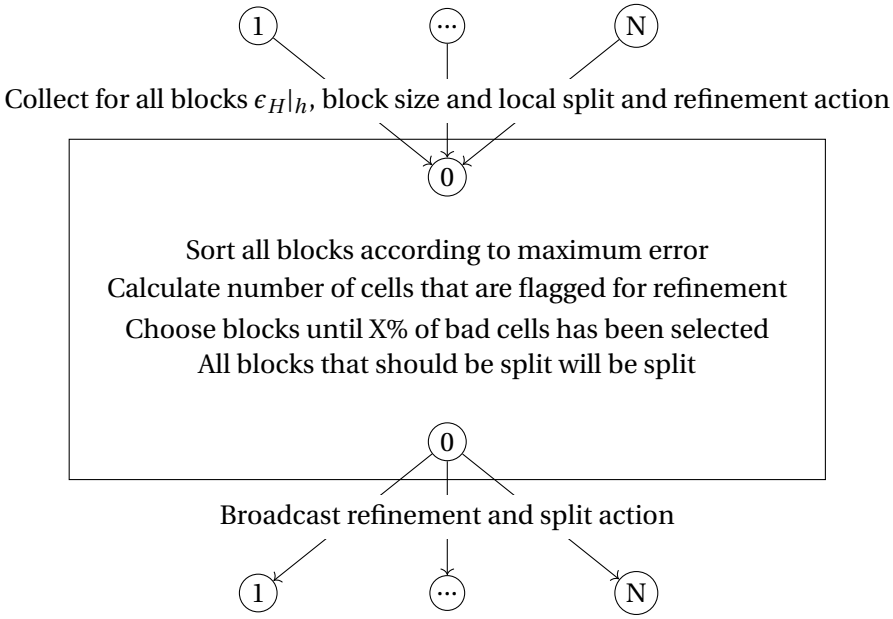


Figure A.3: Schematics of global flag function.

B

Poisson-timestep-cell performance metric

A natural approach to indicate the performance improvements associated with a certain AMR method is to report the cell count savings. The main drawback of this approach is the assumption that cell count directly relates to the time it takes to solve a case. It is obvious that a case with more cells takes longer to compute. However, it also neglects effects that potentially have a much more substantial impact. In the case of a transient simulation with a target CFL number, refining just one single block by another level can increase the wall-clock time by almost an order of magnitude (considering INCA's staggered mesh and a standard refinement ratio of 3) due to a smaller τ. Another observation made during this thesis is that usually the amount of Poisson solver iterations increases when the mesh becomes finer. An obvious solution would be to report the wall-clock time or the core hours of the simulation, as well as the hardware which was used for the computation. This metric, however, also lacks practicality or gets polluted by various external factors. Using a core hour metric means that cases must be computed on the same hardware to ensure compatibility. Depending on the available resources, this is not always practical since one might have to compute its cases on different nodes/workstations. Load balancing is an additional source of large uncertainty, depending on how often load balancing is used, and of course, based upon which load-balancing method is employed, the load between the different CPUs is usually not uniformly distributed, meaning that some cores might idle whereas other cores run at 100% load. Even if all these problems might be avoided, also entirely random external influences can pollute the results. Depending on e.g., the cluster architecture, all users might have to write to the same storage server. This can lead to bottlenecks if various simulations have to write-out simulation results at the same time. To the author's opinion, one is usually interested in the maximum theoretical performance improvement, which can result from an AMR routine instead of the actual improvements, especially because early implementations of new algorithms have not yet reached a high level of maturity and optimization. As a remedy, but surely not a perfect one, the following metric PTC (Poisson-timestep-cell-iterations) is proposed for incompressible cases:

$$PTC = \sum_{NT_{\text{start}}}^{NT_{\text{end}}} \text{cells} * \text{poisson solver iterations}. \tag{B.1}$$

It assumes that the largest wall clock time contributor of a transient incompressible CFD simulation stem from the number of cells, the number of Poisson-solver iterations and the number of time-steps.

C

Mathematical description of periodic hill geometry

The geometry of the left hill can be described by the following set of splines. For the right hill, the coordinates simply have to be mirrored. The definition of the splines stems from ERCOFTAC [79].

Between x=0. and x=0.3214h:

$$h(x) = \min(1, 1 + 0x + 2.420\text{E}{-}04x^2 - 7.588\text{E}{-}05x^3) \tag{C.1}$$

Between x=0.3214h. and x=0.5h:

$$h(x) = 0.8955 + 3.484E\text{E}{-}02x - 3.629\text{E}{-}03x^2 + 6.749\text{E}{-}05x^3 \tag{C.2}$$

Between x=0.5h and x=0.7143h:

$$h(x) = 0.9213 + 2.931\text{E}{-}02x - 3.234\text{E}{-}03x^2 + 5.809\text{E}{-}05x^3 \tag{C.3}$$

Between x=0.7143h and x=1.071h:

$$h(x) = 1.445 - 4.927\text{E}{-}02x + 6.950\text{E}{-}04x^2 - 7.394\text{E}{-}06x^3 \tag{C.4}$$

Between x=1.071h and x=1.429h:

$$h(x) = 0.6401 + 3.123\text{E}{-}02x - 1.988\text{E}{-}03x^2 + 2.242\text{E}{-}05x^3 \tag{C.5}$$

Between x=1.429h and x=1.929h:

$$h(x) = \max(0, 2.0139 - 7.180\text{E}{-}02x + 5.875\text{E}{-}04x^2 + 9.553\text{E}{-}07x^3) \tag{C.6}$$

Bibliography

[1] M. Aftosmis and M. Berger, *Multilevel Error Estimation and Adaptive h-Refinement for Cartesian Meshes with Embedded Boundaries,* in *40th AIAA Aerospace Sciences Meeting & Exhibit*, Aerospace Sciences Meetings (American Institute of Aeronautics and Astronautics, 2002).

[2] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks,* Journal of Computational Physics **54**, 115 (1984).

[3] B. Krank, M. Kronbichler, and W. A. Wall, *Direct Numerical Simulation of Flow over Periodic Hills up to Re H= 10 , 595,* Flow, Turbulence and Combustion **101**, 521 (2018).

[4] M. Breuer, N. Peller, C. Rapp, and M. Manhart, *Flow over periodic hills - Numerical and experimental study in a wide range of Reynolds numbers,* Computers and Fluids **38**, 433 (2009).

[5] M. Meyer, A. Devesa, S. Hickel, X. Y. Hu, and N. A. Adams, *A conservative immersed interface method for Large-Eddy Simulation of incompressible flows,* Journal of Computational Physics **229**, 6300 (2010).

[6] Top500, *Performance development,* (2018), `https://www.top500.org/statistics/perfdevel/`, Accessed: 06.11.2018.

[7] A. Jameson, *Computational fluid dynamics past, present and future,* Presentation at NASA Advanced Modeling & Simulation Seminar Series (2012), `http://aero-comlab.stanford.edu/Papers/NASA_Presentation_20121030.pdf`, Accessed: 06.11.2018.

[8] F. T. Nieuwstadt, B. J. Boersma, and J. Westerweel, *Turbulence: Introduction to Theory and Applications of Turbulent Flows* (Springer International Publishing, 2016) pp. 1–284.

[9] D. A. Venditti and D. L. Darmofal, *Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows,* Journal of Computational Physics **176**, 40 (2002).

[10] S. Toosi and J. Larsson, *Anisotropic grid-adaptation in large eddy simulations,* Computers and Fluids **156**, 146 (2017).

[11] K. J. Fidkowski, *Output-based space–time mesh optimization for unsteady flows using continuous-in-time adjoints,* Journal of Computational Physics **341**, 258 (2017).

[12] R. Löhner, *Applied CFD Techniques: An Introduction based on Finite Element Methods*, 2nd ed. (John Wiley & Sons, 2008) p. 519.

[13] J. Meyers, *Quality and Reliability of Large-Eddy Simulations*, edited by B. Geurts, ERCOFTAC series ; 12 (Springer, 2008) first QLES meeting on Quality and Reliability of Large-Eddy Simulation, held October 22–24, 2007 in Leuven (QLES07).

[14] J. Kim, D. Kim, and H. Choi, *An immersed-boundary finite-volume method for simulations of flow in complex geometries,* Journal of Computational Physics **171**, 132 (2001).

[15] J. Fröhlich, C. P. Mellen, W. Rodi, L. Temmerman, and M. A. Leschziner, *Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constrictions,* Journal of Fluid Mechanics **526**, 19 (2005).

[16] F. Schornbaum and U. Rüde, *Extreme-Scale Block-Structured Adaptive Mesh Refinement,* SIAM Journal on Scientific Computing **40**, C358 (2018).

[17] L. Diachin, R. Hornung, P. Plassmann, and A. Wissink, *Parallel Adaptive Mesh Refinement,* in *Parallel Processing For Scientific Computing*, edited by M. A. Heroux, P. Raghavan, and H. D. Simon (Society for Industrial and Applied Mathematics, 2006) Chap. 8.

[18] M. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics,* Journal of Computational Physics **82**, 64 (1989).

[19] C. Hertel, M. Schümichen, S. Löbig, J. Fröhlich, and J. Lang, *Adaptive large eddy simulation with moving grids,* Theoretical and Computational Fluid Dynamics **27**, 817 (2013).

[20] I. Babuška and M. Suri, *The p and h - p Versions of the Finite Element Method, Basic Principles and Properties,* SIAM Review **36**, 578 (1994).

[21] P. A. Zegeling, *r-refinement for evolutionary PDEs with finite elements or finite differences,* Applied Numerical Mathematics **26**, 97 (1998).

[22] M. Ceze and K. J. Fidkowski, *Anisotropic hp-Adaptation Framework for Functional Prediction,* AIAA Journal **51**, 492 (2013).

[23] H. Jasak and A. D. Gosman, *Element residual error estimate for the finite volume method,* Computers and Fluids **32**, 223 (2003).

[24] M. Klein, *An Attempt to Assess the Quality of Large Eddy Simulations in the Context of Implicit Filtering,* Flow, Turbulence and Combustion **75**, 131 (2005).

[25] B. J. Geurts and J. Fröhlich, *A framework for predicting accuracy limitations in large-eddy simulation,* Physics of Fluids **14**, L41 (2002).

[26] B. J. Geurts, *Interacting errors in large-eddy simulation: a review of recent developments,* Journal of Turbulence **7**, N55 (2006).

[27] I. B. Celik, Z. N. Cehreli, and I. Yavuz, *Index of Resolution Quality for Large Eddy Simulations,* Journal of Fluids Engineering **127**, 949 (2005).

[28] M. Klein, J. Meyers, and B. J. Geurts, *Assessment of LES Quality Measures Using the Error Landscape Approach,* in *Quality and Reliability of Large-Eddy Simulations* (Springer Netherlands, Dordrecht, 2008) pp. 131–142.

[29] F. di Mare, R. Knappstein, and M. Baumann, *Application of LES-quality criteria to internal combustion engine flows,* Computers & Fluids **89**, 200 (2014).

[30] J. Larsson, *Grid-adaptation for chaotic multi-scale simulations as a verification-driven inverse problem,* in *2018 AIAA Aerospace Sciences Meeting*, January (American Institute of Aeronautics and Astronautics, Reston, Virginia, 2018).

[31] H. Jasak, *Error analysis and estimation for the finite volume method with applications to fluid flows.*, Ph.D. thesis, Imperial College London (1996).

[32] J. Dannenhoffer, III and J. Baron, *Grid adaptation for the 2-D Euler equations,* in *23rd Aerospace Sciences Meeting* (American Institute of Aeronautics and Astronautics, Reston, Virigina, 1985).

[33] Ansys Inc, *Ansys fluent user's guide release 13.0,* (2010).

[34] B. Re, C. Dobrzynski, and A. Guardone, *Assessment of grid adaptation criteria for steady, two-dimensional, inviscid flows in non-ideal compressible fluids,* Applied Mathematics and Computation **319**, 337 (2018).

[35] S. Kamkar, A. Wissink, V. Sankaran, and A. Jameson, *Feature-driven Cartesian adaptive mesh refinement for vortex-dominated flows,* Journal of Computational Physics **230**, 6271 (2011).

[36] J. Gou, X. Yuan, and X. Su, *A high-order element based adaptive mesh refinement strategy for three-dimensional unstructured grid,* International Journal for Numerical Methods in Fluids **85**, 538 (2017).

[37] J. Gou, X. Yuan, and X. Su, *Adaptive mesh refinement method based investigation of the interaction between shock wave, boundary layer, and tip vortex in a transonic compressor,* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **232**, 694 (2018).

[38] S. Pope, *Turbulent Flows* (Cambridge University Press, 2000).

[39] L. Berselli, T. Iliescu, and W. Layton, *Mathematics of Large Eddy Simulation of Turbulent Flows*, Scientific Computation (Springer, 2006).

[40] L. Davidson, *Large Eddy Simulations: How to evaluate resolution,* International Journal of Heat and Fluid Flow **30**, 1016 (2009).

[41] S. Gamard and W. K. George, *Reynolds number dependence of energy spectra in the verlap region of isotropic turbulence,* Flow, Turbulence and Combustion **63**, 443 (2000).

[42] S. E. Gant, *Reliability issues of LES-related approaches in an industrial context,* Flow, Turbulence and Combustion **84**, 325 (2010).

[43] G. Daviller, M. Brebion, P. Xavier, G. Staffelbach, J. D. Müller, and T. Poinsot, *A Mesh Adaptation Strategy to Predict Pressure Losses in LES of Swirled Flows,* Flow, Turbulence and Combustion **99**, 93 (2017).

[44] G. Hindi, E. Paladino, and A. A. M. de Oliviera, *Effect of mesh refinement and model parameters on LES simulation of diesel sprays,* International Journal of Heat and Fluid Flow **71**, 246 (2018).

[45] W. L. Oberkampf and C. J. Roy, *Verification and Validation in Scientific Computing* (Cambridge University Press, Cambridge, 2010) pp. 1–767, arXiv:arXiv:1011.1669v3 .

[46] A. Hay and M. Visonneau, *Error estimation using the error transport equation for finite-volume methods and arbitrary meshes,* International Journal of Computational Fluid Dynamics **20**, 463 (2006).

[47] C. Roy, *Review of Discretization Error Estimators in Scientific Computing,* in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, January (American Institute of Aeronautics and Astronautics, Reston, Virigina, 2010) pp. 1–29.

[48] O. C. Zienkiewicz and J. Z. Zhu, *A simple error estimator and adaptive procedure for practical engineerng analysis,* International Journal for Numerical Methods in Engineering **24**, 337 (1987).

[49] R. Verfürth, *A posteriori error estimation and adaptive mesh-refinement techniques,* Journal of Computational and Applied Mathematics **50**, 67 (1994).

[50] D. Haworth, S. El Tahry, and M. Huebler, *A global approach to error estimation and physical diagnostics in multidimensional computational fluid dynamics,* International Journal for Numerical Methods in Fluids **17**, 75 (1993).

[51] F. Juretic, *Error Analysis in Finite Volume CFD*, Ph.D. thesis, Imperial College London (2004).

[52] T. Brandt, *Doctoral Dissertation*, Ph.D. thesis, Helsinki University of Technology Department (2007).

[53] M. J. Berger and J. Oliger, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations,* Journal of Computational Physics **53**, 484 (1984).

[54] S. M. Mitran, *A Comparison of Adaptive Mesh Refinement Approaches for Large Eddy Simulation,* in *Third AFOSR International Conference on DNS/LES* (Arlington, TX, 2001).

[55] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid* (Elsevier, 2000).

[56] A. Syrakos and A. Goulas, *Estimate of the truncation error of finite volume discretization of the Navier-Stokes equations on colocated grids,* International Journal for Numerical Methods in Fluids **50**, 103 (2006).

[57] F. Fraysse, E. Valero, and J. Ponsín, *Comparison of Mesh Adaptation Using the Adjoint Methodology and Truncation Error Estimates,* AIAA Journal **50**, 1920 (2012).

[58] A. Syrakos, G. Efthimiou, J. G. Bartzis, and A. Goulas, *Numerical experiments on the efficiency of local grid refinement based on truncation error estimates,* Journal of Computational Physics **231**, 6725 (2012).

[59] N. Currier and K. Franko, *A Discrete Error Transport Equation Source Model for Mesh Adaptation,* in *52nd Aerospace Sciences Meeting* (American Institute of Aeronautics and Astronautics, Reston, Virginia, 2014) pp. 1–11.

[60] K. Ding, *Efficient Output-Based Adaptation Mechanics for High-Order Computational Fluid Dynamics Methods*, Ph.D. thesis, University of Michigan (2018).

[61] W. C. Tyson, K. Swirydowicz, J. M. Derlaga, C. J. Roy, and E. de Sturler, *Improved Functional-Based Error Estimation and Adaptation without Adjoints,* in *46th AIAA Fluid Dynamics Conference*, June (American Institute of Aeronautics and Astronautics, Reston, Virginia, 2016) pp. 1–18.

[62] N. A. Pierce and M. B. Giles, *Adjoint Recovery of Superconvergent Functionals from PDE Approximations,* SIAM Review **42**, 247 (2000).

[63] N. Pierce and M. Giles, *Adjoint and Defect Error Bounding and Correction for Functional Estimates,* in *16th AIAA Computational Fluid Dynamics Conference* (American Institute of Aeronautics and Astronautics, Reston, Virigina, 2003).

[64] K. J. Fidkowski and P. L. Roe, *An Entropy Adjoint Approach to Mesh Refinement,* SIAM Journal on Scientific Computing **32**, 1261 (2010).

[65] R. Dwight, *Goal-Oriented Mesh Adaptation using a Dissipation-Based Error Indicator,* in *18th AIAA Computational Fluid Dynamics Conference*, June (American Institute of Aeronautics and Astronautics, Reston, Virigina, 2007) pp. 25–28.

[66] R. P. Dwight, *Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation,* Journal of Computational Physics **227**, 2845 (2008).

[67] C. de Boor, *Good Approximation by Splines with Variable Knots,* in *Spline Functions and Approximation Theory* (Edmonton, 1972) pp. 57–72.

[68] K. J. Fidkowski and D. L. Darmofal, *Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics,* AIAA Journal **49**, 673 (2011).

[69] M. Nemec, M. Aftosmis, and M. Wintzer, *Adjoint-Based Adaptive Mesh Refinement for Complex Geometries,* in *46th AIAA Aerospace Sciences Meeting and Exhibit*, January (American Institute of Aeronautics and Astronautics, Reston, Virigina, 2008) pp. 1–23.

[70] S. Hickel, N. A. Adams, and J. A. Domaradzki, *An adaptive local deconvolution method for implicit LES,* Journal of Computational Physics **213**, 413 (2006).

[71] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes,* Mathematics of Computation of the American Mathematical Society **67**, 73 (1998).

[72] H. A. van der Vorst, *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems,* SIAM Journal on Scientific and Statistical Computing **13**, 631 (1992).

[73] H. L. Stone, *Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations,* SIAM Journal on Numerical Analysis **5**, 530 (1968).

[74] S. Hickel, C. P. Egerer, and J. Larsson, *Subgrid-scale modeling for implicit large eddy simulation of compressible flows and shock-turbulence interaction,* Physics of Fluids **26**, 106101 (2014).

[75] U. Fey, M. König, and H. Eckelmann, *A new Strouhal–Reynolds-number relationship for the circular cylinder in the range* $47 < Re < 2x10^5$, Physics of Fluids **10**, 1547 (1998), https://doi.org/10.1063/1.869675 .

[76] M. Dröge and R. Verstappen, *A new symmetry-preserving cartesian-grid method for computing flow past arbitrarily shaped objects,* International Journal for Numerical Methods in Fluids **47**, 979 (2005), https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.924 .

[77] J. F. Gómez, *Multi-fidelity co-kriging optimization using hybrid injected RANS and LES,* (2018).

[78] S. Hickel, *Implicit Turbulence Modeling for Large-Eddy Simulation*, Ph.D. thesis, Technische Universität München (2008).

[79] ERCOFTAC, *The ERCOFTAC Knowledge Base Wiki Home Page - UFR 3-30 Test Case,* (2017), `https://www.kbwiki.ercoftac.org/w/index.php?title=UFR_3-30_Test_Case`, Accessed: 15.12.2019.