H-Adaptive Multiscale Schemes for the Compressible Navier-Stokes Equations – Polyhedral Discretization, Data Compression and Mesh Generation

F. Bramkamp [1], B. Gottschlich-Müller [2], M. Hesse[1], Ph. Lamby [2], S. Müller [2], J. Ballmann[1], K.H. Brakhage[2], W. Dahmen [2]
[1] Lehr- und Forschungsgebiet für Mechanik, RWTH Aachen, Germany
[2] Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Germany

Summary

In this paper we present the main conceptual ingredients and the current state of development of the new solver QUADFLOW for large scale simulations of compressible fluid flow and fluid–structure interaction. In order to keep the size of the discrete problems at every stage as small as possible for any given target accuracy, we employ a multiresolution adaptation strategy that will be described in the first part of the paper. In the second part we outline a new mesh generation concept that is to support the adaptive concepts as well as possible. A key idea is to understand meshes as parametric mappings determined by possibly few control points as opposed to store each mesh cell separately. Finally, we present a finite volume discretization along with suitable data structures which again is to support the adaptation concepts. We conclude with numerical examples of realistic applications demonstrating different features of the solver.

1 Introduction

The numerical simulation of non–stationary fluid–structure interactions poses enormous challenges to computing resources, data management strategies but also to the underlying mathematical concepts that contribute to keeping the computational complexity tractable. However, any increase in computing power is bound to be surpassed by the demands put forth by scientist and engineers through more and more realistic and consequently more complex models. Advancing the current frontiers in numerical simulation calls for combined efforts beyond traditional borders of scientific disciplines. Studying the interaction of aerodynamics and structural mechanics in complex geometries is a typical example. The fact that several severe obstructions such as time–dependency of the involved processes, varying complex geometries and the coupling of physical regimes with different characteristic features combined with the huge amount of data needed to model the process in an

adequate way, indicates the principal limitations of conventional approaches. In order to resolve a typically singular behavior of the solution in complex geometries meshes with several millions of cells are required. Improved hardware or purely data oriented strategies such as parallel computing are not sufficient to overcome the arising difficulties. As important and necessary these aspects may be they have to be complemented in the long run by mathematical concepts that aim at minimizing in the first place the complexity and size of arising discrete problems.

This paper summarizes some recent attempts in this direction. It describes the present stage of an integrated development of dynamic adaptation strategies, mesh generation and discretization which is presented here for a still intermediate state of two–dimensional gas flow. The central objective is to realize adaptively generated discretizations that are able to resolve the physically relevant phenomena at the expense of possibly few degrees of freedom and correspondingly reduced storage demands. This requires a careful coordination of the core ingredients namely the discretization of the underlying system of partial differential equations, the generation and management of suitable meshes and the adaptation mechanisms, all three parts being based on suitable data structures. The following remarks are to indicate the main orientation. Accepting the Navier Stokes equations as the model of choice we give preference to quadrilateral and hexahedral meshes that still facilitate best boundary fitted anisotropic meshes. To retain sufficient geometric flexibility this is combined with block structuring. Complex geometry as well as the anticipated fine scale behavior of the solutions require extremely high resolution at least in parts of the computational domain.

To avoid storage demands which, in the light of the envisaged applications are prohibitive, we wish to employ local mesh refinement. A pivotal role is therefore played by reliable and efficient refinement strategies that are presented in Section 3.1. The main distinction from previous work in this regard lies in the fact that we employ here recent *multiresolution techniques*. The starting point is a proposal by Harten to transform the arrays of cell averages associated with any given finite volume discretization into a different format that reveals insight into the local behavior of the solution. The cell averages on a given highest level of resolution are represented as cell averages on some coarse level where the fine scale information is encoded in arrays of *detail coefficients* of ascending resolution. This requires a *hierarchy of meshes*. In contrast to Harten's idea the multiscale representation is not only used to avoid expensive flux evaluations in regions where the solution is smooth but to create *locally refined meshes*. As long as one works on a uniform mesh (even when in a major part inexpensive finite differences are employed) the computational complexity stays proportional to the number of cells which in 3d calculations with the above objectives is prohibitive. Thus a principal objective is to extract the inherent complexity of the problem by placing as few degrees of freedom so as to still capture the features of the searched for solution within a given tolerance. A central mathematical problem is then to show that the essential information to be propagated in time is still kept with sufficient accuracy when working on locally coarser meshes.

The adaptation strategy gives rise to locally refined meshes of quadtree respectively octree type. The second important ingredient is the generation of such meshes along with the information needed by the flow solver at any stage of a dynamical calculation. A key idea is to represent such meshes with as few parameters as possible while further successive refinements can be efficiently computed based on the knowledge of these parameters. This seems to be of vital importance with regard to (geometrically) non–stationary processes. Roughly speaking the mesh in each block results from evaluating a parametric mapping from the computational domain into the physical domain. Such mappings can be based on B-spline representations in combination with well established concepts from CAGD (computer aided geometric design). The quantities to be updated in time are the relatively few control parameters in those parametric representations, while mesh points on any level of resolution can be efficiently computed due to the locality of the B-spline representation. The fact that one needs indeed only relatively few control points in order to generate meshes of good quality is partly due to the variation diminishing property of B-splines. For this statement we refer to the discussion later in Section 4 of this paper.

Finally, Section 5 is devoted to the outline of a discretization scheme that meets the requirements of the adaptation concept and fits well with the mesh generation. To avoid complicated mesh management within each block and to keep the discretizations of the individual blocks as independent as possible and, in particular, to avoid global geometrical constraints, we insist on meshes with hanging nodes. Since the discretization is based on the finite volume concept this appeared to us as the best compromise. This requires the development of a finite volume scheme for fairly general cell partitions that can cope, in particular, with hanging nodes and possible unstructured parts in complicated regions of the flow domain. Section 5 offers a self contained account of the discretization scheme including data structures, realization of spatial second order, the treatment of convective and viscous fluxes, choice of limiters, the treatment of boundary conditions and the validation of the basic scheme by a proper selection of test cases.

We conclude with several applications to well–known numerical and fluid dynamical test cases that highlight the features of the whole flow solver.

2 Governing Equations

In the present study, laminar viscous fluid flow is described by the time dependent Navier–Stokes equations for a compressible gas. Neglecting body forces and volume supply of energy, the conservation laws for any control volume Ω with boundary $\partial\Omega$ and outward unit normal vector $\mathbf{n}$ on the surface element $dS \subset \partial\Omega$ can be written in integral form as:

$$\int_{\Omega} \frac{\partial \mathbf{u}}{\partial t} dV + \oint_{\partial\Omega} \left(\mathbf{F}^c(\mathbf{u}) + \mathbf{F}^d(\mathbf{u}) \right) \mathbf{n}\, dS = \mathbf{0}\,. \tag{1}$$

To complete the posed problem initial values $\mathbf{u}\,(\mathbf{x}, t_0) = \mathbf{u}_0\,(\mathbf{x})$, $\mathbf{x} \in \Omega$ and boundary conditions $\mathbf{u}\,(\mathbf{x}, t)|_{\partial\Omega} = \mathcal{B}\,(\mathbf{x}, t)$, $\mathbf{x} \in \partial\Omega$ are to be prescribed.
$\mathbf{u} = (\varrho,\ \varrho\mathbf{v},\ \varrho e_{tot})^T$ denotes the vector of the unknown conserved quantities. $\mathbf{F}^c$ and $\mathbf{F}^d$ represent the physically conservative flux and the diffusive flux function, respectively. The physically conservative flux $\mathbf{F}^c$ is

$$\mathbf{F}^c = \begin{pmatrix} \varrho\mathbf{v} \\ \varrho\mathbf{v} \circ \mathbf{v} + p\,\mathcal{I} \\ \varrho h_{tot}\mathbf{v} \end{pmatrix} \tag{2}$$

where ϱ denotes the density, p the static pressure, $\mathbf{v}$ the velocity vector, e_{tot} the total energy and h_{tot} the total enthalpy. The symbol $\circ$ means the dyadic product.

The diffusive flux $\mathbf{F}^d$ contains the viscous stresses and heat conduction

$$\mathbf{F}^d = \begin{pmatrix} 0 \\ -\mathcal{T}^v \\ -\mathbf{v}\mathcal{T}^v + \mathbf{q} \end{pmatrix} \tag{3}$$

where the viscous stress tensor $\mathcal{T}^v$ for an isentropic Newtonian fluid is defined as

$$\mathcal{T}^v = \mu \left(grad\,\mathbf{v} + (grad\,\mathbf{v})^T \right) - \frac{2}{3}\mu\,(div\,\mathbf{v})\,\mathcal{I}\,. \tag{4}$$

Heat conduction is modeled by Fourier's law

$$\mathbf{q} = -\kappa\, grad T \tag{5}$$

where the thermal conductivity is assumed as

$$\kappa = c_p \frac{\mu}{Pr} \tag{6}$$

with Prandtl number $Pr = 0.72$. The variation of the molecular viscosity μ as a function of temperature is determined by the Sutherland formula

$$\mu = \mu_\infty \left(\frac{T}{T_\infty} \right)^{3/2} \frac{T_\infty + S}{T + S}\,, \tag{7}$$

where $S = 110° K$ denotes the Sutherland constant. The static pressure is related to the specific internal energy according to the equation of state for a perfect gas

$$p = \varrho\,(\gamma - 1)\left(e_{tot} - 1/2\,|\mathbf{v}|^2\right), \tag{8}$$

where γ is the ratio of specific heats, which is taken as 1.4 for air.

3 Adaptation Based on Multiscale Analysis

3.1 Problem Formulation and Preliminaries

The numerical schemes under consideration are to be applied to gas flow. Although the full Navier Stokes equations serve as the main model the behavior of the numerical scheme is largely determined by corresponding hyperbolic conservation laws describing the balance of quantities like mass, momentum and energy ignoring heat conduction and viscous effects. We will therefore confine the discussion first to this setting in order to bring out the basics of the concept. The same techniques carry over to viscous fluxes and source terms, see [15]. Omitting the upper index c in the conservative flux and denoting by $\Omega \subset \mathbb{R}^d$ the *computational domain* and by $\mathcal{D} \subset \mathbb{R}^m$ the *space of admissible states* hosting the *conserved quantities* $\boldsymbol{u} : [0, \overline{t}] \times \Omega \to \mathcal{D}$, the basic balance laws have the form

$$\int_V \frac{\partial\, \boldsymbol{u}(t,\boldsymbol{x})}{\partial\, t}\, dV = -\int_{\partial V} \boldsymbol{F}(\boldsymbol{u}(t,\boldsymbol{x}))\, \boldsymbol{n}\, dS, \tag{9}$$

where V is an arbitrary but temporally fixed control volume (with sufficiently smooth boundary such that the Green–Gauss theorem works). Under the assumption that $\boldsymbol{u}$ is smooth one derives from (9) the usual first order system of conservation laws of the form

$$\frac{\partial\, \boldsymbol{u}(t,\boldsymbol{x})}{\partial\, t} + \sum\nolimits_{i=1}^{d} \frac{\partial\, \boldsymbol{F}_i(\boldsymbol{u}(t,\boldsymbol{x}))}{\partial\, x_i} = \mathbf{0}, \tag{10}$$

which describes the temporal evolution of $\boldsymbol{u}$. Throughout this work the fluxes are assumed to be smooth in the state space, i.e., $\boldsymbol{F}_i \in C^2(\mathcal{D}, \mathbb{R}^m)$ and the Jacobian of the normal flux $\boldsymbol{F}(\boldsymbol{u})\,\boldsymbol{n}$ has m real eigenvalues $\lambda_i(\boldsymbol{u}, \boldsymbol{n})$ as well as a complete system of m linearly independent right eigenvectors $\boldsymbol{r}_i(\boldsymbol{u}, \boldsymbol{n})$ and left eigenvectors $\boldsymbol{l}_i(\boldsymbol{u}, \boldsymbol{n})$, respectively, $i = 1, \ldots, m$, for all $\boldsymbol{u} \in \mathcal{D}$ and $\boldsymbol{n} \in \mathbb{R}^d$ with $\|\boldsymbol{n}\|_2 = 1$. The system (10) is then called *hyperbolic*.

As mentioned in Section 2, the solution of the conservation laws is subject to initial conditions and also to appropriate boundary conditions (see Section 5.1.6 for details). Since the choice of the latter conditions does not affect the discussion of adaptive concepts we will suppress this issue (which by itself is certainly a delicate one) and confine the discussion for the moment to pure initial value problems.

The system (10) is to be viewed as a model which has to be properly interpreted in order to cover all physically relevant features such as shocks and contact discontinuities. The common basis is provided by the notion of weak solutions of (10), which solve the variational problem obtained when multiplying (10) by smooth test functions and applying integration by parts. The quantities $\boldsymbol{u}$ are then no longer subjected to derivatives so that certain discontinuities are admissible. By properly varying the test functions one ultimately also recovers (9) which is therefore a natural starting point for discretizations.

Before explaining this one should note though that when passing to weak solutions uniqueness is generally lost even when proper side constraints are imposed. One has to resort to additional selection criteria usually referred to as *entropy conditions* (motivated by the thermodynamical entropy) that restore uniqueness see [21, 20]. This is well understood for scalar conservation laws and summed up by Kruzhkov's entropy concept. The situation for systems of conservation laws is by far less clear and several entropy concepts compete which are generally not equivalent, see for instance [10].

The presence of discontinuities does not only cause problems concerning existence and uniqueness but also crucially effects the design of numerical schemes. For instance, standard finite difference discretizations of (10) will produce oscillations near discontinuities. These schemes have to be stabilized by artificial viscosity which causes a significant loss in accuracy. In particular, discontinuities are smeared. Moreover, stability in the sense that the entropy solution is recovered by a numerical scheme severely limits its order of exactness. Again the design of such schemes is not the issue of the following discussion. We will rather assume at this point that we have a stable scheme at hand to explain then how to develop adaptive refinements based on such a given discretization. Concrete realizations that fit our particular application context best will be developed in Section 5. Therefore we content ourselves with describing the following necessary ingredients.

Recalling (9), a natural choice for the discretization of conservation laws are finite volume schemes. These schemes arise when observing that (9) describes the evolution of *cell averages*. In fact, integration over a time interval yields

$$\hat{\boldsymbol{u}}(t+\tau) = \hat{\boldsymbol{u}}(t) + \frac{\tau}{|V|}\,\boldsymbol{B}(t) \tag{11}$$

where the cell average $\hat{\boldsymbol{u}}(t)$ and the flux balance $\boldsymbol{B}(t)$ defined by

$$\hat{\boldsymbol{u}}(t) := \frac{1}{|V|}\int_V \boldsymbol{u}(t,\boldsymbol{x})\,dV, \quad \boldsymbol{B}(t) := \frac{1}{\tau}\int_t^{t+\tau}\int_{\partial V} \boldsymbol{F}(\boldsymbol{u})\,\boldsymbol{n}\,dS\,dt \tag{12}$$

refer to an arbitrary but fixed cell $V \subset \Omega$ with volume $|V| := \int_V 1\,dV$. Obviously, the cell average changes in time due to the flux balance, i.e., the information that is crossing the cell interface ∂V in the time interval $[t, t+\tau]$, see Figure 1. Approximating the flux balance $\boldsymbol{B}(t)$ then leads to a finite volume scheme.

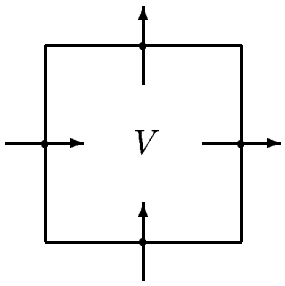


Figure 1 Transport of information across cell interfaces

3.2 The Objectives

In the past, finite volume schemes have been applied not only to academic problems but also to real world problems arising, for instance, in engineering. In this context robustness has been a major issue. However, in order to guarantee the reliability of the computational results the discretization error has to be controlled as well. So far, error estimates are available only for restricted classes of numerical schemes applied to a single conservation law, see [5, 6]. The discretization error can then be shown to be bounded by a constant times h^α, $\alpha \leq 0.5$, where h denotes the largest cell diameter of the underlying mesh. Due to the lack of global smoothness, we cannot expect a convergence rate better than $\alpha = 1$ even for schemes that are higher–order accurate in regions where the solution is locally smooth. This is a severe drawback whenever a prescribed error tolerance is to be met. In fact, accuracy considerations may become increasingly important for nonstationary calculations or when the interaction of fluid flow and structure is to be simulated. In such cases one encounters relevant physical effects over a large range of length scales and different propagation speeds in the coupled regimes. Moreover, the presence of discontinuities, boundary layers, and other singular phenomena requires small mesh sizes at least in parts of the computational domain. In order to resolve these effects adequately by discretizations based on *quasi-uniform meshes* would require a prohibitively large number of cells. In particular, in 3D calculations the resulting necessary memory resources and computation time would by far exceed the capacity of modern computing facilities, e.g. parallel architectures or clusters. In order to make these numerical simulations feasible new algorithmic concepts have to be developed that reduce the computational complexity, i.e., computation time and memory requirements, to a degree that reflects the problem inherent degrees of freedom relative to a desired accuracy. This calls for concepts that automatically adapt the local resolution of a discretization to the local behavior of the solution. This means only in regions where small scale physical effects are present the discretization has to exhibit high resolution while elsewhere the mesh is to be kept as coarse as possible for still meeting given accuracy requirements. This is exactly the objective of adaptive schemes that aim at distributing the degrees of freedom as economically as possible based on current information obtained during the calculation. There are two major issues to be ad-

dressed in this context. First, the decisions made during the calculations have to be reliable, i.e., one has to be sure that no essential features are missed when keeping the mesh size moderate in certain areas, in particular, with regard to non–stationary processes. This task concerns mathematical analysis. Second, handling dynamic mesh adaptation requires the development of much more sophisticated data structures.

In the literature, several adaptive strategies have been discussed or are under current investigation. A standard strategy is to base local mesh refinements on *local indicators* which are typically related to strong gradients in a current approximation. However, no reliable error control is offered by this concept. For this purpose, *a posteriori estimates* have been derived which aim at equilibrating local errors. So far, this type of error estimators are only available for scalar problems, see [19]. Here we will pursue a different strategy which originated from earlier proposals by Harten [18]. The key idea is to transform the arrays of cell averages produced by a given finite volume scheme into a different multiscale format. In this format only very few coefficients represent cell averages for a possibly coarse partition while the other coefficients encode the difference information corresponding to the intermediate length scales that is lost through the coarsening. Quite in the spirit of a *multiscale analysis* the size of these *detail coefficients* will be used as a basis for local mesh refinements. This gives rise to an adaptive mesh with respect to which the evolution step is performed. Of course, the crux in this context is to arrange this procedure in a way that at no stage of the computation there is ever made use of fully refined uniform mesh.

In the remainder of this section, we explain the main ingredients of such a multiscale analysis of an array of cell averages from which an adaptive mesh is derived. Then we outline the construction of the adaptive finite volume scheme and comment on the discretization error. Finally, some numerical results for model problems are presented, that give a first idea of the quantitative performance of the adaptation concept.

3.3 Multiscale Setting

Finite volume schemes are naturally related to cell averages of the solution. In order to detect singularities of the solution, we transform the array of cell averages into a new data format which allows for data compression.

3.3.1 Grid Hierarchy

The starting point for the multiscale setting is a hierarchy of nested grids $\mathcal{G}_j := \{V_{j,k}\}_{k\in I_j}$, $j = 0, \ldots, L$, such that each grid $\mathcal{G}_j$ is a *partition* of the computational domain, i.e., $\Omega = \bigcup_{k\in I_j} V_{j,k}$. Here the coarsest discretization is indicated by 0 and the finest discretization by L. Moreover, the grids are *nested* in the sense that the cells on the coarse grid can be represented as the union of cells on the next finer

grid, i.e.,

$$V_{j,k} = \bigcup_{r \in \mathcal{M}^0_{j,k}} V_{j+1,r}, \quad k \in I_j. \tag{13}$$

The index set $\mathcal{M}^0_{j,k} \subset I_{j+1}$ corresponds to the cells on level $j+1$ resulting from the refinement of the cell $V_{j,k}$. A simple example is shown in Figure 2. We em-

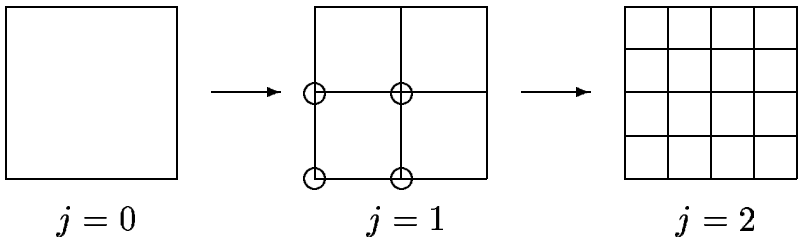


Figure 2 Sequence of nested grids

phasize that the framework presented here can also be applied to unstructured grids and irregular mesh refinements. To keep the presentation as simple as possible we confine, however, the discussion to refinements of cells into a constant number of children, i.e.,

$$\# \mathcal{M}^0_{j,k} = M_r = const.$$

In the above example, $\mathcal{M}^0_{j,k}$ can be identified with the circled nodes in the middle part of Figure 2. With each cell in $\mathcal{G}_j$ we associate the *box function*

$$\tilde{\varphi}_{j,k}(\boldsymbol{x}) := \frac{1}{|V_{j,k}|} \chi_{V_{j,k}}(\boldsymbol{x}), \tag{14}$$

where the characteristic function χ_V corresponding to any set V takes the value one for $\boldsymbol{x} \in V$ and zero elsewhere. Then the cell average of a scalar, integrable function u can be expressed as

$$\hat{u}_{j,k} = \langle u, \tilde{\varphi}_{j,k} \rangle_\Omega, \tag{15}$$

where the scalar product is defined by $\langle u, v \rangle_\Omega := \int_\Omega u\, v\, d\boldsymbol{x}$. Note, that our framework is not restricted to scalar functions but can directly be applied to the components of vector–valued functions. Since the grids are nested, we infer the two–scale relation

$$\tilde{\varphi}_{j,k} = \sum_{r \in \mathcal{M}^0_{j,k}} m^{j,0}_{r,k}\, \tilde{\varphi}_{j+1,r}, \qquad m^{j,0}_{r,k} := \frac{|V_{j+1,r}|}{|V_{j,k}|}, \tag{16}$$

i.e., the coarse grid box function can be represented as a linear combination of the corresponding fine grid box functions. Consequently, the averages of two discretization levels are related to each other by

$$\hat{u}_{j,k} = \sum_{r \in \mathcal{M}^0_{j,k}} m^{j,0}_{r,k}\, \hat{u}_{j+1,r}. \tag{17}$$

Now the goal is to transform these data into a different format of cell averages corresponding to a sequence of resolution levels. This will be motivated by a simple univariate example.

3.3.2 A Univariate Example

We now consider the unit interval $\Omega = [0,1]$ where the grid hierarchy is determined by a uniform dyadic partition of $[0,1]$, i.e., $V_{j,k} = 2^{-j}[k,k+1]$, $k \in I_j := \{0,\dots,2^j-1\}$. Hence, the refinement sets are just given by $\mathcal{M}^0_{j,k} = \{2k, 2k+1\} \subset I_{j+1}$, $k \in I_j$. Then the box function has the form

$$\tilde{\varphi}_{j,k} = 2^j \chi_{[0,1]}(2^j \cdot -k),$$

and the two–scale relation (16) reads here

$$\tilde{\varphi}_{j,k} = \frac{1}{2}\,(\tilde{\varphi}_{j+1,2k} + \tilde{\varphi}_{j+1,2k+1}). \tag{18}$$

We explain now how to decompose the cell averages into averages of a coarser partition and details. To this end, we introduce the box wavelet (which in this case is better known as Haar wavelet)

$$\tilde{\psi}_{j,k} := \frac{1}{2}\,(\tilde{\varphi}_{j+1,2k} - \tilde{\varphi}_{j+1,2k+1}). \tag{19}$$

Then we can write any fine scale box function by means of the coarse box function $\tilde{\varphi}_{j,k}$ and the box wavelet $\tilde{\psi}_{j,k}$.

$$\tilde{\varphi}_{j+1,2k} = \tilde{\varphi}_{j,k} + \tilde{\psi}_{j,k}, \qquad \tilde{\varphi}_{j+1,2k+1} = \tilde{\varphi}_{j,k} - \tilde{\psi}_{j,k}. \tag{20}$$

These relations are motivated by the illustrations in Figure 3. In analogy to (17) we

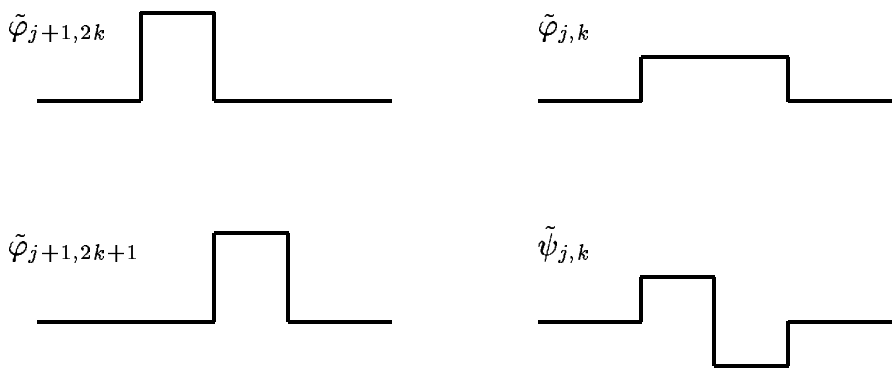


Figure 3 Box function and box wavelet

deduce from (18) the two–scale relations for the cell averages

$$\hat{u}_{j,k} = \frac{1}{2}\,(\hat{u}_{j+1,2k} + \hat{u}_{j+1,2k+1}).$$

On the other hand, (20) implies

$$\hat{u}_{j+1,2k} = \hat{u}_{j,k} + d_{j,k}, \qquad \hat{u}_{j+1,2k+1} = \hat{u}_{j,k} - d_{j,k}, \tag{21}$$

where the details are defined by

$$d_{j,k} := \langle u, \tilde{\psi}_{j,k}\rangle_{[0,1]} = \frac{1}{2}\,(\hat{u}_{j+1,2k} - \hat{u}_{j+1,2k+1}). \tag{22}$$

These relations show how to express fine scale averages in terms of coarse scale ones plus details and vice versa.

The above manipulations concern cell averages of some searched underlying function u. One might view these cell averages just as piecewise constant *approximations* to this function. Such approximations are, however, confined as piecewise constants to be at best of first order accurate. So this view would not carry very far. Instead one should view the cell averages just as *linear functionals* of u from which more accurate reconstructions of u could be derived. So there is a natural *duality* between the space hosting the cell averages and its dual where u itself lives. In order to make this distinction clear and to pave the way for later modifications, we have to introduce another system of functions

$$\varphi_{j,k} := 2^{-j}\,\tilde{\varphi}_{j,k} = \chi_{[0,1]}(2^j \cdot - k), \qquad \psi_{j,k} := 2^{-j}\,\tilde{\psi}_{j,k},$$

in which one should think of u being represented. At this point it differs from the earlier version only by normalizing factors but it will later be seen to be conceptually important to distinguish these systems. In fact, the $\varphi_{j,k}$ are the L_∞–normalized counterparts of the L_1-normalized box function and box wavelet, respectively. We will therefore refer to this system of functions as *primal system* (although we have actually started with the dual one). In particular, the dual system $\{\tilde{\varphi}_{j,k}\}_{k\in I_j} \cup \{\tilde{\psi}_{j,k}\}_{k\in I_j}$ is *biorthogonal* to the primal system $\{\varphi_{j,k}\}_{k\in I_j} \cup \{\psi_{j,k}\}_{k\in I_j}$, i.e.,

$$\begin{aligned} \langle \varphi_{j,k}, \tilde{\varphi}_{j,l}\rangle_{[0,1]} &= \langle \psi_{j,k}, \tilde{\psi}_{j,l}\rangle_{[0,1]} = \delta_{k,l}, \\ \langle \varphi_{j,k}, \tilde{\psi}_{j,l}\rangle_{[0,1]} &= \langle \psi_{j,k}, \tilde{\varphi}_{j,l}\rangle_{[0,1]} = 0. \end{aligned} \tag{23}$$

Of course, these functions satisfy two-scale relations analogous to (19) and (20) with slightly different normalizing factors.

It should already be noted now that one ultimately works only on the dual side while the primal system plays a conceptual role which is, however, important for the analysis and helpful for understanding later developments. To this end, note that given u, the function

$$u_j := \sum_{k\in I_j} \langle u, \tilde{\varphi}_{j,k}\rangle_{[0,1]}\,\varphi_{j,k} \tag{24}$$

is a projection of u onto the span of the level j functions $\varphi_{j,k}$ (which at this point happen to be still piecewise constants as well), where the expansion coefficients are, in view of (15), the cell averages of u with respect to a mesh of level j. It represents an approximate solution of the conservation law on a fixed time level corresponding to the discretization level j.

Note, that if u were (nearly) constant all coefficients in (24) would take (nearly) the same value and hence would be all equally significant. This is certainly not a very economical representation of a simple object such as a constant. We wish to transform the coefficients, namely the cell averages $\langle u, \tilde{\varphi}_{j,k}\rangle_{[0,1]}$ therefore into a format where such (near) redundancy becomes readily apparent. In fact, invoking the two-scale relation (19), (20) in combination with biorthogonality (23), u_j can be rewritten as

$$u_j = \sum_{k\in I_{j-1}} \langle u, \tilde{\varphi}_{j-1,k}\rangle_{[0,1]}\, \varphi_{j-1,k} + \sum_{k\in I_{j-1}} \langle u, \tilde{\psi}_{j-1,k}\rangle_{[0,1]}\, \psi_{j-1,k}. \tag{25}$$

Continuing in the same way with the first sum on the right hand side, eventually leads to a representation of u_j in terms of a coarse level cell averages complemented by a sum over all levels $l < j$ containing *detail coefficients* of the form $d_{l,k} := \langle u, \tilde{\psi}_{l,k}\rangle_{[0,1]}$. Clearly, raising the resolution level j would amount to adding further detail coefficients which explains their update character.

Next note that, in view of (22), the details $d_{j,k}$ vanish whenever the function u is a constant, i.e.,

$$\langle 1, \tilde{\psi}_{j,k}\rangle_{[0,1]} = 0. \tag{26}$$

Moreover, writing u on the support of $\tilde{\psi}_{j,k}$ as a constant plus a fluctuation (first order Taylor expansion) we see that $|d_{j,k}|$ is still small if that fluctuation is small. In fact, one easily verifies that

$$\begin{aligned} |d_{j,k}| &= |\langle u, \tilde{\psi}_{j,k}\rangle_{[0,1]}| = \inf_c |\langle u - c, \tilde{\psi}_{j,k}\rangle_{[0,1]}| \\ &\le \inf_c \|u - c\|_{L_\infty(\operatorname{supp}\tilde{\psi}_{j,k})} \|\tilde{\psi}_{j,k}\|_{L_1(\operatorname{supp}\tilde{\psi}_{j,k})} \le 2^{-j}\|u'\|_{L_\infty(\operatorname{supp}\tilde{\psi}_{j,k})} \end{aligned}$$

Hence, the details may become small in smooth regions of u whereas they constitute significant contributions in (25) where u varies more strongly.

This suggests to neglect all sufficiently small details in order to keep only essential information on the function u thereby creating a *compressed (approximate) representation* of u. To pursue this line, however, at least two essential issues arise.

(i) For a compression to be really efficient one needs a more refined test than just annihilating constants as in (26) which only exploits first order smoothness. In fact, if one had instead of (26) $\langle p, \tilde{\psi}_{j,k}\rangle_{[0,1]} = 0$ for all polynomials p up to degree $m-1$ (order m) the same argument (now using higher order Taylor polynomials in place of the constant c above) would show that, when u has locally bounded mth order derivatives, the coefficients $d_{j,k}$ would be of the

order 2^{-jm} thereby exhibiting a much stronger decay for higher smoothness. (This can be achieved by means of higher order biorthogonal systems instead of piecewise constants for the *primal system*. In case of the unit interval we refer to [8, 13].) Note that this is the same as saying that the functions $\varphi_{j,k}$ in the projection (24) reproduces all polynomials of order m exactly, i.e., is of higher order. But it is important to note that we actually never have to realize this higher order primal system as long as we manage to come up with dual wavelets having a higher order of vanishing moments. It will be explained below how to realize such higher order vanishing moments in an even more realistic setting of boundary fitted meshes.

(ii) Even if one realizes higher order vanishing moments, discarding small coefficients $d_{j,k}$ resulting from the corresponding multiscale transformation raises the question how this perturbation of the coefficients affects the underlying function in the relevant norm. This is the issue of *stability* of the wavelet expansions, which ultimately does involve knowledge about the primal system. Thus the primal system is *not* needed for the algorithmic realization but is important for the analysis of the scheme.

3.3.3 Higher Order Biorthogonal Systems

We will address now the above issues (i) and (ii) for a more realistic setting of mesh hierarchies. The first step is to transmit the above univariate case constructing box wavelets for an arbitrary nested mesh hierarchy. The resulting wavelets will serve as a starting point for the construction of modified box wavelets with improved cancellation properties with the aid of a concept introduced in [3].

For the construction of the generalized box wavelets we now proceed analogously to the univariate case. As a counterpart to (19) we introduce the box wavelet as a linear combination of the fine scale box functions $\tilde{\varphi}_{j+1,r},\ r \in \mathcal{M}^0_{j,k}$, related to the refinement of the cell $V_{j,k}$,

$$\check{\psi}_{j,k,e} := \sum_{r \in \mathcal{M}^0_{j,k}} \check{m}^{j,e}_{r,k}\, \tilde{\varphi}_{j+1,r}, \quad e \in E^* := E \backslash \{0\}, \tag{27}$$

with $E := \{0, \ldots, M_r - 1\}$. The index e plays the same role as the distinction between even and odd indices earlier in (20). The (mask or filter) parameters $\check{m}^{j,e}_{r,k}$ are determined in such a way that the resulting system of box functions and box wavelets and their L_∞–normalized counterparts defined by

$$\varphi_{j,k} := |V_{j,k}|\, \tilde{\varphi}_{j,k}, \quad \psi_{j,k,e} := |V_{j,k}|\, \check{\psi}_{j,k,e}, \tag{28}$$

are biorthogonal in the sense of (23) and the box wavelets have one vanishing moment, i.e.,

$$\langle 1, \check{\psi}_{j,k,e} \rangle_\Omega = 0.$$

Moreover, one can then find in analogy to (20) parameters $\check{g}_{r,k}^{j,e}$, $r \in \mathcal{M}_{j,k}^0$, such that

$$\tilde{\varphi}_{j+1,k} = \sum_{e \in E} \sum_{r \in \mathcal{M}_{j,k}^0} \check{g}_{r,k}^{j,e} \, \check{\psi}_{j,r,e} \tag{29}$$

holds, where it is convenient to define $\check{\psi}_{j,k,0} := \tilde{\varphi}_{j,k}$. As in the univariate case (see (19), (20)), the two–scale relations (16), (27) and (29) allow one to realize a local change of basis between the systems formed by coarse scale functions $\varphi_{j,k}$ complemented by the oscillatory functions $\{\check{\psi}_{j,k,e}\}_{e \in E}$ and the fine scale functions $\{\varphi_{j+1,r}\}_{r \in \mathcal{M}_{j,k}^0}$. The corresponding details can be found in [23].

Introducing as in the univariate case the *details*

$$\check{d}_{j,k,e} := \langle u, \check{\psi}_{j,k,e} \rangle_\Omega, \qquad e \in E^*, \tag{30}$$

one then infers from (27) the two–scale relation

$$\check{d}_{j,k,e} = \sum_{r \in \mathcal{M}_{j,k}^0} \check{m}_{r,k}^{j,e} \, \hat{u}_{j+1,r}, \quad e \in E^*. \tag{31}$$

Note, that a similar relation holds for the cell averages, see equation (17).

Hence, in the same way as in the univariate case the resulting projection u_j of an integrable function u for the refinement level j

$$u_j := \sum_{k \in I_j} \langle u, \tilde{\varphi}_{j,k} \rangle_\Omega \, \varphi_{j,k} \tag{32}$$

can be split in analogy to (25) into a coarse scale part and a complementary detail part as

$$u_j = \sum_{k \in I_{j-1}} \langle u, \tilde{\varphi}_{j-1,k} \rangle_\Omega \, \varphi_{j-1,k} + \sum_{e \in E^*} \sum_{k \in I_{j-1}} \langle u, \check{\psi}_{j-1,k,e} \rangle_\Omega \, \psi_{j-1,k,e}. \tag{33}$$

Again the primal system consists still of piecewise constants. So by the very same arguments provided in the preceding section the decay of the detail coefficients obtained from a successive repetition of the above splitting into ever coarser parts and corresponding details would only be of order 2^{-j} on level j. As explained before, in order to get a better compression by exploiting higher order smoothness we have to raise the order of vanishing polynomial moments. The basic idea is to modify the box wavelet $\check{\psi}_{j,k,e}$ by some coarse grid box functions, leading to the ansatz

$$\tilde{\psi}_{j,k,e} := \check{\psi}_{j,k,e} + \sum_{l \in \mathcal{L}_{j,k}^e} l_{l,k}^{j,e} \, \tilde{\varphi}_{j,l}, \quad e \in E^*, \tag{34}$$

with parameters $l_{l,k}^{j,e}$ that are yet to be determined. Here the stencil $\mathcal{L}_{j,k}^e \subset I_j$ denotes the cells $V_{j,l}$ in the neighborhood of the cell $V_{j,k}$. Then the parameters $l_{l,k}^{j,e}$ are chosen such that

$$\langle p, \tilde{\psi}_{j,k,e} \rangle_\Omega = 0 \tag{35}$$

holds for all polynomials p of degree less than an arbitrary but fixed number M. The details of the construction can be found in [14, 23].

Note, that the modified wavelets $\tilde{\psi}_{j,k,e}$ still satisfy two–scale relations of the form (27) and (29). This can be seen by inserting (16) and (27) into (34), providing

$$\tilde{\psi}_{j,k,e} = \sum_{r \in \mathcal{M}^e_{j,k}} m^{j,e}_{r,k}\, \tilde{\varphi}_{j+1,r} \tag{36}$$

for some index set $\mathcal{M}^e_{j,k} \subset I_{j+1}$. Moreover, one can explicitly determine filter coefficients for reversing the two–scale relations (16) and (36) and express the fine scale functions through coarse scale functions and the wavelets $\tilde{\psi}_{j,k,e}$ by substituting (34) in (29), i.e.,

$$\tilde{\varphi}_{j+1,k} = \sum_{r \in \mathcal{G}^0_{j,k}} g^{j,0}_{r,k}\, \tilde{\varphi}_{j,r} + \sum_{e \in E^*} \sum_{r \in \mathcal{G}^e_{j,k}} g^{j,e}_{r,k}\, \tilde{\psi}_{j,r,e} \tag{37}$$

for some index set $\mathcal{G}^e_{j,k} \subset I_j$. Hence, the two–scale relations (16), (36) and (37) realize a change of basis between the systems $\{\tilde{\varphi}_{j,k}\}_{k \in I_j} \cup \{\tilde{\psi}_{j,k,e}\}_{k \in I_j, e \in E^*}$ and $\{\tilde{\varphi}_{j+1,k}\}_{k \in I_{j+1}}$.

The above manipulations are based upon the basic fact that if one forms a matrix $\check{M} = (\check{m}^{j,e}_{r,k})_{(k,e) \in E \times I_j, r \in I_{j+1}}$ the transpose of its inverse $\check{G}$ provides the filter coefficients for the biorthogonal primal system (here denoted by $\check{g}^{j,e}_{r,k}$). Therefore the above modifications of the wavelet filters $\check{m}^{j,e}_{r,k}$, $e \in E^*$ will change the $\check{G}$ to a new matrix G, where, however, now also the filter coefficients of the original scaled box function $\varphi_{j,k}$ have been modified. It should be emphasized that it is not clear a priori that such modifications will actually always produce a G such that now new modified functions $\varphi_{j,k}$ *exist* that satisfy the modified two scale relation. This will certainly depend on the choice of the coefficients $l^{j,e}_{l,k}$ in (34) and perhaps on the geometry of the cells. Of course, it is hoped that a modified system exists which will then actually have a corresponding higher accuracy providing better approximations to u on the primal side. That this is indeed the case can be shown under more restrictive assumptions on the mesh. For a general discussion of this issue and its relation to subdivision schemes see e.g. [11]. This is also closely related to the second issue (ii) in Section 3.3.2. As pointed out before the algorithmic realization relies only on the higher vanishing moments of the dual system and has so far never shown any indications that the primal system may fail to exist in any of the tests for realistic geometries.

3.3.4 Multiscale Transformation

As pointed out before, the reason for dealing with multiscale bases of the above type is merely to transform as in (28) an array of cell averages into a new data format that is more suitable to compression. Moreover, the degree of this compression as well as the analysis of the stability of the transformation in the sense of (ii) is still best

analyzed through such bases. Now the transformation of data readily results from the two–scale relations (16), (36) and (37). In the same fashion as the relations (21), (22) have been derived in the univariate case, one substitutes the two–scale relations for the basis functions in the definition of the cell averages $\hat{u}_{j,k} := \langle u, \tilde{\varphi}_{j,k} \rangle_\Omega$ and the details $d_{j,k,e} := \langle u, \tilde{\psi}_{j,k,e} \rangle_\Omega$ to obtain in view of (17),

$$\hat{u}_{j,k} = \sum_{r \in \mathcal{M}^0_{j,k}} m^{j,0}_{r,k}\, \hat{u}_{j+1,r}, \quad d_{j,k,e} = \sum_{r \in \mathcal{M}^e_{j,k}} m^{j,e}_{r,k}\, \hat{u}_{j+1,r} \tag{38}$$

and

$$\hat{u}_{j+1,k} = \sum_{r \in \mathcal{G}^0_{j,k}} g^{j,0}_{r,k}\, \hat{u}_{j,r} + \sum_{e \in E^*} \sum_{r \in \mathcal{G}^e_{j,k}} g^{j,e}_{r,k}\, d_{j,r,e}. \tag{39}$$

Denoting by $\hat{\boldsymbol{u}}_j := (\hat{u}_{j,k})_{k \in I_j}$ and $\boldsymbol{d}_j := (d_{j,k,e})_{k \in I_j, e \in E^*}$ the arrays of cell averages, respectively details on level j, the array $\hat{\boldsymbol{u}}_L$ of cell averages on the finest level L can be decomposed into a sequence of coarse scale averages $\hat{\boldsymbol{u}}_0$ and details $\boldsymbol{d}_j$, $j = 0, \ldots, L-1$. This is done by applying the two–scale relations (38) successively from fine to coarse levels, see Figure 4. This decomposition is called the *multiscale*

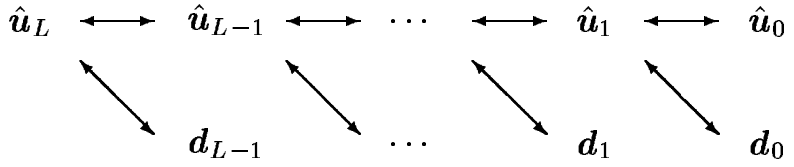


Figure 4 Pyramid scheme of multiscale transformation

transformation. It can be reversed by the *inverse multiscale transformation* based on the relations (39) ascending from coarser to finer levels. Note, that the multiscale transformation and its inverse require $\mathcal{O}(\# I_L)$ operations provided that the modified box wavelets are locally finite, i.e., the number of functions of level j that do not vanish in $\boldsymbol{x} \in \Omega$ is uniformly bounded. This is fulfilled whenever the stencils $\mathcal{L}^e_{j,k}$ are uniformly finite.

Recall that the multiscale transformation has been set up so as to realize a change of basis from a representation of the form (32) for ($j = L$ say) in single scale format into a representation with respect to the multiscale basis $\{\tilde{\varphi}_{0,k}\}_{k \in I_0} \cup \bigcup_{j=0}^{L-1} \{\tilde{\psi}_{j,k,e}\}_{k \in I_j, e \in E^*}$, i.e.,

$$u_L := \sum_{k \in I_L} \hat{u}_{L,k}\, \varphi_{L,k} = \sum_{k \in I_0} \hat{u}_{0,k}\, \varphi_{0,k} + \sum_{j=0}^{L-1} \sum_{k \in I_j} \sum_{e \in E^*} d_{j,k,e}\, \psi_{j,k,e}. \tag{40}$$

Whether small perturbations of the coefficients $d_{j,k,e}$ cause only small perturbations in u with respect to the L_∞-norm or of $\hat{\boldsymbol{u}}_L$ in a discrete L_1-norm relies on the *stability* of the multiscale basis which in turn depends on the existence of a regular primal basis, recall (ii) in Section 3.3.2, an issue that goes beyond the scope of this paper.

3.3.5 Local Grid Refinement

In the previous section we outlined the transformation of an array of cell averages corresponding to a finest uniform discretization level into a new data format. The new format is composed of cell averages on a coarsest discretization level and arrays of details describing the difference information between the cell averages on two successive discretization levels. At this point we assume that the underlying bases are stable. We now take advantage of the new representation exploiting the fact that details are guaranteed to decay at a certain rate depending on the local smoothness of the underlying function. The idea is simply to discard all coefficients $d_{j,k,e}$ in (40) that fall in absolute value below a certain threshold. For this purpose, we introduce the index set

$$\mathcal{D}_{L,\varepsilon} := \{(j,k,e)\ ;\ |d_{j,k,e}| > \varepsilon_j,\ e \in E^*,\ k \in I_j,\ j \in \{0,\ldots,L-1\}\}$$

corresponding to what will be referred to as *significant details*. Here $\varepsilon_j = 2^{j-L}\varepsilon$ is a level–dependent threshold value which is smaller on coarser levels.

It is important to distinguish this index set form an associated index set $\mathcal{G}_{L,\varepsilon}$ which determines an *adaptive grid* and is generated from $\mathcal{D}_{L,\varepsilon}$ as follows. First of all it selects cells $V_{j,k}$ from different levels in the grid hierarchy such that

$$\Omega = \bigcup_{(j,k)\in\mathcal{G}_{L,\varepsilon}} V_{j,k}.$$

The index set $\mathcal{G}_{L,\varepsilon}$ can be formed by traversing through the levels from coarse to fine. If there is a significant detail $d_{j,k,e} \in \mathcal{D}_{L,\varepsilon}$ for some $e \in E^*$ then the cell $V_{j,k}$ is locally refined according to (13), i.e., the index (j,k) is removed from $\mathcal{G}_{L,\varepsilon}$ and the refinement set $\mathcal{M}^0_{j,k}$ is added to $\mathcal{G}_{L,\varepsilon}$. Note, that the index set $\mathcal{G}_{L,\varepsilon}$ is initialized by all indices of the coarsest discretization, i.e., $(0,k) \in \mathcal{G}_{L,\varepsilon}$, $k \in I_0$. It is clear that the adaptive grid reflects local refinements and therefore involves *hanging nodes*, i.e., cell vertices of neighboring cells do not necessarily coincide, see Figure 5.

It is therefore important that the finite volume discretization employed in combination with the multiscale transforms copes well with hanging nodes.

We emphasize that this procedure only works provided that the index set of significant details correspond to a *graded tree*, the levels of neighboring cells differ at most by one. Since the set $\mathcal{D}_{L,\varepsilon}$ is in general not graded, we have to apply in addition

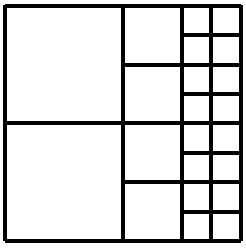


Figure 5 Locally refined grid with hanging nodes

a grading procedure to $\mathcal{D}_{L,\varepsilon}$. This will slightly inflate the index set but has so far been observed not to spoil the complexity reduction of floating point operations in any significant way. In fact, from the nature of singularities occurring in flow computations one expects the distribution of significant wavelet coefficients to exhibit at least nearly tree structure. If a high level wavelet overlaps a discontinuity this will be seen also by its coarser ancestors in the same region. For details on the grading procedure we refer to [23].

The actual time evolution will refer again to cell averages corresponding to the adaptive grid. Therefore it is important to interrelate the local cell averages and the significant details

$$\hat{\boldsymbol{u}}_{L,\varepsilon} := \{\hat{u}_{j,k}\}_{(j,k)\in\mathcal{G}_{L,\varepsilon}} \quad \boldsymbol{d}_{L,\varepsilon} := \{d_{j,k,e}\}_{(j,k,e)\in\mathcal{D}_{L,\varepsilon}}$$

corresponding to the projection of u_L on the *locally refined spaces* determined by the basis functions

$$\Phi_{L,\varepsilon} := \{\varphi_{j,k}\}_{(j,k)\in\mathcal{G}_{L,\varepsilon}}, \quad \Psi_{L,\varepsilon} := \{\varphi_{j,k}\}_{(j,k)\in I_0} \cup \{\psi_{j,k,e}\}_{(j,k,e)\in\mathcal{D}_{L,\varepsilon}},$$

respectively. Since the bases $\Phi_{L,\varepsilon}$ and $\Psi_{L,\varepsilon}$ span the same space, the projection u_L can equivalently be written as

$$u_L = \sum_{(j,k)\in\mathcal{G}_{L,\varepsilon}} \hat{u}_{j,k}\,\varphi_{j,k} = \sum_{k\in I_0} \hat{u}_{0,k}\,\varphi_{0,k} + \sum_{(j,k,e)\in\mathcal{D}_{L,\varepsilon}} d_{j,k,e}\,\psi_{j,k,e}. \tag{41}$$

The change of basis between the (locally) single–scale and the wavelet representation of u_L is realized by a *local* multiscale transformation analogously to (38) and (39), respectively. This transformation is to be realized with an optimal complexity, i.e., the number of operations should stay proportional to $\#\mathcal{D}_{L,\varepsilon}$. The algorithms of the local transformations are outlined in [23]. Here again, it turns out that feasibility of the algorithms is related to the gradedness of $\mathcal{D}_{L,\varepsilon}$. In particular, it is verified how the grading depends on the number of vanishing moments of the modified box wavelets.

3.4 Adaptive Finite Volume Schemes

We now outline the concept of adaptive finite volume schemes based on the above transformation concept. Of course, it will be crucial that at no stage of the algorithm

the full uniform grid of level L is visited. Therefore, when evolving an approximate solution on an adaptive grid at time n to time level $n+1$ one has to guess a possibly economic new mesh which of course may vary when dealing with non–stationary processes. Here it turns out that the construction of the numerical fluxes on a locally refined grid and the *prediction* of the significant details on the new time level are the core ingredients which crucially influence the performance as well as the reliability of the adaptive scheme.

First work on this subject has been reported by Harten [17, 18]. His primary objective, however, has been to accelerate a *given* finite volume scheme on a grid of essentially *uniform* resolution by a *hybrid* flux computation. As a core ingredient he also used a *multiscale decomposition* (although derived differently) similar to that in Section 3.3. It can be utilized to distinguish smooth regions of the flow field from regions with local strong variations in the solution. In particular, the hybrid flux evaluation can be controlled by the decomposition, i.e., expensive upwind discretizations are only applied near discontinuities of the solution. Elsewhere cheaper linear combinations of already computed numerical fluxes on coarser scales are used instead. These correspond to finite difference approximations. In the meantime Harten's originally one–dimensional concept has been extended to multidimensional problems on Cartesian grids [2, 4], curvilinear patches [12] and triangulations [25, 1, 7].

The bottleneck of Harten's strategy is the fact that the *computational complexity*, i.e., the number of floating point operations as well as the memory requirements can at best be reduced by a fixed factor independent of the size of the mesh. Thus computational work and storage requirements still grow at least proportionally to a globally finest uniform grid. For multidimensional applications this is a severe obstruction which is even enhanced by additional requirements concerning varying domains and coupling with structure calculations. Recently, a genuine *adaptive* approach has been presented in [16] and has been investigated in [9]. Here the computational complexity can be kept proportional to the problem–inherent degrees of freedom. A selfcontained account of the adaptive concept for conservation laws can be found in [23].

3.4.1 Construction

Essentially any standard explicit or implicit finite volume scheme for a system of conservation laws (capable of handling hanging nodes) may serve as starting point for the adaptive scheme. Suppose that for a given (highest) discretization level L it can be written in the form

$$\boldsymbol{v}_{L,k}^{n+1} + \theta\, \lambda_{L,k}\, \boldsymbol{B}_{L,k}^{n+1} = \boldsymbol{v}_{L,k}^{n} - (1-\theta)\, \lambda_{L,k}\, \boldsymbol{B}_{L,k}^{n} \tag{42}$$

with the ratio $\lambda_{L,k} := \tau/|V_{L,k}|$ and a parameter $\theta \in [0,1]$. Here a fixed time interval $[0,T]$ is decomposed by a temporal mesh $0 = t_0 < \ldots < t_N = T$. To keep the exposition simple and since we focus here on spatial adaptation the temporal partition is assumed to be uniform, i.e., $t_{n+1} = t_n + \tau$ for all $n = 0, \ldots, N-1$.

This is no constraint imposed by the adaptive strategy. The numerical *flux balance* is defined by

$$\boldsymbol{B}_{L,k}^n := \sum_l |\Gamma_{k,l}^L| \, \boldsymbol{F}_{k,l}^{L,n}$$

approximating the exact flux balance (12) corresponding to the control volume $[t_n, t_{n+1}] \times V_{L,k}$. Here $\Gamma_{k,l}^L$ denotes the interface of the cell $V_{L,k}$ to the neighbor cell $V_{L,l}$ and $\boldsymbol{F}_{k,l}^{L,n}$ the corresponding numerical flux, see Figure 6. Note, that the numerical fluxes depend on some cell averages of the *finest* discretization level for reasons to be discussed later. Furthermore the numerical fluxes are assumed to

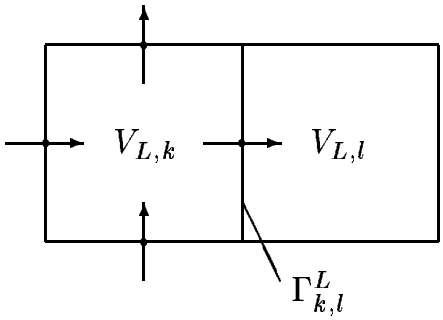


Figure 6 Finite volume scheme

be conservative, i.e.,

$$\boldsymbol{F}_{k,l}^{L,n} = -\boldsymbol{F}_{l,k}^{L,n}. \tag{43}$$

Then, for a given time level n, we introduce the cell averages $\boldsymbol{v}_{j,k}^n$ and details $\boldsymbol{d}_{j,k,e}^n$ on the coarser scales $j = L-1, \ldots, 0$. In fact, relations

$$\boldsymbol{v}_{j,k}^n := \sum_{r \in \mathcal{M}_{j,k}^0} m_{r,k}^{j,0} \, \boldsymbol{v}_{j+1,r}^n, \quad k \in I_j, \tag{44}$$

$$\boldsymbol{d}_{j,k,e}^n := \sum_{r \in \mathcal{M}_{j,k}^e} m_{r,k}^{j,e} \, \boldsymbol{v}_{j+1,r}^n, \quad k \in I_j, \; e \in E^*, \tag{45}$$

are derived from the two–scale relations for the box function and the modified box wavelet. Applying the multiscale transformation (38) to (42), we obtain discrete evolution equations for the cell averages and the details

$$\boldsymbol{v}_{j,k}^{n+1} + \theta \, \lambda_{j,k} \, \boldsymbol{B}_{j,k}^{n+1} = \boldsymbol{v}_{j,k}^n - (1-\theta) \, \lambda_{j,k} \, \boldsymbol{B}_{j,k}^n, \tag{46}$$

$$\boldsymbol{d}_{j,k}^{n+1} + \theta \, \lambda_{j,k} \, \boldsymbol{D}_{j,k}^{n+1} = \boldsymbol{d}_{j,k}^n - (1-\theta) \, \lambda_{j,k} \, \boldsymbol{D}_{j,k}^n, \tag{47}$$

where the flux balances $\boldsymbol{B}_{j,k}^n$ and their details $\boldsymbol{D}_{j,k}^n$ are determined by

$$\boldsymbol{B}_{j,k}^n := \sum_{r \in \mathcal{M}_{j,k}^0} \boldsymbol{B}_{j+1,r}^n = \sum_{V_{L,r} \subset V_{j,k}} \boldsymbol{B}_{L,r}^n, \tag{48}$$

$$\boldsymbol{D}_{j,k}^n := \sum_{r \in \mathcal{M}_{j,k}^e} \frac{m_{r,k}^{j,e}}{m_{r,k}^{j,e}} \boldsymbol{B}_{j+1,r}^n.$$

Here we employed the definition of the filter coefficients $m_{r,k}^{j,0}$ and $\lambda_{j,k} := \tau/|V_{j,k}|$. So far, we only derived evolution equations for the averages and details corresponding to the *full* grids. The adaptive finite volume scheme is now determined by a selection of evolution equations corresponding to the adaptive grid $\mathcal{G}_{L,\varepsilon}^{n+1}$, i.e.,

$$\boldsymbol{v}_{j,k}^{n+1} + \theta\,\lambda_{j,k}\,\boldsymbol{B}_{j,k}^{n+1} = \boldsymbol{v}_{j,k}^{n} - (1-\theta)\,\lambda_{j,k}\,\boldsymbol{B}_{j,k}^{n}, \quad (j,k) \in \mathcal{G}_{L,\varepsilon}^{n+1}. \tag{49}$$

It remains to outline (i) the efficient computation of the local flux balances $\boldsymbol{B}_{j,k}^{n}$ and (ii) the construction of $\mathcal{D}_{L,\boldsymbol{\varepsilon}}^{n+1}$ by means of $\mathcal{D}_{L,\boldsymbol{\varepsilon}}^{n}$.

3.4.2 Local Flux Evaluation

According to the definition of the flux balances (48) the computation of $\boldsymbol{B}_{j,k}^{n}$ requires *all* flux balances $\boldsymbol{B}_{L,r}^{n}$ corresponding to the cells $V_{L,r} \subset V_{j,k}$ of the *finest* level. However, exploiting the conservation property of the fluxes (43) the right hand side of (48) can be rewritten as

$$\boldsymbol{B}_{j,k}^{n} = \sum_{l} |\Gamma_{k,l}^{j}|\,\boldsymbol{F}_{k,l}^{j,n} \tag{50}$$

with the local numerical fluxes

$$\boldsymbol{F}_{k,l}^{j,n} = \sum_{\Gamma_{k',l'}^{j+1} \subset \Gamma_{k,l}^{j}} \boldsymbol{F}_{k',l'}^{j+1,n} = \sum_{\Gamma_{k',l'}^{L} \subset \Gamma_{k,l}^{j}} \boldsymbol{F}_{k',l'}^{L,n}. \tag{51}$$

This is sketched in Figure 7 for a dyadic grid refinement. According to (48) we have

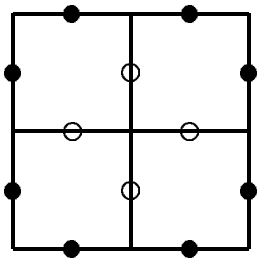


Figure 7 Computation of flux balance

to compute all fluxes marked by $\bullet$ and $\circ$. However, the internal fluxes corresponding to $\circ$ cancel each other out due to the conservation property (43). Thus, an efficient computation of flux balances should be based on (50) rather than (48). Furthermore, we observe that the computation of the numerical fluxes $\boldsymbol{F}_{k',l'}^{L,n}$ requires the computation of cell averages on the *finest* discretization level which have to be provided by means of the inverse two–scale transformation (39). In general, this inflates the complexity by a logarithmic factor depending on the spatial dimension d. In order to treat multidimensional problems a less expensive approximation is desirable. In [23] two alternatives are discussed where the local numerical fluxes $\boldsymbol{F}_{k,l}^{j,n}$ are either

computed by cell averages corresponding to the locally finest discretization level or by the cell averages provided by the adaptive grid. Although these alternatives are less expensive we have to be aware that an error is introduced. As investigations in [9] show, this error might become significant in case of a first order finite volume scheme. However, if the reference scheme is a higher–order accurate scheme employing a high–order reconstruction, then no significant loss in accuracy has been observed.

3.4.3 Prediction

Before the evolution step (49) can be performed, we have to determine the adaptive grid on the *new* time level. Since the corresponding averages, respectively details are not yet available, we have to *predict* all details that may become significant due to the evolution by means of the details on the *old* time level. In order to guarantee the adaptive scheme to be *reliable* in the sense that no significant future feature of the solution is missed, the prediction set $\tilde{\mathcal{D}}_{\varepsilon}^{n+1}$ has to satisfy

$$\mathcal{D}_{L,\varepsilon}^{n} \cup \mathcal{D}_{L,\varepsilon}^{n+1} \subset \tilde{\mathcal{D}}_{L,\varepsilon}^{n+1}, \tag{52}$$

where, of course $\mathcal{D}_{L,\varepsilon}^{n+1}$ is not known yet. In [18] Harten suggested a heuristic approach taking into account that (i) details in a local neighborhood of a significant detail may also become significant within one time step due to the finite speed of propagation and (ii) gradients may become steeper causing significant details on a higher refinement level due to the developing of discontinuities. So far Harten's approach could not be rigorously verified to satisfy (52). However, a slight modification of Harten's prediction strategy has recently been shown to lead to a reliable prediction strategy in the sense of (52), at least for a certain class of *explicit* finite volume schemes applied to *one–dimensional scalar* conservation laws on *uniform dyadic grids* as base hierarchies, see [9]. Here the explicit evolution equations ($\theta = 0$) for the details (47) are employed.

3.5 Error Analysis

The above adaptation strategy based on compressing the arrays of detail coefficients is essentially a perturbation technique. Therefore the objective of the proposed adaptive scheme is to reduce for a given finite volume scheme computational complexity while preserving up to a fixed constant factor the accuracy provided by the reference scheme corresponding to the uniform finest discretization level. In order to make this precise we introduce the averages $\hat{\boldsymbol{u}}_L^n$ of the exact solution, the averages $\boldsymbol{v}_L^n$ determined by the finite volume scheme and the averages $\overline{\boldsymbol{v}}_L^n$ of the adaptive scheme. We stress, that the sequence $\overline{\boldsymbol{v}}_L^n$ is *not* generated by the adaptive scheme on the full uniform highest level L. In order to facilitate the intended accuracy comparisons these data are conceptually generated from the adaptive array $(\boldsymbol{v}_{j,k}^n)_{(j,k)\in\mathcal{G}_{L,\varepsilon}^n}$ by

means of the inverse multiscale transformation (39) after assigning the value zero to the non–significant details.

An ideal strategy would be to prescribe an error tolerance tol and determine during the computation a possibly small number of refinement levels L such that the accuracy requirement is met, i.e.,

$$\|\hat{\boldsymbol{u}}_L^n - \overline{\boldsymbol{v}}_L^n\| \leq tol$$

for possibly small L. Here the error is measured in the weighted l_1–metric

$$\|\boldsymbol{u}_L\| := \sum_{k \in I_L} |V_{L,k}|\, |u_{L,k}|,$$

i.e., the L_1–norm of a piecewise constant scalar function, which is usually applied in the error analysis of finite volume schemes. In order to estimate the discretization error we split the error into two parts corresponding to the *discretization error* $\boldsymbol{\tau}_L^n := \hat{\boldsymbol{u}}_L^n - \boldsymbol{v}_L^n$ of the reference finite volume scheme and the *perturbation error* $\boldsymbol{e}_L^n := \boldsymbol{v}_L^n - \overline{\boldsymbol{v}}_L^n$, i.e., it would suffice to make sure that the sum of the unperturbed discretization error and the perturbation stays below the tolerance

$$\|\hat{\boldsymbol{u}}_L^n - \overline{\boldsymbol{v}}_L^n\| \leq \|\boldsymbol{\tau}_L^n\| + \|\boldsymbol{e}_L^n\| \leq tol. \tag{53}$$

Note, that $\boldsymbol{e}_L^n$ describes indeed a perturbation of the reference scheme because the adaptive scheme coincides with the reference scheme when the threshold value ε is zero. At the current stage we are not able to extract a reliable value of L from the computation yet. Therefore we now assume that there is an a priori error estimate of the discretization error such that

$$\|\boldsymbol{\tau}_L^n\| \leq C\, 2^{-\alpha\, L}. \tag{54}$$

Here the cell diameters are assumed to be proportional to 2^{-L} and α denotes the convergence order of the reference scheme. Then the number of necessary refinement levels is of course determined by $2^{-\alpha\, L} \sim tol$. In order to preserve the accuracy of the reference scheme we now may admit a perturbation error which is proportional to the discretization error.

So far, this ideal concept can only be rigorously founded for scalar conservation laws because rigorous error estimates are only available for this class of problems, see [5, 6]. In this case it remains to estimate the perturbation error. To this end, we first note that in each time step a threshold error is introduced that accumulates in time. Therefore the best we can hope for is an estimate of the form

$$\|\boldsymbol{e}_L^n\| \leq C\, n\, \varepsilon \tag{55}$$

where the constant C is independent of L, n, τ and ε. Since the threshold error may in addition be amplified in each evolution step, the existence of such a constant is not obvious. However, if the multiscale transformation is stable, the explicit scheme

is l_1–*contractive* and if the prediction is reliable then one can show that the constant C is indeed independent of the number of refinement levels L, the threshold value ε and the number of time steps n, [9]. From (54) and (55) we then deduce that the perturbation error stays proportional to the discretization error provided the threshold value is chosen as

$$\varepsilon \sim 2^{-(1+\alpha)\,L}. \tag{56}$$

3.6 Numerical Results

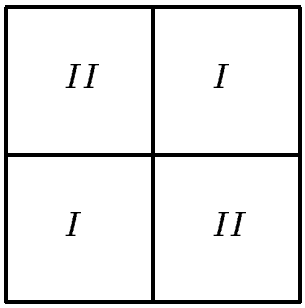
In the previous sections we have outlined a new concept for the construction of adaptive finite volume schemes employing multiscale techniques. For scalar conservation laws this could be verified to be reliable in the sense of (53). In [9] parameter studies have been presented for scalar one–dimensional problems which confirm the analytical results. In the sequel, we verify numerically that the adaptive concept works also reliably for *systems of conservation laws*, i.e., (i) all kinds of singularities, e.g., discontinuities and kinks, corresponding to physically relevant effects are detected by means of the multiscale decomposition and adequately resolved; (ii) the computational complexity of the adaptive scheme with respect to computational costs and memory requirements is proportional to the number of significant details and (iii) the perturbation error is stable in the sense of (55). For this purpose, we present several computations for the two–dimensional Euler equations for perfect air, see also [23]. These have been carried out on PC's with a 600 MHz processor (Pentium III). Further numerical results for real life problems treated in the collaborative research center SFB 401 will be presented and discussed later in Section 6. They have been produced by the solver QUADFLOW incorporating the above adaptive concepts.

Below we consider a two–dimensional Riemann problem where the initial configuration is determined by four states corresponding to the four quadrants of the coordinate system, see Figure 1. Away from the origin of the coordinate system, the solution exhibits a one–dimensional wave pattern consisting of a rarefaction wave, a contact surface and a shock wave. Close to the origin the different one–dimensional waves interact forming a genuinely two–dimensional wave pattern. Note, that the solution is non–stationary.

The reference finite volume scheme for the problem solved here is an essentially non–oscillatory (ENO) scheme characterized by a one–dimensional second order accurate reconstruction technique, and the numerical fluxes are determined by Roe's approximate Riemann solver. For more details on the scheme we refer to [22].

The computational domain $\Omega = [-0.4, 0.4]^2$ is discretized by a uniform Cartesian grid with spatial step size $h_L = 0.04 \times 2^{-L}$ where we perform $n_L = 150 \times 2^{L-2}$ time steps on the time interval $[0, 0.15]$ with a temporal step size $\tau_L = 0.001 \times 2^{2-L}$. Note that the discretization is chosen such that the CFL condition holds on the *finest* discretization level.

Table 1 2D Riemann problem: Initial configuration



	State I	State II
ϱ	0.125	1.000
u_x	1.000	0.000
u_y	1.000	0.000
p	0.100	1.000

The multiscale analysis is based on the modified box wavelets, see Section 3.3.3, where the degree of vanishing moments is chosen as $M = 3$. For details on the choice of the stencils $\mathcal{L}^e_{j,k}$ in case of Cartesian grids see [23].

In particular, the thresholding procedure is slightly modified for vector–valued functions. Here we take into account that the conservative quantities may vary by several orders of magnitude. In principle, we could choose a specific threshold value for each quantity. In practice, a different strategy turned out to be preferable. To this end, we introduce for a vector valued detail $\boldsymbol{d}_{j,k,e}$ its counterpart $\boldsymbol{d}^*_{j,k,e}$ by

$$(\boldsymbol{d}^*_{j,k,e})_i := \begin{cases} (\boldsymbol{d}_{j,k,e})_i & , \quad u_i^\infty < tol \\ \frac{(\boldsymbol{d}^*_{j,k,e})_i}{u_i^\infty} & , \quad \text{elsewhere} \end{cases}$$

where each component is scaled by the maximum

$$u_i^\infty := \max_{(j,k)\in\mathcal{G}_{L,\varepsilon}} |(\hat{u}_{j,k})_i|, \quad i = 1, \ldots, m.$$

Here tol is a machine depending tolerance to avoid division by 0. Alternatively, we could have rewritten the equations (10) in dimensionless form. Then we call a detail significant, if there is an $e \in E^*$ such that $|(\boldsymbol{d}^*_{j,k,e})_i| > \varepsilon_j$ for at least one component $i = 1, \ldots, m$.

Finally, we have to specify the local flux evaluation. Here we do not apply the exact computation according to (51) but the cheaper alternative based on the evaluation of the numerical fluxes with respect to a locally structured grid because the reference numerical flux is only defined on a *structured* grid.

For the test configuration under consideration we have carried out a parameter study with respect to an increasing number of refinement levels L where the threshold value is kept fixed by $\varepsilon = 0.001$ for all computations. This is not quite in agreement with the ideal strategy outlined in Section 3.5 but we are lacking reliable a-priori estimates in the present cases and focus therefore on the *stability* of the scheme, a point that will be taken up again later. In Figure 8 the density is presented for $t = 0.15$ [s]. The corresponding adaptive grid is shown in Figure 9. These figures

reflect the computational results for the highest number of refinement levels listed in the Table 2. We observe that the higher resolution levels are only accessed near discontinuities. This verifies that the adaptation criterion based on significant details is able to detect the relevant physical effects in the flow field and to resolve them adequately.

Table 2 2D Riemann problem: Parameter study

L	N_L	$N_{\mathcal{G}}$ %	C_{MS} [min]	Mem [MB]	S_{MS}	$\|\boldsymbol{e}_L\|_*$
3	25600	58	4.53E+0	16	0.98	1.1E−3
4	102400	34	2.33E+1	39	1.62	1.2E−3
5	409600	19	1.12E+2	90	2.67	1.2E−3
6	1638400	10	4.99E+2	186	4.79	
7	6553600	5	1.98E+3	337	9.67	

Furthermore we are interested in the computational complexity of our scheme with regard to computation time and memory that is documented in Table 2. First of all, we consider the number $N_{\mathcal{G}} := \#\,\mathcal{G}_{L,\boldsymbol{\varepsilon}}/N_L$ representing the number of cells corresponding to the largest adaptive grid determined during the computation relative to the number $N_L = 400 \times 4^L$ of all cells corresponding to the full grid on level L. We note that the relative number of cells reduces exponentially almost by a factor 2. In fact, the total number of cells increases by a factor 2 for the adaptive grid instead of 4 for the uniform grid. This is also reflected by the memory size Mem which corresponds to the peak of memory that has been allocated in one time step. Comparing the numbers Mem for the different computations corresponding to an increasing number of refinement levels, we conclude that memory size is increasing accordingly by a factor 2. Analogously, this is reflected in the computation times C_{MS} used by the adaptive scheme and C_{FVS} of the reference scheme on the full grid of level L. The computation time C_{MS} increases with each additional refinement level because the number of time steps is doubled according to the CFL condition for the finest level. Since the number of cells in the adaptive grid increases in addition by a factor 2, the computation time C_{MS} increases by a factor 4 instead of 8 for the reference scheme. This results in an exponential behavior of the speedup rates $S_{MS} := C_{FVS}/C_{MS}$, i.e., the computation time becomes significantly smaller for the adaptive scheme in comparison to the computation of the reference scheme on the full grid. These observations confirm that the number of floating point operations as well as the memory size of the resulting scheme stays proportional to the number of cells in the adaptive grid.

The realization of such optimal computational complexity behavior depends heavily on appropriate data structures. In our implementation we use hash tables. For details on the design and the implementation of the data structures see [24, 23].

Of course, the objective of the adaptive strategy is to reduce the computational complexity while maintaining (up to a constant) the accuracy in comparison with the reference scheme on a uniform grid. Since for systems of conservation laws bounds on the global discretization error are not available we focus in the following just on the stability of the perturbation error in the sense of (55). That is, instead of balancing discretization and perturbation error we fix the threshold value ε and explore the dependence of the perturbation error on the level L, the number of time steps n and the temporal step size τ. For this purpose, the perturbation error is also listed in Table 2 as far as the computation for the reference scheme on the uniform discretization level could be performed on the available computers. Here the perturbation error $\boldsymbol{e}_L$ is measured in the weighted l_1–metric where each conservative quantity is scaled by its global maximum in the computational domain, i.e.,

$$\|\boldsymbol{e}_L\|_* := \max_{i=1,\dots,m} \|\boldsymbol{e}_{L,i}\| / \|\overline{\boldsymbol{v}}_{L,i}\|_\infty$$

where the vectors $\boldsymbol{e}_{L,i} := ((\boldsymbol{e}_{L,k})_i)_{k\in I_L} = ((\overline{\boldsymbol{v}}_{L,k})_i) - (\boldsymbol{v}_{L,k})_i)_{k\in I_L}$ and $\overline{\boldsymbol{v}}_{L,i} := ((\overline{\boldsymbol{v}}_{L,k})_i)_{k\in I_L}$ correspond to the perturbation error and the results of the adaptive scheme for a single conservative quantity $i = 1, \dots, m$. Recall that the thresholding is also applied with respect to the scaled details. For instance, see Figure 10 where the contours of the perturbation error are shown for the density. We observe that the error is proportional to the threshold value ε, in particular, it does not increase with increasing number of refinement levels L.

We conclude the discussion on the parameter studies with a remark on the "ideal" computation as discussed in Section 3.5. Whenever the perturbation error is smaller than the discretization error, we waste performance in the sense of computation time and memory size, because the discretization error is still dominating. Conversely, we loose accuracy if the perturbation error is larger than the discretization error. Therefore the ideal computation of the parameter study corresponds to the refinement level where both the perturbation error and the discretization error are balanced. Since in our parameter studies the threshold value ε is fixed, we cannot directly conclude on the discretization error. If we *assume* that the discretization error for a uniform grid of level L behaves like $2^{-\alpha L}$ we see from Table 2 at which level the recorded perturbation error is of comparable size. This gives an impression of the interrelation between the overall accuracy and the amount of computational work. However, the computations on the full uniform grid of level L can only be realized for a small number of refinement levels L due to the huge amount of memory needed. This constitutes a significant advantage of adaptive schemes over non–adaptive ones on uniform meshes, i.e., under relevant accuracy requirements simulations become computationally tractable that would not be possible to perform with the aid of quasi–uniform meshes.

3.7 Summary of Requirements

We conclude the discussion of adaptation strategies with briefly summarizing the main requirements on the finite volume discretizations and the corresponding mesh generators that are put forward by the above concepts.

(i) The multiscale setting as outlined in Section 3.3.1 is based on a *grid hierarchy* of *nested grids*. Although we never access to the *full* grids of this hierarchy the grid generator has to provide an efficient representation of this hierarchy depending only on a small number of parameters in comparison to the full grids. Moreover, this representation has to allow for a *fast computation of higher order moments* with regard to the construction of higher order biorthogonal systems, see Section 3.3.3.

(ii) The efficiency of the flux computation crucially depends on the assumption that the numerical fluxes of the underlying finite volume discretization are *conservative*, see Section 3.4.2. In addition, the computation of the numerical flux balances can be further improved when computing the incorporated numerical fluxes by means of the averages corresponding to the adaptive grid instead of the averages on highest resolution level. This introduces an error that can be compensated when using a *higher order reconstruction* in the finite volume scheme.

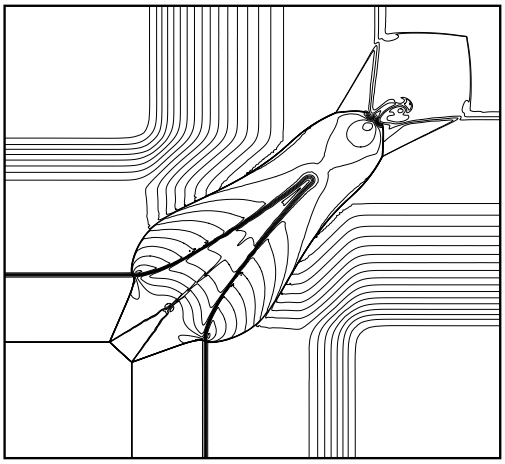


Figure 8 2D Riemann problem: Density contours ($L = 7$)

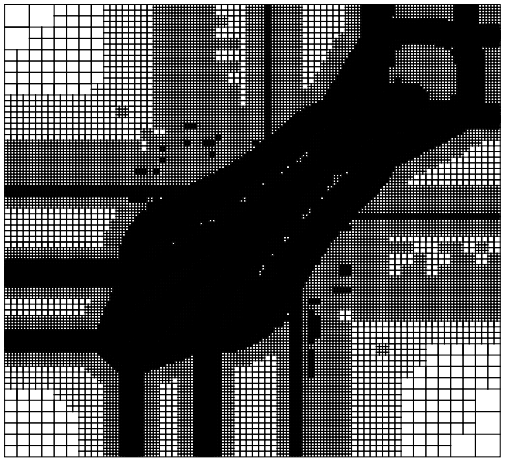


Figure 9 2D Riemann problem: Adaptive grid ($L = 7$)

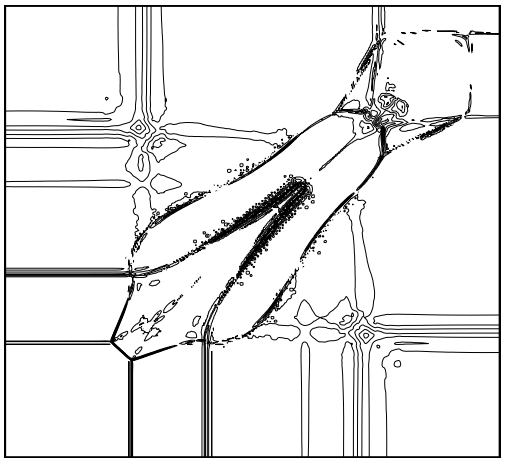


Figure 10 2D Riemann problem: Contours of perturbation error in density ($L = 5$)

4 High Quality Mesh Generation Using B-Splines

4.1 Objectives and Basic Concept

The algorithms and data structures of a grid generation code have to reflect the needs of the numerical scheme applied to solve the problem at hand. The QUADFLOW project aims at developing an adaptive code for flow problems in time varying geometries. Hence the grid generation task cannot be viewed as preprocessing, because the grid generation aspects are tightly connected to the solution process due to transformations and refinements. Here, the design of the grid generation concept used within QUADFLOW is motivated by the following objectives.

(i) With regard to the stability and accuracy of the flow calculations the usual grid quality measures such as smoothness and orthogonality have to be met. It is widely accepted that hexahedral based meshes are preferable for the discretization of viscous flows, because they facilitate best the generation of boundary fitted anisotropic meshes. In particular this is important for the resolution of boundary layers. Thus, *block structured*, *boundary conforming* grids have been employed.

(ii) The grid generator has to fulfill the requirements posed by the multiscale technique, which are summarized in Section 3.7. Due to the dynamic adaptation it is not known a priori, where a fine scale discretization is needed in the flow field. Therefore it seems useful to separate the geometric aspect of grid generation from the discretization aspect. This is achieved by representing the logically Cartesian grids by their analytic counterparts, namely, by parametric mappings from the unit square or unit cube to the physical domain, see Figure 12.

Common experience shows that the main disadvantage of structured grid generation consists in the problem to generate suitable block decompositions around complex geometries. This tends to be a time consuming and difficult task and usually requires human expertise and interaction. Although it will be difficult or even impossible to automate this work completely, software tools have to be provided, that are simple and intuitive to use and reduce the necessity of user interaction as much as possible. Otherwise this problem will inevitably become the most severe obstacle to realistic applications. A typical example for a two-dimensional block structured grid as used in the current work is given in Figure 11. The boundaries of the blocks have been defined interactively.

In the grid generation community a lot of work has been devoted to exploiting the excellent abilities of B-Splines and existing CAD programs, see for example [26]. Many user interfaces of grid generation programs provide B-Splines or NURBS as tools for modeling boundary curves or surfaces. In the present approach we go

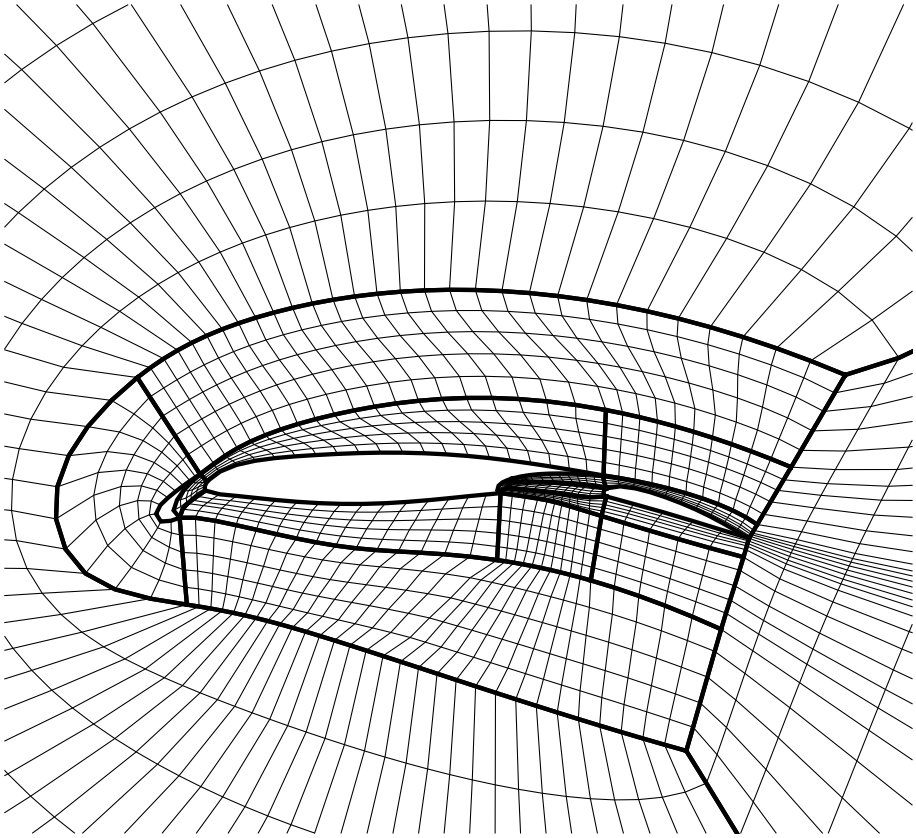


Figure 11 Block decomposition around high lift configuration

even one step further and aim at representing the blocks as independent curvilinear coordinate systems using B-Spline tensor products.

In the following we describe and analyze the above concept in more detail and present some tools that have been developed to realize this concept.

4.2 Representation of Grids as Parametric Mappings

The discussion in the previous section indicates that the grid generation has to meet the following requirements.

(R1) *Boundary conforming* grids which are appropriate for finite volume calculations of flows based on the Euler and Navier-Stokes equations, have to be generated.

(R2) A hierarchy of nested grids for the multiscale algorithm has to be provided.

(R3) The local adaptation of the grid during the solution process has to be efficient.

(R4) The transformation of the grids due to moving boundaries has to be cheap with respect to computational time.

In the following, it will be shown that these requirements suggest to represent grids as parametric mappings and to realize these mappings using B-Splines.

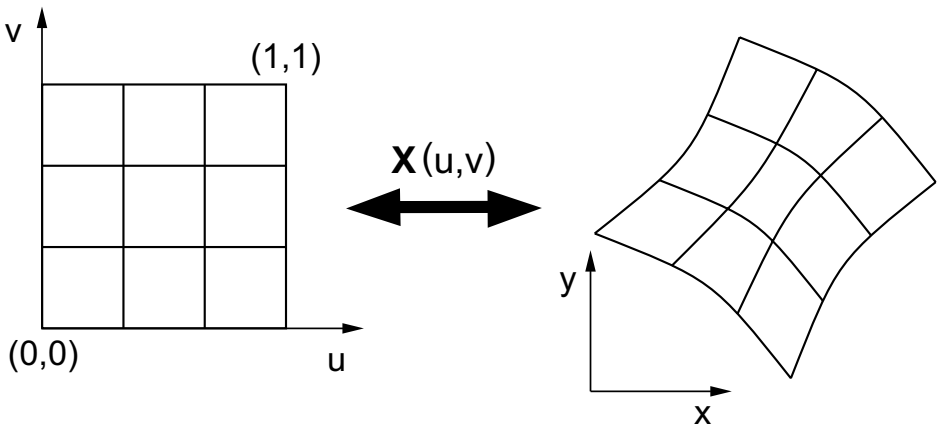


Figure 12 Parametric Mappings

Requirement (R2) states that each cell of a coarse grid must be the union of its subcells. This is indeed a severe constraint, because it implies that the geometric approximation of the physical domain is determined completely by the shape of the coarse grid. This conflicts with the demand, that the shape of the body must be represented precisely. Indeed, one cannot define the coarse cells simply as quadrilaterals in 2D respectively hexahedra in 3D connecting its corner points with straight lines, because this would result in a poor approximation of the boundary curve or surface. Another method would be to calculate the fine grid cells first and to define the coarse grid cells as unions of its subcells. But this would lead to complicated non-smooth shapes of the coarse grid cells, what makes the proper computation of fluxes across the cell interfaces more expensive, and requires the calculation of a refined grid in a preprocessing step. This would in turn contradict the inherent philosophy of adaptive codes, where calculations are initialized on the coarse grid and the grid is refined only where necessary. However, representing grids as curvilinear coordinates allows to define the cells as images of the corresponding cells in computational space. Then grid refinement is easily performed by evaluating the grid function. This method has the following three advantages:

(i) The approximation of the contour does not depend on the refinement level, but only on the quality of the geometric model.

(ii) The grid generation process becomes completely independent of the discretization parameters, for example the number of grid points in the coordinate directions, as the flow solver receives its data simply by evaluating the grid function, see Figure 12.

(iii) From the same representation, grids suitable for both Euler and Navier-Stokes flows can be computed, because the latter only requires the application of an appropriate blending function to resolve the boundary layer.

Note, that the grid cells now have curved edges. This complicates the computation of some geometric quantities, for instance cell volumes, see Section 4.4.5.

Since in realistic problems the geometry will not be given in terms of elementary analytic mappings, such as Cartesian or polar coordinates, one has to approximate given free form geometries. B-Splines seem to be a very appropriate tool for this task, because

(i) they posses excellent approximation properties,

(ii) modeling with B-Splines is intuitive and fast,

(iii) evaluation of B-Splines is fast and numerically stable.

In particular, the first property is important, because it guarantees that one has to compute and store only few parameters to represent a grid with sufficient accuracy. This is of interest with respect to requirement (R4), because one has only to modify these parameters, when the mesh has to be frequently updated in time.

Of course, it is also important that the well established grid generation methods can be integrated into our concept. Indeed, due to the development of fast interpolation and approximation algorithms, the B-Spline representation can be used as post-processing to existing grid generating tools. We feel though, that such a point of view is too restrictive. In fact, the present approach makes use of the genuine advantages of the parametric B-Spline representation within a large variety of algorithms.

A possible alternative are NURBS (”Non-Uniform Rational B-Splines”), because they are even more flexible and are capable of representing typical geometric objects as for example circles and ellipses exactly. But the evaluation of the derivatives of a rational function is rather expensive. This can become a time critical aspect in algorithms that depend on geometrical parameters as for example the curvature of a curve or surface. Therefore we restrict ourselves to integral B-Splines.

4.3 Block Decomposition

The crucial point in structured grid generation is the definition of the block structure. The shape of the blocks is often decisive for the quality of the grid and the calculations performed on it. For the QUADFLOW grid generator the following aspects are of special interest:

(B1) In the vicinity of body contours high quality boundary conforming grids are required in order to resolve the boundary layers.

(B2) In the far-field as many blocks as possible should be Cartesian, because this will enhance the performance of the multiscale algorithm significantly. In fact, the mask coefficients in the multiscale transformation can then be derived

by dilatation and translation of corresponding quantities for a single reference volume.

(B3) There should be as few topological constraints on the decomposition as possible in order to support the automation of the block decomposition.

With regard to the last point, we permit geometrically non-conforming block partitions, i.e. faces of adjacent blocks need not match, and do not impose any continuity conditions across the block borders. Here of course, we exploit the ability of the flow solver to handle hanging nodes.

For the description of the grid topology we have adopted the connectivity hierarchy from blocks to faces to edges to vertices proposed in [45] to the case of parametric mappings. In this concept the geometric description of all topological elements, namely, the blocks, faces, edges and vertices is stored separately. Blocks are viewed as parametric mappings of the unit cube, faces as mappings of the unit square and edges as mappings of the unit interval. In addition we have to store the following *connectivity relations*:

(i) for each block by which six faces it is bordered,

(ii) for each face by which four edges it is bordered, and

(iii) for each edge by which two vertices it is spanned.

For two-dimensional grids we just omit the first kind of relations. Nonconforming block interfaces are realized by allowing faces and edges to be divided into subfaces respectively subedges. The information about the relations between faces and their subfaces respectively edges and their subedges has also to be stored. From this four types of connectivity relations all other topological information can be derived.

To fulfill requirements (B1), we have implemented methods that automatically generate smooth offset curves around the body – see below in Section 4.6. These offsets are curvature dependent and can roughly be interpreted in the context of hyperbolic grid generation. They are generated in a first step, before the rest of the block decomposition is defined. This strategy is illustrated in Figure 13.

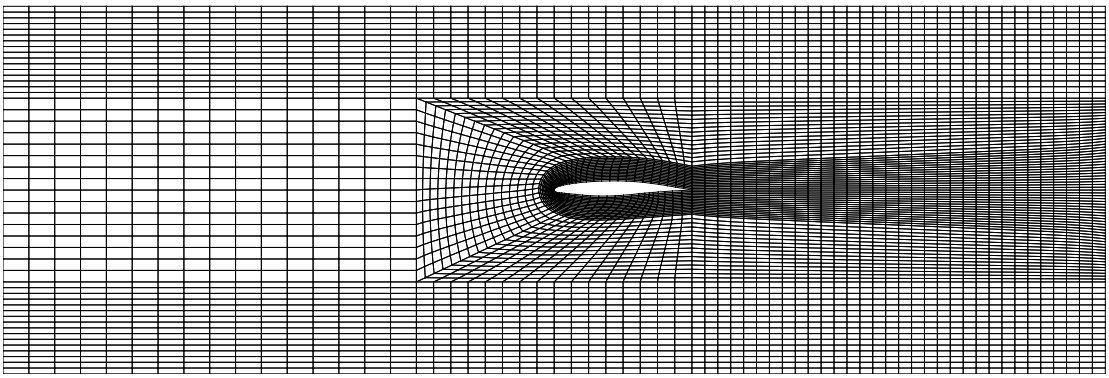


Figure 13 Grid in 12 blocks around reference configuration in a channel

So far only a traditional, user interactive concept of block decomposition has been realized. By means of an existing CAD program [29, 30], which provides the tools based on the B-Spline techniques introduced in Section 4.4, the curves representing the body contour, the offset curves and the block interfaces are defined interactively. These curves serve as input for the grid generation code, that applies the algebraic and/or elliptic methods described in Section 4.5 to the individual blocks. The integration of block-decomposition and grid generation in a common user interface and the automation of these tasks is work in progress.

4.4 B-Spline Basics

In this section we summarize the basic properties of B-Splines needed in this paper. We start with B-Spline functions. Then we show how to build curves and planar grids, surfaces and volume grids. After that we turn to the essential interpolation and approximation algorithms. Details and proofs can be found in many textbooks about numerical analysis and computer graphics, e.g. [31], [35], [40]. In CAD (Computer Aided Design) usually curves and surfaces are treated. It is easy but technical to extend these techniques to volume grids as needed in the 3D-case. For this purpose we have developed fast algorithms for interpolation and approximation. They make extensive use of the tensor-product structure of the B-Splines, leading to an enormous reduction of computation time.

4.4.1 B-Spline Functions

Let $T = (t_0, t_1, \ldots, t_m)$ be a nondecreasing and non-stationary sequence of real numbers, i.e. $t_i \leq t_{i+1}, i = 0, 1, \ldots, m-1.$ and $t_i < t_{i+p}, i = 0, 1, \ldots, m-p.$ T is called "knot vector". Then for $i = 0, \ldots, m-p$ the B-Spline functions $N_{i,p,T}(t) =$

$N_{i,p}(t)$ of order p are recursively defined as

$$N_{i,1}(t) := \chi_{[t_i,t_{i+1})}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{57}$$

$$N_{i,p}(t) := \frac{t - t_i}{t_{i+p-1} - t_i} N_{i,p-1}(t) + \frac{t_{i+p} - t}{t_{i+p} - t_{i+1}} N_{i+1,p-1}(t). \tag{58}$$

They are piecewise polynomials of degree $p-1$, have compact support $[t_i, t_{i+p}]$, are nonnegative and together form a partition of unity.

In the case that T contains multiple knots equation (58) can yield the quotient $0/0$; we define this quotient to be zero. B-Spline functions of order p are C^{p-l-1}-continuous at a knot of multiplicity l.

A p-th order B-Spline curve is now defined by

$$\boldsymbol{x}(t) = \sum_{i=0}^{m-p} \boldsymbol{p_i}\, N_{i,p}(t). \tag{59}$$

The $\boldsymbol{p}_i$ are called *control points* or *de Boor points*. They form in ascending order the control polygon. Evaluation of the curve can be performed very efficiently applying the *de Boor algorithm* which is a recursion formula derived from equation (58).

Geometric modeling with B-Splines is very intuitive, because the shape of the control polygon roughly represents the shape of the B-Spline curve. For planar curves the point $\boldsymbol{x}(t)$, with $t_i < t < t_{i+1}$ always lies in the convex hull of the control points $\boldsymbol{p}_{i-p+1}, \ldots, \boldsymbol{p}_i$ and its position is only influenced by these p control points. Accordingly, moving the control point $\boldsymbol{p}_i$ changes $\boldsymbol{x}(t)$ only in the interval $[t_i, t_{i+p}]$. This is especially convenient for modeling curves at a graphics terminal, because changing the position of one control point has only a local effect on the curve.

In addition B-Splines possess the following *variation diminishing property:* A planar B-Spline curve crosses any straight line at most as many times as its control polygon does. From this it follows for example, that a convex control polygon will necessarily define a convex spline curve.

It is often useful to insert additional control points to an existing B-Spline curve without modifying the shape of the curve itself. This is easily accomplished by *knot insertion*. Successive knot insertion causes the control polygon to converge towards the curve.

In practice we concentrate on cubic B-Spline curves (order $p = 4$) and therefore write in the following $N_i(t)$ instead of $N_{i,4}(t)$. In addition, we use only knot vectors that start and end with p equal knots. In this case the spline function is defined on the interval $[t_{p-1}, t_{m-p+1}] =: [a, b]$. This has the advantage that the first and the last point of the curve coincide with the first and last polygon vertices. Furthermore, the slope of the B-Spline curve at the first and last polygon vertices is equal to the slope of the first and last polygon spans.

Planar grids and surfaces are represented as B-Spline tensor products, taking a net $\boldsymbol{p}_{i,j}$ of control points, two knot vectors U, V and building products of the univariate B-Spline functions:

$$\boldsymbol{x}(u,v) = \sum_{i=0}^{m-p} \sum_{j=0}^{n-q} \boldsymbol{p}_{i,j}\, N_{i,p,U}(u)\, N_{j,q,V}(v), \tag{60}$$

see Figure 14. The subscripts U and V are usually omitted. Bearing this simplification in mind, one defines volume grids in a similar way:

$$\boldsymbol{x}(u,v,w) = \sum_{i,j,k} \boldsymbol{p}_{i,j,k}\, N_{i,p}(u)\, N_{j,q}(v)\, N_{k,r}(w). \tag{61}$$

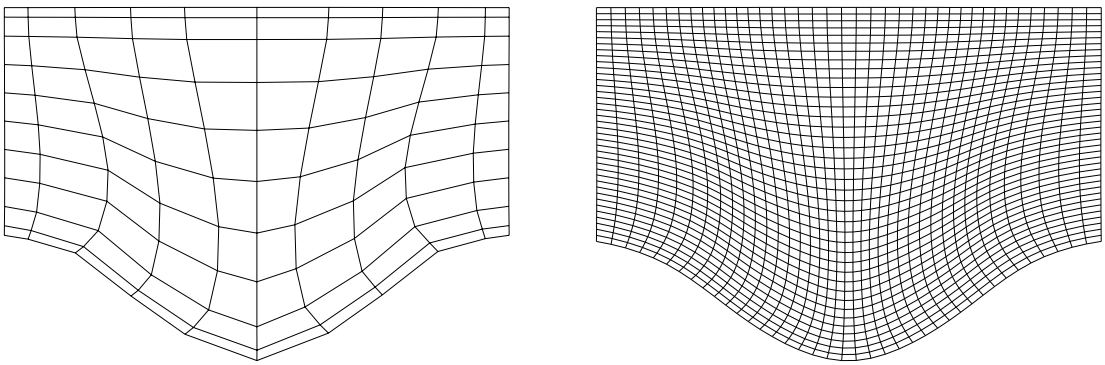


Figure 14 Control Points and Evaluation of Grid Function

The derivative $\boldsymbol{x}'(t)$ of a p-th order B-Spline curve is again a B-Spline curve but of order $p-1$. For surfaces and volumes partial differentiation reduces the order for corresponding *directions*.

4.4.2 Interpolation of Curves

Often the geometric configuration is not given as a spline representation but only as sequences of points. In this case the first task is to transform this point-description into a spline representation via interpolation or approximation.

In the case of cubic B-Spline interpolation we search a curve

$$\boldsymbol{x}(t) = \sum_{i=0}^{m-4} \boldsymbol{p_i} N_i(t) \tag{62}$$

that interpolates the given curve at its knots

$$\boldsymbol{x}(t_{i+3}) = \boldsymbol{x}_i,\ i = 0, 1, \ldots, n := m - 2p + 2 = m - 6. \tag{63}$$

Here we assume, that an appropriate data adapted knot vector with p-fold knots at its ends and no multiple interior knots has been priorly determined. For suitable

strategies see for example [35]. So there are $n+1$ conditions for the $n+3$ control points $\boldsymbol{p}_i$. Thus we have to pose two more conditions. Usually one prescribes the derivatives at the beginning and the end of the curve ($\boldsymbol{x}'(a) = \boldsymbol{v}_0$, $\boldsymbol{x}'(b) = \boldsymbol{v}_n$) (”complete spline interpolation”) or if such data is not available demands vanishing curvature at the beginning and the end of the curve ($\boldsymbol{x}''(a) = \boldsymbol{x}''(b) = 0$). (”natural spline interpolation”). In any case it follows from the theorem of Schoenberg and Whitney that the interpolation problem has a unique solution. Because of the locality of the B-Spline basis the interpolation problem leads to a tridiagonal matrix, that is totally positive due to a result of Karlin and therefore can be solved effectively by Gauss elimination without pivoting.

The following theorems, formulated for scalar functions, give additional reasons for the popularity of the complete spline interpolation [39].

Theorem: Let s be the complete interpolation spline of a function $f \in C^4[a,b]$ with respect to the knots t_i with $h := max_i|t_{i+1} - t_i|$. Then

$$||f - s||_\infty \leq \frac{5}{384} h^4 ||f^{(4)}||_\infty \tag{64}$$

Theorem: Let s be the complete or the natural interpolation spline of a function f and let $y \in C^2[a,b]$ be another interpolating function that fulfills the same boundary conditions. Then

$$\int_a^b s''(t)^2\, dt \;\leq\; \int_a^b y''(t)^2\, dt. \tag{65}$$

The last result, often cited as *best approximation property*, is of certain interest in the context of grid generation as smoothness is an important measure of grid quality. Of course, its value should not be overestimated, because it does not make any statement about the geometric shape of the interpolation curve.

4.4.3 Approximation of curves

Indeed, despite these results, it turns out that interpolation generally is not a sufficient tool for generating suitable representations of boundaries

The top graphs of Figure 15 show curvature plots of two B-Spline representations of the same air-foil, the SFB 401 cruise configuration. The first one was interpolated from a set of 193 discrete points. Although there were no deviations of these points from the original description of the configuration the interpolated spline shows oscillations in the curvature. This effect is typical and may be explained by the fact, that it is not easy to find an appropriate knot vector, and with it a good parameterization of the spline corresponding to the distribution of the given data. This effect can be reduced by approximating more points $\boldsymbol{x}_i$ than control points $\boldsymbol{p}_j$. The second representation was produced by approximation of an enlarged data set of 2000 points with 200 control points. To estimate the influence on the flow solution in the bottom row the pressure distribution on the air-foil calculated by QUADFLOW is plotted.

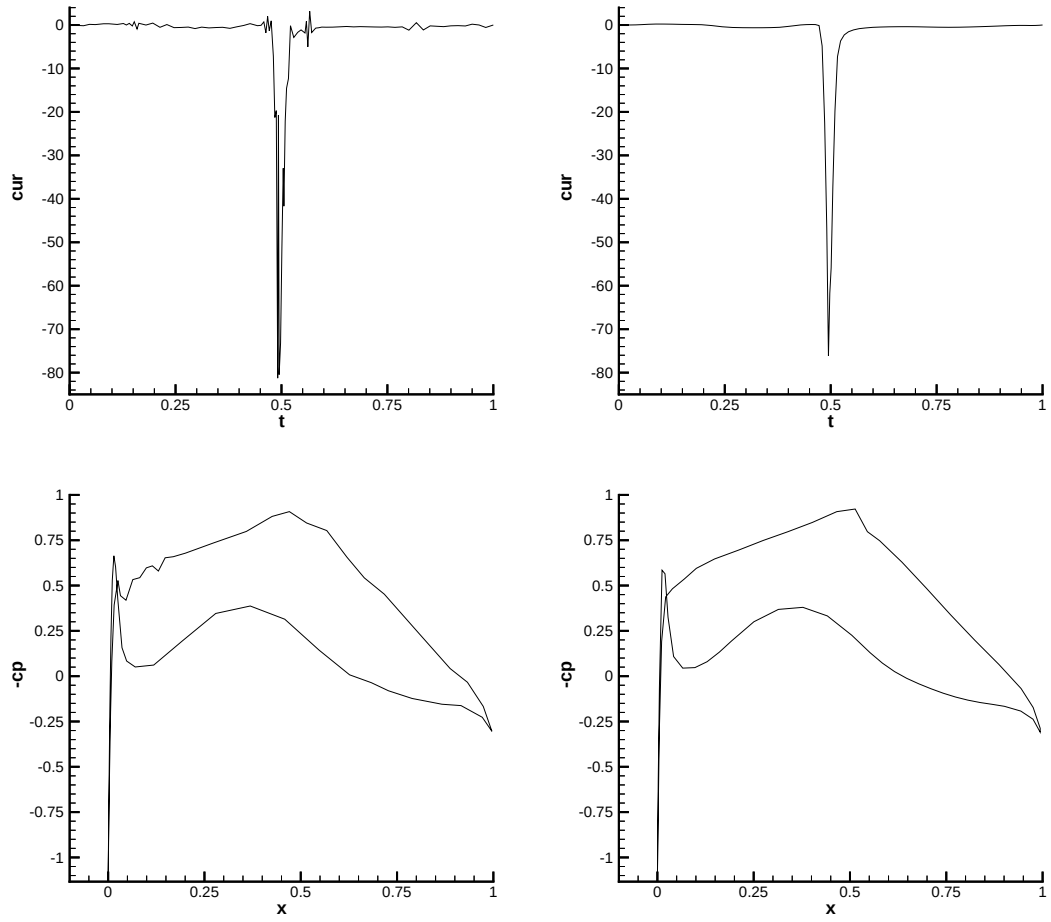


Figure 15 Influence of boundary-approximations

This example shows, that an adaptive code will – in contrast to a method on a fixed grid – react sensibly to even small scale inaccuracies in the description of the geometry. In order to avoid the necessity to work on large numbers of control-points and large data-sets, we implemented fairing methods based on the algorithms given in [37],[34] and [33]. Indeed using these methods, the above mentioned profile can be precisely and smoothly approximated with less than 30 control points.

In the basic approximation algorithm we have to choose a knot vector for the spline and to assign suitable parameter values τ_j to the data points $\boldsymbol{x}_j = \boldsymbol{x}(\tau_j)$. This gives rise to the "least squares problem"

$$||A\boldsymbol{P} - \boldsymbol{X}||_2 \longrightarrow \min \tag{66}$$

where $A = (N_i(\tau_j))_{i,j}$. Since we have more equations than control points we have conditions in between the knots. This enlarges the bandwidth of the matrix A from three to four and it is no longer quadratic. Usually the least squares problem is solved via solving the normal equations, see for example [31]. In view of the extension to the multidimensional case and the worse condition of the normal equations we prefer the solution via QR-decomposition.

4.4.4 Tensor Product Interpolation and Approximation

In the context of grid generation typically the problem arises to find a B-Spline tensor product of the form (60) that approximates a given net $\boldsymbol{x}_{r,s}$ of grid points at predefined parameters (u_r, v_s), $r = 0, \ldots, N$, $s = 0, \ldots, M$. In the case of interpolation, i.e. $M = m - p$ and $N = n - q$ we obtain the following matrix equation:

$$\left(\sum_{i,j} A_{r,i} B_{s,j} \boldsymbol{p}_{i,j} \right)_{\begin{pmatrix} r=0,1,\ldots,M \\ s=0,1,\ldots,N \end{pmatrix}} = A\boldsymbol{P}B^T = \boldsymbol{X}. \tag{67}$$

Here we have assembled the control points $\boldsymbol{p}_{i,j}$ to be computed in the matrix $\boldsymbol{P}$ and the interpolation points in the matrix $\boldsymbol{X}$. Note that the collocation matrices $A = (A_{r,i}) = (N_r(u_i))$ and $B = (B_{s,j}) = (N_s(v_j))$ have the same structure as the corresponding 1D collocation matrices. Therefore equation (67) can be solved easily by LR-decomposition of these matrices.

If M and N are large it is often preferable not to interpolate but to approximate the given points for sake of data reduction . In the following we assume $M > m - p$ and $N > n - q$. Although it is possible to set up and solve the standard least squares problem

$$\sum_{r,s} (\boldsymbol{x}(u_r, v_s) - \boldsymbol{x}_{r,s})^2 \longrightarrow \min, \tag{68}$$

this task is computationally rather expensive, because it requires the solution of fairly large linear systems. Since it is sometimes necessary to solve the approximation problem repeatedly, for example if one wants to approximate the given data within a given accuracy but with minimal number of control points, we want to avoid this computation. Instead we measure the approximation error in another norm in order to exploit the tensor product structure of the problem. To do this, we note that the least squares problem (68) can be formulated equivalently as the problem to minimize the *Frobenius norm* of the matrix $R := A\boldsymbol{P}B^T - \boldsymbol{X}$, where A, B, $\boldsymbol{P}$, and $\boldsymbol{X}$ are the same matrices as in equation (67).

$$||R||_F = \left(\sum_{i,j} R_{i,j}^2 \right)^{\frac{1}{2}} \longrightarrow \min. \tag{69}$$

The matrix R contains the approximation errors in every data point in their natural order. But instead of the Frobenius norm we propose to minimize of the 2-norm, i.e. the *spectral norm* of R.

This can be justified by the following elementary norm equivalence, that holds for every matrix $M \in \mathbb{R}^{m \times n}$ with $m > n$:

$$||M||_2 \leq ||M||_F \leq \sqrt{n}||M||_2. \tag{70}$$

This lemma states that the solution of the modified minimization problem delivers reasonable approximations of the solution of the least squares problem, as the two norms differ at most by a factor that equals the square root of the number of data points in one coordinate direction.

The advantage of the spectral norm is that it is invariant under multiplication with orthogonal matrices. So we can solve the problem analogously to the interpolation case but using QR instead of LR-decomposition. In matrix notation this reads as

$$\begin{aligned} ||A\boldsymbol{P}B^T - \boldsymbol{X}||_2 &= ||Q_A R_A \boldsymbol{P} (Q_B R_B)^T - \boldsymbol{X}||_2 \\ &= ||R_A \, \boldsymbol{P} \, R_B^T \; - \; Q_A^T \, \boldsymbol{X} \, Q_B \, ||_2 \longrightarrow \text{min.} \end{aligned} \tag{71}$$

Note that again A and B are 1D collocation matrices, which are generally banded matrices with low bandwidth, for bicubic approximation, for example, the bandwidth is ≤ 4. So the computation of the QR-decompositions needed is cheap. In every case this results in computational costs proportional to the number of given points.

In the case of 3D interpolation we are given a 3-dimensional array $\boldsymbol{X} = (\boldsymbol{x}_{i,j,k})$ of grid points and have to determine an interpolating B-Spline tensor product in form (61). We can easily generalize the notation used above to

$$\left(\sum_{i,j,k} A_{r,i} B_{s,j} C_{t,k} P_{i,j,k} \right)_{\begin{pmatrix} r=0,1,\ldots,M \\ s=0,1,\ldots,N \\ t=0,1,\ldots,L \end{pmatrix}} = B(A\boldsymbol{P}C^T) = \boldsymbol{X}. \tag{72}$$

Again solving this system is reduced to the task to compute LR-decompositions of the 1D-collocation matrices A, B, and C. Since $\boldsymbol{P} \in (\mathbb{R}^3)^{L \times M \times N}$ the different matrix products work on the first, second and third index of $\boldsymbol{P}$, that is running on the other ones. Analogously we treat the 3D approximation case, just by employing the QR-decompositions instead of LR decompositions. In this case we cannot give a motivation based on equivalence of matrix norms similar to the 2D case, but the algorithm is fast and yields satisfactory results.

4.4.5 Computation of Volumes and Moments

For the multiscale analysis (here in 2D) the computation of the cell moments $M_{n,m} := \int_V x^n y^m \, dV$ is required, e.g. for $n = m = 0$ one gets the volume of V. As we have defined a grid-cell to be the image of a cell in computational space, the calculation of this integral is expensive, because the cells have curved edges. To simplify the evaluation of the integrals we use the Gaussian integral theorem in the following way. Setting

$$\boldsymbol{f}(\boldsymbol{x}) := \left(\frac{1}{n+1} x^{n+1} y^m, \, 0 \right) \tag{73}$$

we can reduce the cell integral to an integral over its boundary:

$$M := \int_V x^n y^m dV = \int_V div\, \boldsymbol{f}(x) dx = \int_{\partial V} \boldsymbol{f} \cdot \boldsymbol{n} ds. \tag{74}$$

Here $\boldsymbol{n}$ denotes the outer unit normal on the boundary of V. Assuming V to be the image of the cell $[u_1, u_2] \times [v_1, v_2]$ and using the parameterization of the cell boundaries implicitly given by the grid function, we obtain from the substitution rule

$$\begin{aligned} M = & \int_{v_1}^{v_2} \boldsymbol{f}(\boldsymbol{x}(u_1, \tau)) \boldsymbol{n}(u_1, \tau) ||\boldsymbol{x}_v(u_1, \tau)||_2 \, d\tau \\ & + \int_{v_1}^{v_2} \boldsymbol{f}(\boldsymbol{x}(u_2, \tau)) \boldsymbol{n}(u_2, \tau) ||\boldsymbol{x}_v(u_2, \tau)||_2 \, d\tau \\ & + \int_{u_1}^{u_2} \boldsymbol{f}(\boldsymbol{x}(\tau, v_1)) \boldsymbol{n}(\tau, v_1) ||\boldsymbol{x}_u(\tau, v_1)||_2 \, d\tau \\ & + \int_{u_1}^{u_2} \boldsymbol{f}(\boldsymbol{x}(\tau, v_2)) \boldsymbol{n}(\tau, v_2) ||\boldsymbol{x}_u(\tau, v_2)||_2 \, d\tau. \end{aligned} \tag{75}$$

Noticing that the product of the outer unit normal and the norm of the derivative is except of the sign just the normal to the tangent itself, the above equation simplifies to (with $(v_1, v_2)^\perp = (-v_2, v_1)$)

$$\begin{aligned} M = & \int_{v_1}^{v_2} \boldsymbol{f}(\boldsymbol{x}(u_1, \tau)) \boldsymbol{x}_v^\perp(u_1, \tau) \, d\tau \\ & - \int_{v_1}^{v_2} \boldsymbol{f}(\boldsymbol{x}(u_2, \tau)) \boldsymbol{x}_v^\perp(u_2, \tau) \, d\tau \\ & - \int_{u_1}^{u_2} \boldsymbol{f}(\boldsymbol{x}(\tau, v_1)) \boldsymbol{x}_u^\perp(\tau, v_1) \, d\tau \\ & + \int_{u_1}^{u_2} \boldsymbol{f}(\boldsymbol{x}(\tau, v_2)) \boldsymbol{x}_u^\perp(\tau, v_2) \, d\tau. \end{aligned} \tag{76}$$

We now see that the integrands are products of piecewise univariate polynomials and that the integrals therefore can be evaluated exactly. But due to the high degree of these polynomials this calculation turns out to be rather expensive. Therfore it should be performed only where it is really necessary, i.e. near the body contours. In the far field the use of simpler grid models is preferable.

4.5 Generation of Boundary Conforming Grids

4.5.1 Basic Procedure

Consider a simply connected bounded domain Ω in two dimensional space. Suppose that Ω is bounded by four edges and that these edges are given as B-Splines $\boldsymbol{x}_{u0}(u)$,

$\boldsymbol{x}_{u1}(u)$, $\boldsymbol{x}_{0v}(v)$ and $\boldsymbol{x}_{1v}(v)$. The task is to generate a grid function $\boldsymbol{x}(u,v)$, that interpolates the given curves along its boundaries:

$$\boldsymbol{x}(u,0) = \boldsymbol{x}_{u0}(u), \qquad \boldsymbol{x}(0,v) = \boldsymbol{x}_{0v}(v), \tag{77}$$

$$\boldsymbol{x}(u,1) = \boldsymbol{x}_{u1}(u), \qquad \boldsymbol{x}(1,v) = \boldsymbol{x}_{1v}(v). \tag{78}$$

As we want to represent the grid by a B-Spline tensor product we have to assure first, that the opposite edges possess the same knot vector. This can easily be achieved by knot insertion. Since we only use knot vectors with p-fold knots at their ends the control polygons of the boundary B-Splines are always equal to the corresponding border lines of the control net for any tensor product grid function conforming to the above boundary conditions.

Most standard methods cannot manipulate the B-Spline control points directly, for example the finite difference discretization of an elliptic grid generator uses point representations of the grid. In this case we evaluate the B-Spline at its knots and after successful grid generation interpolate the points again. Since the solution of the interpolation is unique it is guaranteed that the original boundary curve is reproduced.

4.5.2 Algebraic Grid Generation

We employ several variants of transfinite interpolation, which is based on Gordon's method [38].

We only mention the standard linear blend. Denoting the four corner points of the given domain with $\boldsymbol{x}_{00} = \boldsymbol{x}(0,0)$, $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{01}$ and $\boldsymbol{x}_{11}$ it can be written in the form

$$\begin{aligned} \boldsymbol{x}(u,v) = \quad & (1-u \quad u) \begin{pmatrix} \boldsymbol{x}_{0v}(v) \\ \boldsymbol{x}_{1v}(v) \end{pmatrix} + (\boldsymbol{x}_{u0}(u) \quad \boldsymbol{x}_{u1}(u)) \begin{pmatrix} 1-v \\ v \end{pmatrix} \\ - \quad & (1-u \quad u) \begin{pmatrix} \boldsymbol{x}_{00} & \boldsymbol{x}_{01} \\ \boldsymbol{x}_{10} & \boldsymbol{x}_{11} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}. \end{aligned} \tag{79}$$

This method is especially easy to apply, because, as proven in [36], a linear blend obtained from four cubic B-Spline boundary curves is indeed itself a bicubic tensor product B-Spline and its control net can be obtained directly from the boundary control polygons, just by interpreting all the control points $\boldsymbol{p}_{ij}$, $i = 0,\ldots,N$, $j = 0,\ldots,M$ as values $\boldsymbol{p}(i/N, j/M)$ of a virtual grid function $\boldsymbol{p}$, and applying the formula defining the linear blend to the inner control grid points.

The other basic method is to prescribe in addition the normal derivatives to the given curves and to interpolate these data with a bicubic Coons-Patch.

4.5.3 Elliptic Grid Generation

It is common experience that in practical application the use of an elliptic grid generator is inevitable, because elliptic methods are superior to all alternatives with

respect to robustness of the algorithm and smoothness of the generated grids. The disadvantage of elliptic methods is that they require the very time consuming solution of a nonlinear PDE and are therefore not well suited for time varying geometries or interactive use.

We implemented the grid generator by Spekreijse [43], [44], because his concept of coupling elliptic with algebraic transformations is theoretically well founded and easily integrated into our approach, that already provides a variety of algebraic methods.

To speed up the solution of the PDE we begin with generating very coarse discrete grids with the aid of the elliptic generator. Then this grid will be converted into a B-Spline representation. If the quality of the grid achieved is satisfactory we stop here, if not we evaluate the function at a higher number of points and use this grid as initial guess for a new iteration with the elliptic solver. Since this starting grid is then usually much better than for example an initial guess based on transfinite interpolation, the number of Newton or fix-point iterations used for the solution of the nonlinear PDE can be reduced. But it turns out that usually it is sufficient to compute and interpolate very coarse elliptic grids to achieve a sufficient quality for flow calculations, as long as enough grid-points are chosen to prevent discrete grid folding. For planar blocks we use typically 30×30 points. This makes interactive elliptic grid generation possible, at least in many practical cases.

Without laying claim to a rigorous mathematical foundation of this method, one may justify this concept by the remark, that basically elliptic grid generation is founded on the minimization of the functional

$$\int_{\Omega} u_{xx}^2 + v_{yy}^2 \longrightarrow \min \tag{80}$$

and, as pointed out above, B-Splines fulfill a similar variation diminishing property, at least in the functional 1D-case.

The following example is taken from [41]. Using the algorithms proposed by Spekreijse we first generated an elliptic grid in this domain demanding orthogonality and control of the first layer of cells along the curved borders.We have discretized the domain with 100×100 grid-points (only every second line is plotted for the sake of better visualization) and then compared it with the grid obtained when we discretize the elliptic equation with 30×30 points, and evaluate the interpolating spline again at 50×50 points. The smoothness of both grids is nearly the same, only the high resolution elliptic grid produced a somewhat thinner cell layer at the right boundary of the domain.

4.5.4 Reproduction of Grids

The approximation tools enable us to reproduce grids from external sources automatically, for example pointwise given grids used for calculation with other solvers than QUADFLOW. Within this project for example we are often asked to compare

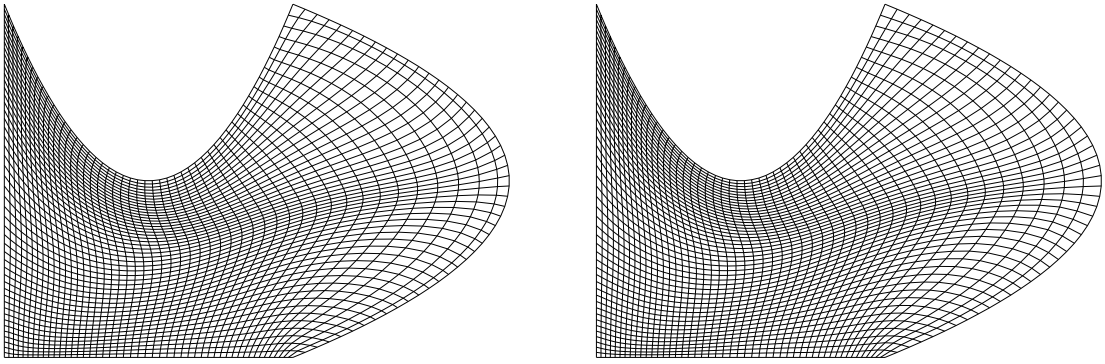


Figure 16 Fine and coarse elliptic grid

the results of QUADFLOW with former results produced by the FLOWer code. In order to avoid the work of generating block-decompositions ourselves, we have automated the task of converting the pointwise given grids into B-Spline representations. Thereby only one constraint must be respected. In practical grid generation often grids are given that contain non-differentiable corners in one boundary. The simplest example is a ramp that is connected at a certain angle with a plain surface.

This is a technical problem, because cubic B-Splines without multiple interior knots are C^2-continuous. It is possible to model such corners with $p-1$ coalescing control points [32]. But then the derivative in this point would vanish what may result in grid folding near this point. Another method is to use knot vectors with multiple knots, but in this case further manipulation of this spline is more difficult, because evaluating the spline at its knots and reproducing it again by interpolation as proposed at the top of this section is not possible if the knot vector contains multiple interior knots. So the simplest method is to split the block in question into two and to apply the methods described above in both blocks separately. Since it is not easy to decide from the discrete set of points automatically, where such cusps occur, a certain extent of user interaction remains.

4.5.5 Extensions

It is clear that the detour described above, namely first to evaluate the spline at certain points and then to interpolate these points again, can only serve as provisional strategy. It has been chosen in order to make use of already existing tools. Future developments will aim at directly manipulating the B-Spline control points which opens particularly promising perspectives with regard to temporally moving meshes.

The extension of this concept to 3D is straight forward. Here we assume that the simply connected domain Ω is bounded by six faces, given as B-Spline surfaces. The volume grid is represented as 3-dimensional tensor product. So the whole geometry is processed within a unified framework.

4.6 Offset-Curves

In this section we describe how we compute the offset curves required for the boundary fitted blocks, see also [27].

Let a regular planar curve C be explicitly/parametrically given by

$$\boldsymbol{x}(u) = \begin{pmatrix} x(u) \\ y(u) \end{pmatrix} : [u_0, u_1] \longrightarrow \mathbb{R}^2 \tag{81}$$

The curvature of the curve C is given by (for simplicity we omit the arguments)

$$\kappa = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} = \frac{\begin{vmatrix} \dot{x} & \dot{y} \\ \ddot{x} & \ddot{y} \end{vmatrix}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}. \tag{82}$$

Now we can define curvature dependent offset curves. Consider the curve as a front $\boldsymbol{x}(u,t)$ propagating along its normal direction with curvature dependent speed $F(\kappa)$.

$$\boldsymbol{x}(u,0) = \boldsymbol{x}(u), \tag{83}$$

$$\frac{\partial}{\partial t}\boldsymbol{x}(u,t) = F(\kappa(u))\boldsymbol{n}(u). \tag{84}$$

This idea was introduced to grid generation by Sethian [42]. To compute the evolution of the curve, Sethian embeds the interface as the zero level set of a higher dimensional function and solves a partial differential equation to follow its propagation. The advantage of the level set approach is primarily that topological changes such as breaking and merging of several fronts are well defined and performed without additional effort. But since Sethian's method leads to complex and time consuming algorithms, and topological changes cannot occur in the present context, we use instead the parametric B-Spline representations for the offset surfaces. So we achieve a stable, easy to implement and fast algorithm that is more suited to the needs of grid generation.

We assume, that a B-Spline representation $\hat{\boldsymbol{x}}_{t_i}(u)$ of the propagating front at time level t_i is given. We want to calculate a B-Spline representation $\hat{\boldsymbol{x}}_{t_{i+1}}$ at the next time-level $t_{i+1} = t_i + \Delta t$. Therefore we just choose an appropriate, i.e. curvature dependent set of parameter values u_j – they may differ in every time-step – and compute sampling points for the next offset curve by

$$\hat{\boldsymbol{x}}_{t_{i+1}}(u_j) = \hat{\boldsymbol{x}}_{t_i}(u_j) + \Delta t\, F(\kappa(u_j))\boldsymbol{n}(u_j). \tag{85}$$

Remember that the relevant geometric quantities, curvature and normals, can be exactly computed from the B-Spline representation. These sampling points are approximated afterwards by a new B-Spline using the approximation tools and fairing methods from Section 4.4.3, in order to guarantee that all level sets become optimally smooth.

We have tested the same functions F as used in [42], namely

(i) $F(\kappa) = 1 - \varepsilon\kappa$

(ii) $F(\kappa) = e^{-\varepsilon\kappa}$

(iii) $F(\kappa) = \max\{1 - \varepsilon\kappa, F_{threshold}\}$.

Here $F_{threshold} > 0$ is chosen to prevent the front from moving inwards. In [42] ε and $F_{threshold}$ are computed only once. With our method their computation is very cheap so we decided to adapt them in every step to use an optimal step size Δt.

Two examples are given in Figure 17. The left plot shows several offset curves around the SFB 401 reference profile, but not the final grid. The lines in cross direction only symbolize curvature dependent sampling points. To achieve good grid quality after the computation of the offset curves one can add smooth iso-lines orthogonal to the surfaces. For this purpose a hyperbolic method based on the solution of ordinary differential equations has been developed in [28]. The right plot demonstrates the robustness of the method with the aid of an academic but more difficult example. It is clear, that because of the concave parts of the boundary the usual techniques from CAD to generate parallel curves would fail here. Additionally the flexibility of splines with regard to boundary conditions was used to enforce $GC2$-periodicity across the cut, yielding a smooth closed curve.

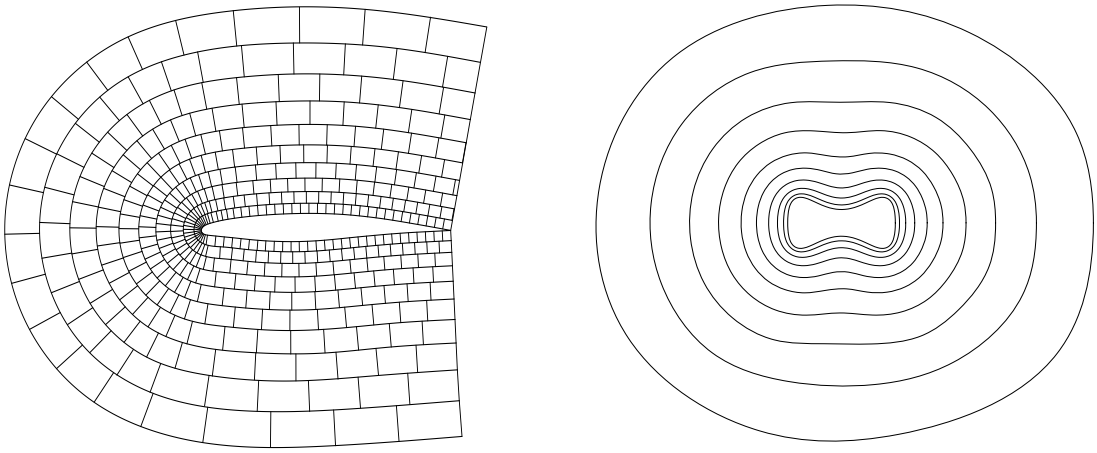


Figure 17 Offset-Curves around cruise configuration (left) and artifical example (right)

The last example given by Figure 18 demonstrates that the B-Spline method is robust enough to handle even non-differentiable corner points. The initial configuration consists of two circles connected by a straigth line. This example might seem to be somewhat artificial, but really is of practical relevance, because similar situations occur in practice, when one wants to generate offset curves around multiple connected domains. In a first step a smooth curve approximating the intitial configuration is defined applying the knot insertion algorithm near the corner points.

Approximatively 100 control points are needed to represent this curve. After that the above algorithm is applied. Here it is particulary important, that one can choose an adopted, optimally large stepsize independently in every time step, because in the beginning only very small timesteps can be performed.

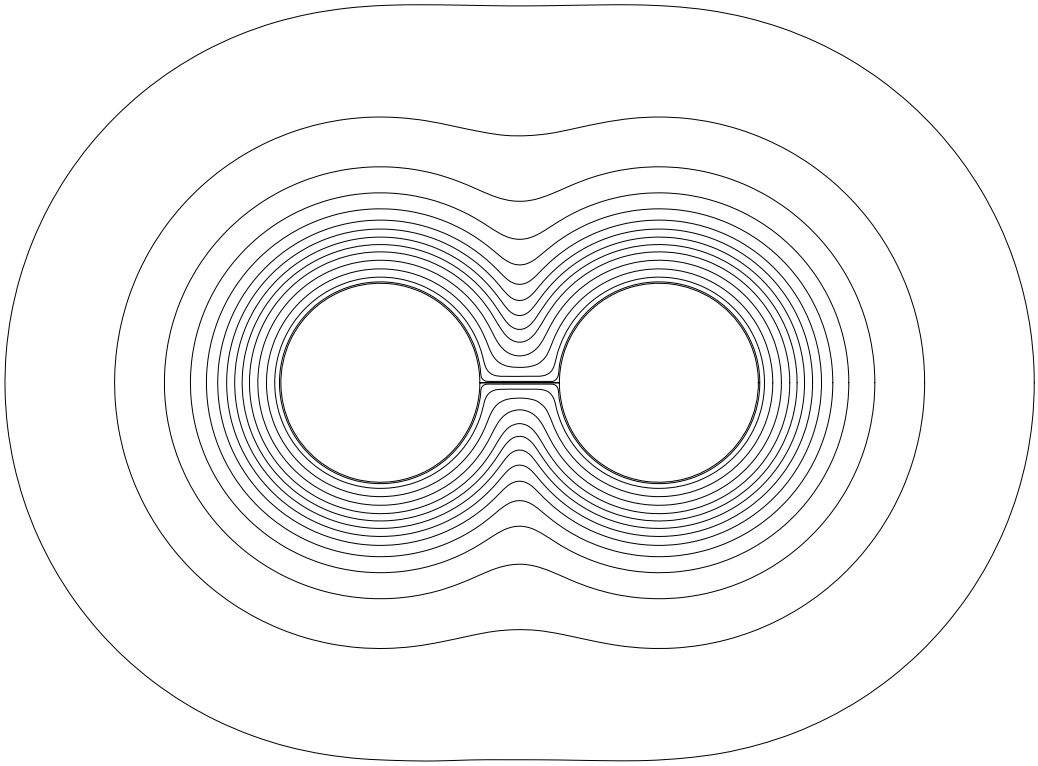


Figure 18 Offset-Curves around double circle

The extension of this method to 3D is straight forward. From a parametrically given surface

$$\boldsymbol{x}(u,v) : [u_0, u_1] \times [v_0, v_1] \longrightarrow \mathbb{R}^3 \tag{86}$$

one can first compute the relevant geometric quantities, then the sampling points for the next offset curve, and after that approximate these by a smooth B-Spline surface. In this case we have more freedom, as one can choose between several notions of curvature, for example mean curvature, Gaussian curvature or principal curvatures. Of course, the primary selection criterion for a suitable curvature notion is to prevent grid folding. First results with this method are promising and will be subject of further investigations.

5 Discretization of the Navier-Stokes Equations

5.1 Finite Volume Method

The occurance of hanging nodes due to local mesh adaptation poses particular difficulties concerning the discretization of the governing equations. In the following section we present a finite volume method, the special merits of which are its flexibility to operate on meshes of any arbitrary topology. This approach offers a unified way to incorporate hanging nodes. Its main ingredients will be discussed in detail, including data structures, realization of spatial second order accuracy, treatment of convective and viscous fluxes, choice of limiters, treatment of boundary conditions and implicit time integration. The section will be concluded by the validation of the basic scheme applying it to a proper selection of test cases for inviscid and viscous flows.

In the present study, we will concentrate on the discretization of the governing equations in two space dimensions. Its extension to three space dimensions is subject to current work.

5.1.1 Data Structure

The discretization of the governing equations (1) is based on a cell centered finite volume scheme. The grid is treated as a fully unstructured mesh, composed of simply connected elements with otherwise arbitrary topology. In particular, the flow solver is not restricted to operate on locally refined quadrilateral cells, but may be combined with any grid topology, e.g. mixed element type grids. This flexibility is crucial for supporting the adaptive concept.

Different element types are processed in a unified manner, rather than being distinguished. This property is often related to as *grid transparency*. Hanging nodes, which occur due to local adaptation, do not require any special treatment. The use of transition elements at refined interfaces is not necessary.

The data structure of the flow solver is primarily based on the faces of the grid. A face based data structure has the advantage that there are no limitations on the number of faces, which can be connected to a cell, see Figure (19) for illustration. The evaluation of fluxes and their contribution to cells can be efficiently implemented by sweeps over the faces.

The mesh is composed of the basic grid objects: cells, faces and nodes, see Figure (20). Grid objects are related to each other via connectivity lists. Two type of pointers are required: A link between faces and cells, that share the face in question, and a connection between faces and nodes, that belong to the face:

	Pointer type	Data structure
•	Face to cell	`face_2_cell(iface,iC)`
•	Face to node	`face_2_node(iface,iN)`

where `iC` $\in [1,2]$ denotes the left and right neighbour of the face and `iN` $\in [1,2]$ represents the adjoining nodes, respectively. See also Figure (21) for illustration.

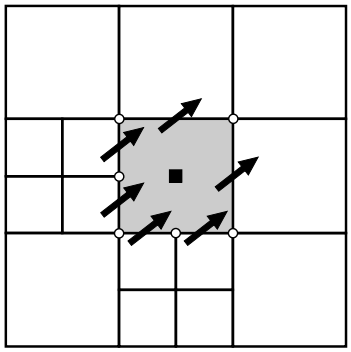


Figure 19 Collection of fluxes for polygonally bounded control volume in 2D

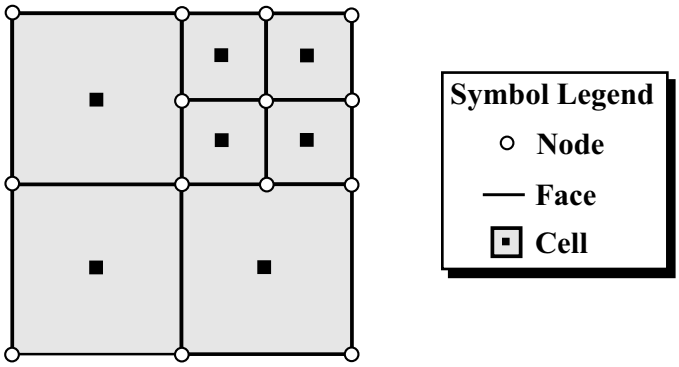
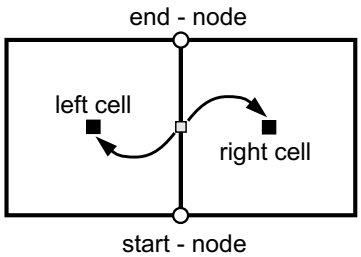
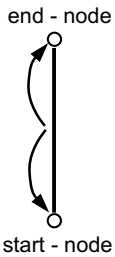


Figure 20 Basic grid objects of unstructured mesh



a) Connectivity between faces and cells



b) Connectivity between faces and nodes

Figure 21 Logical relation between different grid objects

5.1.2 Discretization of Conservative Fluxes

The conservative fluxes are determined by solving quasi-one-dimensional Riemann problems at cell interfaces. The rotational invariance of the Euler equations

$$\mathbf{F}^c\left(\mathbf{u}\right)\mathbf{n} = \mathcal{R}^T\left(\mathbf{n}\right)\mathbf{F}^c\left(\tilde{\mathbf{u}}\right) \tag{87}$$

is utilized to express the projection of the conservative flux into normal direction, where $\tilde{\mathbf{u}} = \mathcal{R}\left(\mathbf{n}\right)\mathbf{u}$ and $\mathcal{R}$ represents the rotational matrix

$$\mathcal{R}\left(\mathbf{n}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & n_x & n_y & 0 \\ 0 & -n_y & n_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \tag{88}$$

Within the present paper, the conservative flux function is based on the HLLC scheme proposed by Batten and Leschziner [46]. The method is capable of exactly preserving isolated shocks, contact, and shear waves, which are desired attributes to predict viscous flows accurately. Further, the entropy condition is inherently enforced by the scheme, i.e. neither an entropy correction nor a shock fix is required.

To define the HLLC scheme, we first introduce the following quantities:

$$S_L = min\left[\lambda_1\left(\mathbf{u}_l\right), \lambda_1\left(\mathbf{u}^{Roe}\right)\right], \; S_R = max\left[\lambda_m\left(\mathbf{u}^{Roe}\right), \lambda_m\left(\mathbf{u}_r\right)\right] , \tag{89}$$

$$S_M = \frac{\varrho_r v_{n,r}\left(S_R - v_{n,r}\right) - \varrho_l v_{n,l}\left(S_L - v_{n,l}\right) + p_l - p_r}{\varrho_r\left(S_R - v_{n,r}\right) - \varrho_l\left(S_L - v_{n,l}\right)} , \tag{90}$$

$$\Omega_l = \left(S_L - S_M\right)^{-1} , \tag{91}$$

$$p^* = \varrho_l\left(v_{n,l} - S_L\right)\left(v_{n,l} - S_M\right) + p_l . \tag{92}$$

The subscripts l and r denote the left and right state of the Riemann problem. $\lambda_1\left(\mathbf{u}^{Roe}\right)$ and $\lambda_m\left(\mathbf{u}^{Roe}\right)$ represent the smallest and largest eigenvalues of the Roe matrix [47] and v_n, v_t are the velocity components normal and tangential to a cell interface, respectively.

Introducing the intermediate vector of state $\mathbf{u}_l^*$:

$$\mathbf{u}_l^* = \begin{pmatrix} \varrho_l^* \\ \left(\varrho v_n\right)_l^* \\ \left(\varrho v_t\right)_l^* \\ \left(\varrho e_{tot}\right)_l^* \end{pmatrix} = \Omega_l \begin{pmatrix} \varrho_l\left(S_L - v_{n,l}\right) \\ \left(S_L - v_{n,l}\right)\left(\varrho v_n\right)_l + p^* - p_l \\ \left(S_L - v_{n,l}\right)\left(\varrho v_t\right)_l \\ \left(S_L - v_{n,l}\right)\left(\varrho e_{tot}\right)_l - p_l v_{n,l} + p^* S_M \end{pmatrix} , \tag{93}$$

the numerical flux function is expressed as follows:

$$\mathbf{F}_{HLLC} = \begin{cases} \mathbf{F}_l & \text{if } S_L > 0 \\ \mathbf{F}\left(\mathbf{u}_l^*\right) & \text{if } S_L \leq 0 < S_M \\ \mathbf{F}\left(\mathbf{u}_r^*\right) & \text{if } S_M \leq 0 \leq S_R \\ \mathbf{F}_r & \text{if } S_R < 0 \end{cases} \tag{94}$$

with

$$\mathbf{F}\left(\mathbf{u}_l^*\right) = \begin{pmatrix} \varrho_l^* S_M \\ \left(\varrho v_n\right)_l^* S_M + p^* \\ \left(\varrho v_t\right)_l^* S_M \\ \left(\left(\varrho e_{tot}\right)_l^* + p^*\right) S_M \end{pmatrix} . \tag{95}$$

$\mathbf{F}_l$ and $\mathbf{F}_r$ represent the conservative flux $\mathbf{F}^c(\mathbf{u})$, evaluated at the corresponding state. The relevant expressions for $\mathbf{F}\left(\mathbf{u}_r^*\right)$ are obtained by interchanging the subscripts $l \leftrightarrow r$ and $L \leftrightarrow R$.

5.1.3 Higher Order Method in Space

In order to obtain second-order accuracy in space, for each control volume Ω_i a linear reconstruction of the primitive flow variables $w \in \{\varrho, u, v, p\}$ is determined:

$$w\left(\mathbf{x}\right)\big|_{\Omega_i} = w_i + \varphi_i \left(\mathbf{x} - \mathbf{x}_i\right)^T \cdot \nabla w_i \,, \tag{96}$$

where w_i represents its mean value at the centroid $\mathbf{x}_i$ of Ω_i and φ_i denotes a limiter function. The approximate gradient ∇w_i is calculated by a multi-dimensional, second order truncated Taylor series expansion around the centroid of Ω_i

$$w_j = w_i + \left(\mathbf{x}_j - \mathbf{x}_i\right)^T \cdot \nabla w_i, \; j \in N_i \,. \tag{97}$$

N_i is the set of neighbouring cells of Ω_i, which support the reconstruction of w. The fixed stencil N_i consists of cells, that share a face with Ω_i (face neighbours).

In practice, equation (97) represents an overdetermined system of equations for the gradient of w_i. It is solved in a least squares sense, using normal equations. To improve the condition of the problem, the difference vector $\left(\mathbf{x}_j - \mathbf{x}_i\right)$ is locally rescaled independently in x and y direction

$$x_j - x_i \rightarrow \left(x_j - x_i\right)/\tilde{x}_i \,, \quad y_j - y_i \rightarrow \left(y_j - y_i\right)/\tilde{y}_i \,.$$

$\tilde{x}_i$ and $\tilde{y}_i$ denote the width of the cell in the respective coordinate direction, which is measured as the maximum distance between the cell centroid and the corresponding midpoint of the faces, that form the control volume in question.

Numerical experiments have demonstrated, that solving the least squares problem by means of a numerically more robust Householder transformation does not yield higher accuracy for the considered applications, but is significantly more expensive.

Alternatively to the least squares method, the gradient ∇w_i may be computed by using the Green-Gauss theorem

$$\nabla w_i = \frac{1}{\Gamma_i} \oint_{\partial \Gamma_i} w \, \mathbf{n} \, d\partial\Gamma_i \tag{98}$$

where Γ_i denotes an auxiliary control volume, which is defined by the centroids of the neighbouring cells of Ω_i, which support the reconstruction.

In the current study, the Green-Gauss technique is only used in case of the so-called Ringleb flow, discussed in Section 5.2.1. Everywhere else in the paper, reconstruction is based on the least squares method.

5.1.4 Monotonicity Enforcement

At local extrema and discontinuities, the recovery polynomial may generate new extrema and therefore yield oscillations in the numerical solution. In order to circumvent this problem, limiter functions with TVD (total variation diminishing) property are used. The slope limiter by Venkatakrishnan [48] is employed. which is defined as follows:

$$\varphi_{i,g} = \begin{cases} \dfrac{w_{i,g}^{+^2} + 2w_{i,g}^{+} w_{i,g}^{-} + \varepsilon}{w_{i,g}^{+^2} + w_{i,g}^{+} w_{i,g}^{-} + 2w_{i,g}^{-^2} + \varepsilon}, & \text{if } w_{i,g}^{-} \neq 0; \\ 1 \qquad\qquad\qquad\qquad\quad , & \text{if } w_{i,g}^{-} = 0; \end{cases} \tag{99}$$

with

$$\begin{aligned} w_{i,g}^{-} &= w_g - w_i, \\ w_{i,g}^{+} &= \begin{cases} w_i^{max} - w_i, & \text{if} \quad w_{i,g}^{-} > 0 \\ w_i^{min} - w_i, & \text{if} \quad w_{i,g}^{-} < 0 \end{cases} \\ w_i^{max} &= max_{j=1,\ldots,N_i} \left\{ w_i, w_j \right\}, \\ w_i^{min} &= min_{j=1,\ldots,N_i} \left\{ w_i, w_j \right\}. \end{aligned}$$

w_g denotes the unlimited reconstructed value of w at the Gauss quadrature point g. The coefficient ε is typically set to $\varepsilon = 10^{-4}$ in our applications. Since condition (99) has to be fulfilled in every quadrature point of the faces, the final limiter associated with the control volume Ω_i is taken as the corresponding minimum value $\varphi_i = min\,(\varphi_{i,g})$.

5.1.5 Discretization of Viscous Fluxes

At cell interfaces, the gradients of the velocity and temperature are determined by utilizing the divergence theorem. As proposed by Coirer [49], a secondary volume is

introduced, the edges of which are formed by connecting the two vertices of an edge with the centroids of the cells, that share the interface in question, see Figure 22(a). It is commonly referred to as “diamond path”. At refined interfaces, the standard diamond path due to Coirer leads to a non-symmetric control volume, which may result in a loss of accuracy. Therefore, in case of locally adapted meshes we use a modified diamond path proposed by Delanaye [50], which maintains a regular symmetric shaped control volume even in the presence of hanging nodes. To apply the divergence theorem, an interpolation is required to evaluate the flow quantities at the vertices. The interpolation is supported by the set of cells N_{v_j} that share the vertex v_j in question, see Figure 22(b). A linearity-preserving procedure based on the pseudo-Laplacian formula according to Holmes and Connell [51] is used to determine the averaged quantity $\widetilde{w}$ at the vertex v_j:

$$\widetilde{w}_{v_j} = \frac{\sum_{i=1}^{N_{v_j}} \xi_i w_i}{\sum_{i=1}^{N_{v_j}} \xi_i}, \tag{100}$$

where w_i represents the corresponding flow quantity at the cell centroids $i \in N_{v_j}$. The dimensionless weights ξ_i are given by

$$\xi_i = 1 + \lambda_x (x_i - x_{v_j}) + \lambda_y (y_i - y_{v_j}) \tag{101}$$

with

$$\lambda_x = \frac{I_{xy} R_y - I_{yy} R_x}{I_{xx} I_{yy} - I_{xy}^2}, \tag{102}$$

$$\lambda_y = \frac{I_{xy} R_x - I_{xx} R_y}{I_{xx} I_{yy} - I_{xy}^2} \tag{103}$$

where the moments I_{xx}, I_{yy}, I_{xy}, R_x and R_y are defined as:

$$R_x = \sum_{i=1}^{N_{v_j}} (x_i - x_{v_j}), \tag{104}$$

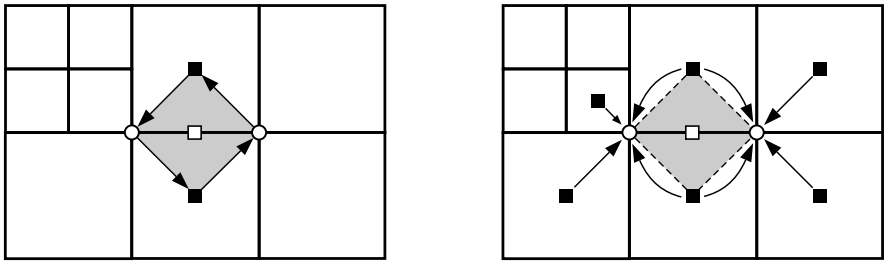
$$R_y = \sum_{i=1}^{N_{v_j}} (y_i - y_{v_j}), \tag{105}$$

$$I_{xx} = \sum_{i=1}^{N_{v_j}} (x_i - x_{v_j})^2, \tag{106}$$

$$I_{yy} = \sum_{i=1}^{N_{v_j}} (y_i - y_{v_j})^2, \tag{107}$$

$$I_{xy} = \sum_{i=1}^{N_{v_j}} (x_i - x_{v_j})(y_i - y_{v_j}). \tag{108}$$

At the midpoint of each face, the velocity and temperature are averaged from the corresponding quantities at the vertices.



(a) Diamond path due to Coirer **(b)** Averaging procedure at grid nodes

Symbol legend: ■ cell center, ∘ node, □ face midpoint

Figure 22 Calculation of gradients at cell interfaces, using divergence theorem

5.1.6 Boundary Conditions

Euler Walls

At solid walls at rest, the contact condition $\mathbf{v} \cdot \mathbf{n} = 0$ is imposed via the modification of the conservative fluxes through the boundary face. The contact condition yields the normal flux formula

$$\mathbf{F}^c \cdot \mathbf{n}\big|_{wall} = (0, p \cdot \mathbf{n}, 0)^T \ , \tag{109}$$

where the static pressure p is extrapolated from the interior domain, using the multi-dimensional reconstruction.

Navier–Stokes Walls

The treatment of Navier–Stokes walls is based on an approach proposed by Anderson and Bonhaus [52], which has also successfully been used e.g. by Geuzaine [53] in the context of turbulent flows. At wall boundary cells, the coordinate location that is associated with the vector of unknowns, is shifted from the cell centroid to the midpoint of the corresponding wall edge. The governing equations directly express the temporal evolution of the flow quantities on the wall itself. This approach offers several advantages: First, at every time step the boundary conditions are strongly enforced for the wall boundary cells, rather than just being considered through the modification of the fluxes. Second, the boundary conditions are implemented in such a way that no assumption on the pressure distribution is required. The latter is often utilized in other approaches, which are based on modification of the fluxes or some mirror principle.

No-slip condition: The no-slip condition $\mathbf{v} = \mathbf{0}$ on a wall at rest is imposed by modification of the momentum equations

$$\int_\Omega \frac{\partial \varrho \mathbf{v}}{\partial t}\Big|_{wall} dV = \mathbf{0} \tag{110}$$

with initial conditions $\varrho\mathbf{v}\,(t = 0,\, \mathbf{x}_{wall}) = \mathbf{0}$.

Isothermal wall: In the present study, only isothermal walls are considered. In order to impose a constant wall temperature T_{wall}, the energy equation is modified in such a way that during the temporal evolution of the flow quantities, a constant wall temperature is automatically maintained. This approach does not require any assumption on the pressure distribution, e.g. $\partial p/\partial n = 0$, which is actually a predicate of a component of the gradient of pressure appearing explicitly in the local balance of moment and therefore not truly appropriate as a boundary condition in a boundary value problem.

The temperature at the wall is related to the internal energy of the fluid via

$$T_{wall} = \frac{1}{c_v} e_{wall} = \frac{1}{c_v} \frac{\varrho e_{tot}}{\varrho}\Big|_{wall}, \tag{111}$$

because here the specific internal energy e is locally equivalent to the total energy e_{tot}, due to the no-slip condition. By means of eq. (111), the temporal evolution of the energy is expressed in terms of the temporal evolution of the density:

$$\int_\Omega \frac{\partial \varrho e_{tot}}{\partial t}\Big|_{wall} dV = c_v T_{wall} \int_\Omega \frac{\partial \varrho}{\partial t}\Big|_{wall} dV\,, \tag{112}$$

where the temporal evolution of the density is determined by the continuity equation.

Far-Field Boundaries

At artificial boundaries of the computational domain which replace the true far–field boundaries, the type of boundary condition of each individual face may either be explicitly prescribed, or fully automatically detected, depending on the local flow solution. The latter approach is particularly useful for C-type grids.

Two different approaches are used here to treat subsonic inflow and outflow conditions. The first one is based on standard characteristic boundary conditions [54]. Outgoing Riemann invariants are determined by quantities being extrapolated from the interior domain, while incoming Riemann invariants are set by free stream conditions.

The second approach, which is especially suited for stationary planar flow, is based on a point vortex correction according to Thomas and Salas [55]. In this case, the flow quantities at far–field are specified by a solution according to full potential flow

theory. The corrected velocity components are prescribed by

$$u = |\mathbf{v}_\infty| \, cos\alpha + V_\Gamma \, sin\theta \,, \tag{113}$$
$$v = |\mathbf{v}_\infty| \, sin\alpha - V_\Gamma \, cos\theta \,. \tag{114}$$

The vortex induced velocity magnitude V_Γ is given by

$$V_\Gamma = \frac{\Gamma}{2\pi r} \frac{\sqrt{1 - M_\infty^2}}{1 - M_\infty^2 sin^2\left(\theta - \alpha\right)} \tag{115}$$

where the circulation Γ of the vortex is determined by $\Gamma = 1/2 c_{ref} \, |\mathbf{v}_\infty| \, C_L$. c_{ref} denotes the chord length and C_L is the lift coefficient. r and θ are the radius and the polar angle, measured from the airfoil quarter–chord to the individual boundary face at the far–field, respectively. The remaining flow quantities are determined under the assumption of isentropic flow and constant total enthalpy.

5.1.7 Time Integration

For steady fluid flow time plays the role of an iteration parameter to achieve asymptotically stationary flow in the computation. The solution is advanced in time by an implicit Euler method. Local time steps are applied. For the flux balance of the cell Ω_i we define the so-called residual vector

$$\mathbf{R}_i := \oint_{\partial\Omega_i} \mathbf{F}^c\left(\mathbf{u}\right) \mathbf{n} \, dS \,. \tag{116}$$

Introducing the definition

$$\widehat{\mathbf{R}}^{n+1}\left(\mathbf{u}^{n+1}\right) := \mathbf{R}^{n+1}\left(\mathbf{u}^{n+1}\right) + \frac{\Omega}{\Delta t}\left(\mathbf{u}^{n+1} - \mathbf{u}^n\right) \tag{117}$$

the conservation laws (1) are expressed in semi-discrete form as

$$\widehat{\mathbf{R}}^{n+1}\left(\mathbf{u}^{n+1}\right) = \mathbf{0} \,. \tag{118}$$

Equation (118) is solved by a Newton scheme. For stationary flows we take one Newton iteration per physical time step. In this case the Newton scheme is:

$$\widehat{\mathcal{J}}\left(\mathbf{u}^n\right) \Delta\mathbf{u}^n = -\mathbf{R}\left(\mathbf{u}^n\right) \tag{119}$$

with

$$\widehat{\mathcal{J}} = \frac{\Omega}{\Delta t}\mathcal{I} + \left.\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right|^n \,, \quad \Delta\mathbf{u}^n := \mathbf{u}^{n+1} - \mathbf{u}^n \,. \tag{120}$$

The linear system of equations (119) is solved by an iterative Krylov subspace method. In this study, we employ the GMRES algorithm [56], preconditioned by an incomplete LU-factorization. Typically, a Krylov subspace method requires only

the inner product of a Jacobian matrix $\mathcal{J}$ and a vector $\boldsymbol{\nu}$, and not $\mathcal{J}$ explicitely [57]. This product can also be approximated by a difference quotient of the form

$$\mathcal{J}(\mathbf{u})\,\boldsymbol{\nu} \approx \frac{\mathbf{R}(\mathbf{u}+\varepsilon\boldsymbol{\nu}) - \mathbf{R}(\mathbf{u})}{\varepsilon}, \quad \text{with} \quad \varepsilon \in \mathbb{R}. \tag{121}$$

The difference parameter ε is chosen as

$$\varepsilon = \hat{\varepsilon}\frac{\sqrt{1.0 + ||\mathbf{u}||_2}}{||\boldsymbol{\nu}||_2} \tag{122}$$

with $\hat{\varepsilon} = 10^{-8}$ in our application. The advantage of using a matrix free version of a Krylov subspace method is, that the matrix vector product may consistently be discretized according to the evaluation of the residual vector, that represents the right hand side of equation (119). Nevertheless, a Jacobian is still required for the application of the $ILU(n)$ pre-conditioner. The pre-conditioning matrix is based on a first order accurate scheme in space. The Jacobian is derived numerically [58], using one sided difference operators of the form:

$$\frac{\partial \mathbf{F}_i(\mathbf{u})}{\partial \mathbf{u}_j} = \lim_{\tilde{\varepsilon}\to 0} \frac{\mathbf{F}_i(\mathbf{u}+\tilde{\varepsilon}\mathbf{e}_j) - \mathbf{F}_i(\mathbf{u})}{\tilde{\varepsilon}}, \quad \text{with} \quad \tilde{\varepsilon} \in \mathbb{R}, \tag{123}$$

where $\mathbf{F}_i$ is the i-*th* component of the numerical flux vector and $\mathbf{e}_j$ is the j-*th* unit vector, associated with the dependent variable $\mathbf{u}_j$.

Presently, viscous terms are not included within the Jacobian. For the computation of inviscid flows presented in this paper, we employ the standard GMRES algorithm using a Jacobian derived by numerical means. For viscous flows, we utilize a matrix–free GMRES, which consistently accounts for the viscous terms within the matrix vector product. In this case, only the pre-conditioner suffers from neglecting the viscous terms.

The infrastructure related to the Newton-Krylov method in this paper is based on the PETSc [59] library of Argonne National Laboratory. For further details of the numerical method we refer to [60], [61].

5.2 Validation of the Basic Scheme

In the following, various test cases are considered using regular, non-adaptive structured grids in order to validate the basic finite volume scheme. Particular emphasis is put on to the spatial accuracy of the method. For inviscid flows, the Ringleb flow is investigated, for which an analytic solution is available. For viscous flows, the laminar flow over an isothermal flat plate is considered, for which the similarity solution due to Blasius serves as a reference. Further, the transonic flow about an airfoil according to the SFB 401 cruise configuration is used to verify the method in the presence of shocks. In this case, results obtained with the FLOWer code are used for comparison. Finally, convergence acceleration to steady state using implicit time integration is investigated.

5.2.1 Ringleb flow

The so-called Ringleb flow [62] has been used as a test case to assess the spatial accuracy of the basic scheme for inviscid flows. It is an exact solution of the Euler equations obtained by hodograph transformation. The flow depends on the inverse of the stream function k and the velocity magnitude q. In our application we choose the values $k = 0.4$ and $k = 0.8$ to define the rigid walls, and $q = 0.3$ to define the inlet- and outlet boundaries. The flow is fully subsonic within the domain. The inlet Mach number is 0.3027 and the maximum Mach number reaches a value of 0.8567 at the center of the bottom wall. The error E_{h_j} and the spatial order of the scheme *EOC* (*Experimental Order of Convergence*) are estimated in the L_1 norm of the Mach number:

$$E_{h_j} = \frac{1}{N_j} \sum_{i=1}^{N_j} \left| f_{h_j} - 1 \right|, \; EOC = \frac{log\left(\frac{E_{h_j}}{E_{h_{j+1}}}\right)}{log\left(\frac{h_j}{h_{j+1}}\right)}$$

where f_{h_j} represents the ratio of the numerical solution and the exact solution on a grid with the characteristic length scale $h_j = 1/\sqrt{N_j}$; N_j is the number of grid cells. The accuracy analysis is carried out on four different grids with a resolution of 16x4, 32x8, 64x16 and 128x32 cells, respectively. In addition, the grids are distorted randomly in order to assess the sensitivity of the solution with regard to mesh quality, see Figure 23. Figure 24 illustrates the Mach number distribution along the bottom wall for three different regular grids. The linear Green-Gauss reconstruction method according to eq. (98) has been employed. For the highest resolution, the maximum error between the numerical solution and the exact solution is about 0.02%. Figure 25 shows the error E_{h_j} in the Mach number, depending on the characteristic length scale h_j of the mesh. A constant and a linear reconstruction, based on the Green-Gauss method, have been considered. For the linear reconstruction, the sensitivity of the error with regard to grid distortion is weak. The order of accuracy can be estimated between 2.5-2.9 for the linear reconstruction, see Table 1. The constant reconstruction scheme is numerically less then first order accurate for the considered grids.

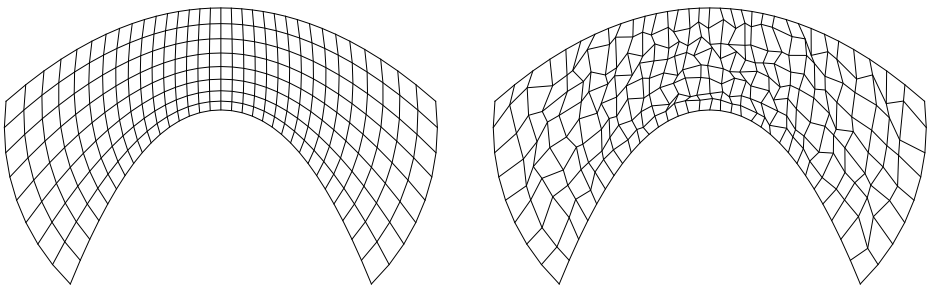


Figure 23 Ringleb flow: Grids (32x8 cells); left figure: regular grid; right figure: randomly distorted grid

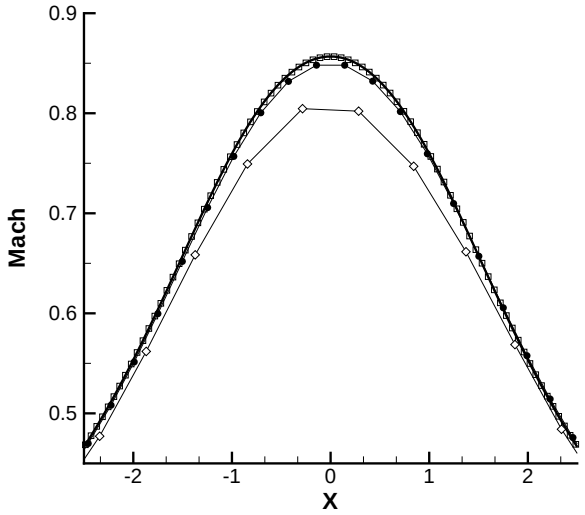


Figure 24 Ringleb flow: Mach number distribution at bottom wall (partial view); linear Green-Gauss reconstruction on regular grids; – exact solution, □128x32 cells, •32x8 cells, ⋄16x4 cells

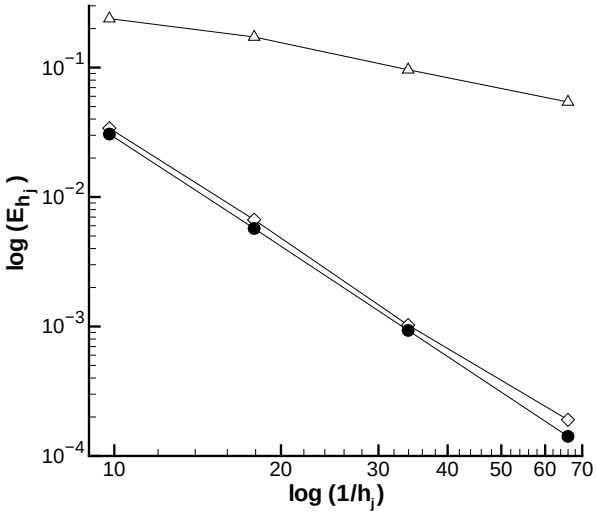


Figure 25 Ringleb flow: Error in the Mach number as function of the characteristic length scale h_j of the mesh. Linear Green-Gauss reconstruction: • regular grid, ⋄ distorted grid ; △ constant reconstruction (regular grid)

Table 3 Ringleb flow: experimental order of convergence (EOC) for linear reconstruction and constant reconstruction on regular and distorted grids

	EOC		
grid	linear rec. regular	linear rec. distorted	constant rec. regular
32x8	2.79	2.70	0.543
64x16	2.83	2.93	0.909
128x32	2.84	2.53	0.862

5.2.2 Laminar Boundary Layer

The next test case considers the laminar flow over an isothermal flat plate. In order to reduce compressibility effects, a free stream Mach number of $M_\infty = 0.2$ is chosen. The Reynolds number is $Re_\infty = 10^4$, based on unit length. The wall is considered as isothermal, with $T_{wall} = T_\infty = 273.0K$. A linear dependence of the molecular viscosity on the temperature is assumed. The Prandtl number is $Pr = 1$. For purpose of validation, the similarity solution according to Blasius [63] for an incompressible laminar fluid flow serves as a reference.

A structured, non-adaptive grid with 176×56 cells (9856 cells all together) is considered, see Figure 26. The plate extends on the x-axis between $x = 0.0$ and $x = 2.0$, with 96 cells located on the plate itself. Upstream of the leading edge, the lower boundary of the domain is modeled as an inviscid impermeable wall. The first grid spacing normal to the wall is 10^{-3}, which equals $y^+ \approx 1$ at $Re = 10^4$. The grid is clustered about the leading edge, measuring a first grid spacing of 10^{-3} in streamwise direction. Far-field boundary conditions are applied as follows: At inflow, the temperature and velocity vector are prescribed, while the static pressure is extrapolated from the interior domain. At outflow, the static pressure is prescribed and the density as well as the velocity vector are extrapolated from the interior domain. The upper boundary of the domain is treated as an outflow.

Figure 27(a) and Figure 27(b) present a comparison between the computed u-velocity and v-velocity profiles with the theoretical solution at $Re_x = 9169$, respectively. Both velocity components are predicted accurately, compared with the theoretical solution. At the outer part of the boundary layer ($\eta > 4$) the v-velocity is slightly overpredicted by the numerical scheme.

Figure 28 illustrates the evolution of the skin friction coefficient along the plate (*please note*: the graph has been truncated for $Re < 1000$). The computed skin friction agrees well with the Blasius solution.

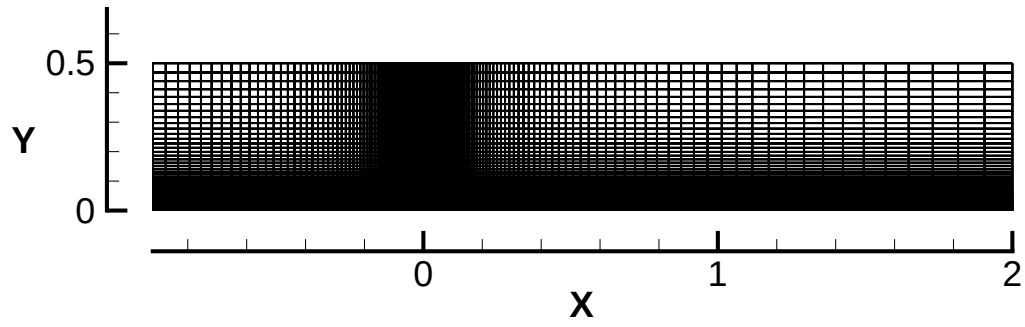
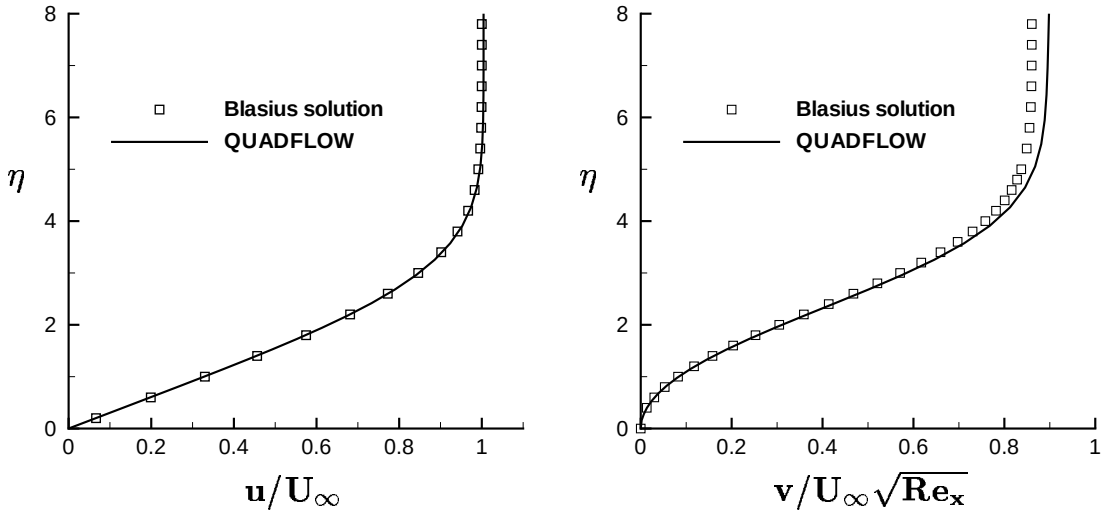


Figure 26 Structured, non-adapted grid for laminar flow over a flat plate. Plate extension: $x \in [0, 2]$. Grid resolution: 176x56 cells, 96 cells located on the plate.



(a) u-velocity profile **(b)** v-velocity profile

Figure 27 Boundary layer profiles at $Re_x = 9169$ for the laminar flow over an isothermal flat plate ($M_\infty = 0.2$, $Re_\infty = 10^3$, $T_w = T_\infty$)

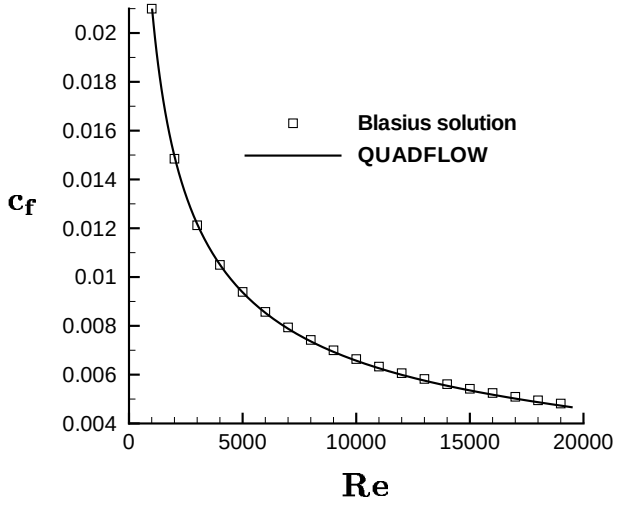


Figure 28 Distribution of the skin friction coefficient for the laminar flow over an isothermal flat plate as function of Reynolds number ($M_\infty = 0.2$, $Re_\infty = 10^3$, $T_w = T_\infty$)

5.2.3 SFB 401 Cruise Configuration

The third test case deals with the inviscid flow about an airfoil according to the transonic SFB 401 cruise configuration at $M_\infty = 0.85$, $\alpha = 0°$. A regular multiblock structured grid consisting of 256x32 cells is employed, with 192 cells located on the airfoil itself. The far-field boundary is placed about 20 chord lengths away from the airfoil. Far-field conditions are determined by using the point vortex correction according to eq. (113) and eq. (114). In Section 6.2, we will revisit this test case, using the fully adaptive algorithm.

Figure 29 presents the computational grid and the isobars for the present flow conditions. The flow exhibits a strong shock at the trailing edge of the upper surface. On the lower surface of the airfoil, two shocks can be identified. The shock located near the leading edge is very smeared, due to insufficient grid resolution, see Figure 29(b).

Figure 30 shows the pressure distribution on the airfoil surface. For the purpose of validation, numerical computations conducted with the FLOWer code [64] serve as a reference. In the latter case, the spatial discretization is based on central differences, stabilized by artificial dissipation of Jameson type. Vortex correction included. Both computational results are in good agreement. As expected, the upwind scheme (QUADFLOW) provides sharper shock resolution than the spatial discretization based on central differences (FLOWer). The aerodynamic lift coefficient is $C_L = 0.463527$ determined by the FLOWer code, and $C_L = 0.497806$ by QUADFLOW, which differ by about 7.4%.

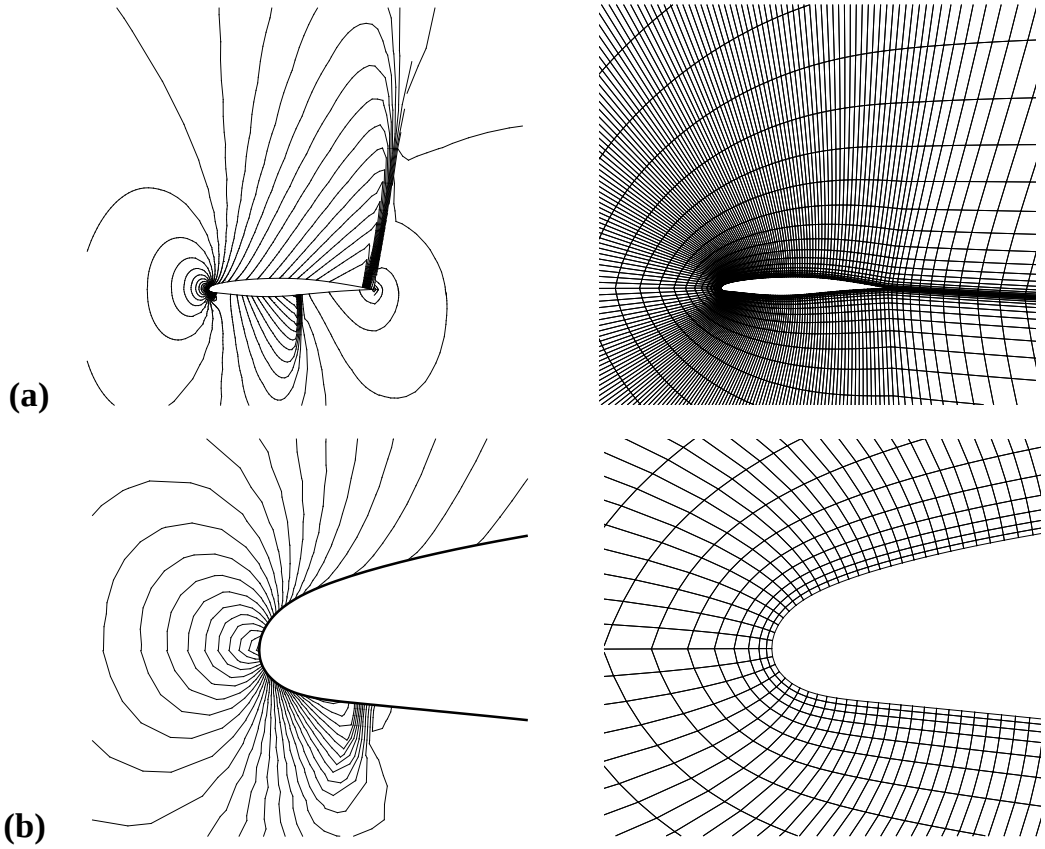


Figure 29 SFB 401 cruise configuration, $M_\infty = 0.85$, $\alpha = 0^\circ$. *Left Figure*: contours of c_p distribution, $c_{p,min} = -1.35$, $c_{p,max} = 1.2$, $\Delta c_p = 0.075$. *Right Figure*: Non-adapted multiblock structured grid, 256x32 cells. **a)** Total view of airfoil. **b)** Partial view of leading edge.

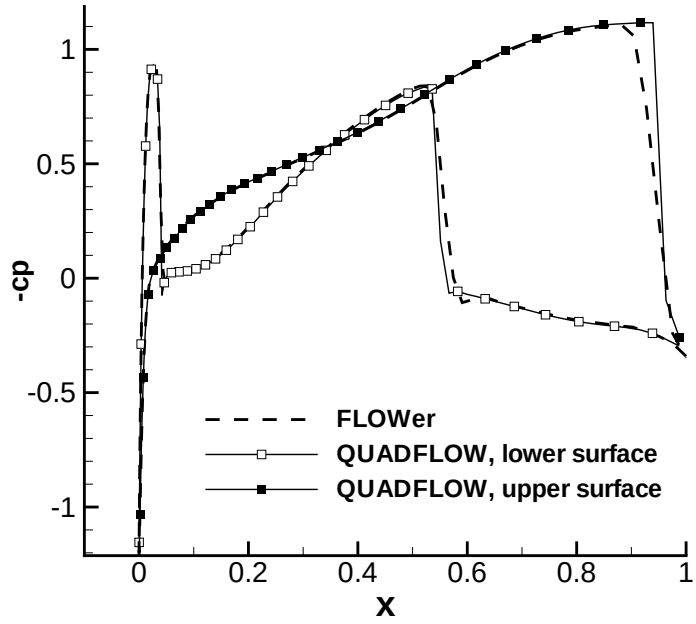


Figure 30 Distribution of the pressure coefficient for the SFB 401 cruise configuration, $M_\infty = 0.85$, $\alpha = 0°$. Grid resolution: 256x32 cells.

5.2.4 Performance of Implicit Time Integration

In order to demonstrate the feasibility of the implicit time integration to accelerate convergence to steady state, we consider the fully subsonic inviscid flow about the SFB 401 cruise configuration at $M_\infty = 0.5$, $\alpha = 0°$. The analysis of the implicit scheme concentrates only on inviscid flow, since viscous terms have not been accounted for within the preconditioner yet. Its extension for viscous flows is subject to current research. The flow conditions have been chosen in order exclude secondary effects associated with discontinuities, which may reduce the rate of convergence. In particular, limiter clipping in vicinity of discontinuities, reflection of entropy waves at far-field boundaries in the presence of shocks, and CFL restrictions due to the non-global convergence property of the Newton scheme may have a significant impact on the rate convergence. In complex applications, these effects are difficult to separate in detail. Therefore, a fully subsonic flow is investigated to solely concentrate on the implicit time integration itself and to exclude the above mentioned effects as far as possible.

Computations have been carried out on a structured, non-adaptive mesh consisting of 256x32 cells like in the previous test case. Cell ordering is arranged in a quasi-lexicographic manner, such that a small bandwidth is achieved. For the solution of the linear system, the residual is converged three orders of magnitude. The Krylov subspace has a dimension of 30 search directions. No restart was required for the considered test cases.

Figure 31 illustrates the convergence history of the non-linear residual for three different CFL numbers, namely $CFL = 10$, 10^2 and 10^3. The residual is monitored in the L_1 norm of the density. $ILU(2)$ serves as preconditioner. The convergence is greatly accelerated with increased CFL number. *Remark*: The scheme remained stable even for $CFL = 10^8$, but did not exhibit further acceleration.

Next, four different levels of fill-in for the $ILU(n)$ preconditioner are considered, namely $n \in [0, 1, 2, 4]$. $CFL = 1000$ is chosen, since it has demonstrated the best rate of convergence, according to Figure 31. Figure 32 illustrates that in this case $ILU(2)$ is the best choice, in terms of computation time.

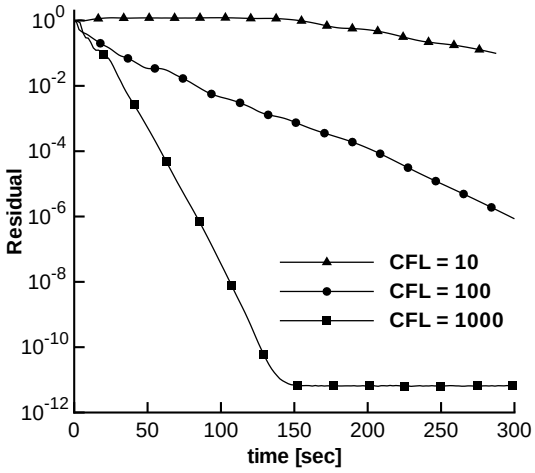


Figure 31 Convergence histories for SFB 401 cruise configuration, $M_\infty = 0.5$, $\alpha = 0°$. $ILU(2)$ preconditioner

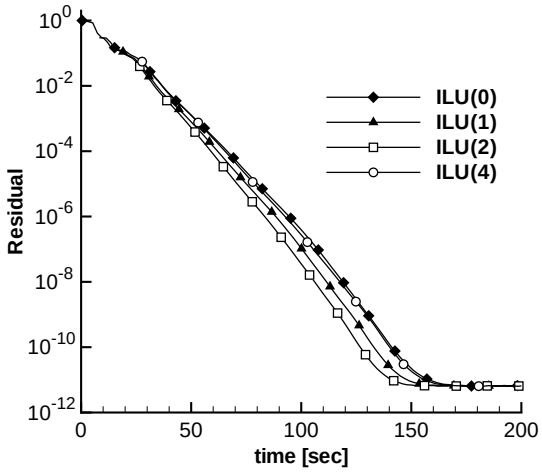


Figure 32 Convergence histories for different levels of $ILU(n)$, $CFL = 1000$. SFB 401 cruise configuration, $M_\infty = 0.5$, $\alpha = 0°$

6 Numerical Results of the Adaptive Scheme

After validation of the basic scheme, we now employ the fully adaptive algorithm. First, the subsonic flow about a NACA0012 airfoil at $M_\infty = 0.63$, $\alpha = 2°$ is investigated. We can demonstrate, that the adaptive algorithm is capable of providing highly accurate flow solutions. Next, we revisit the transonic flow about an airfoil according to the SFB401 cruise configuration, that has been studied in Section 5.2.3 using regular structured grids. Further, the transonic flow over a NACA0012 airfoil at $M_\infty = 0.95$, $\alpha = 0°$ is considered in order to demonstrate the advantage of adaptation to highly resolve complex shock configurations. Finally, the laminar flow over a flat plate is investigated, employing the fully adaptive scheme.

6.1 Subsonic NACA0012 airfoil

The first test case considers the subsonic flow about a NACA0012 airfoil at $M_\infty = 0.63$, $\alpha = 2.0°$. For this configuration, the aerodynamic lift coefficient may be predicted as $C_{L,pot} = 0.333$, by utilizing a full potential method.

Accurate prediction of the aerodynamic coefficients greatly depends on grid resolution and far–field conditions. Both aspects are investigated here. The computational C-type grid extends about 20 chords away from the airfoil. At the far–field, two types of boundary conditions have been applied: The first one utilizes characteristic based boundary conditions. Riemann invariants that enter the computational domain are determined by free–stream conditions, i.e. without application of vortex correction. In the second case, far–field boundary conditions are specified by using the point–vortex correction.

To assess the influence of grid resolution, two different threshold values $\varepsilon = 5 \cdot 10^{-2}$ and $\varepsilon = 3 \cdot 10^{-3}$ are employed for adaptation. The sensitivity of the adaptation primarily influences the grid resolution in the vicinity of the airfoil. In particular, adequate resolution of the stagnation area of the leading edge is of significant importance for the overall prediction of the aerodynamic coefficients.

Computations are initialized on a coarse mesh, consisting of 400 cells, with 10 cells on each side of the airfoil surface itself. A maximum number of $L_{max} = 9$ grid levels are permitted. 15 cycles of adaptation have been performed. The actual grid level that has locally been reached during the computation is $L = 7$ for $\varepsilon = 5 \cdot 10^{-2}$ and $L = 9$ for $\varepsilon = 3 \cdot 10^{-3}$. The highest refinement levels are located within the stagnation area at the leading edge. The residual decreased five orders of magnitude, measured in the L_1-norm of the density-residual. No limiter has been applied.

Figure 33 and Figure 34 show the isolines of the Mach number and the Mach distribution on the airfoil, respectively. Vortex correction has been applied, $\varepsilon = 3 \cdot 10^{-3}$. Table 4 summarizes the results for the considered parameter. The aerodynamic lift coefficient, its derivation from the potential flow solution and the maximum Mach number on the airfoil surface are presented. The variation of the threshold parameter has a large influence on the number of grid cells, which almost increases by a factor of three. Without vortex–correction, the aerodynamic lift is underestimated by about 3%, measured on the finest mesh. Application of the vortex–correction greatly enhances the solution. On the fine grid, the potential flow solution is reached very accurately. On the coarser grid, the aerodynamic lift is underpredicted by about 2%, even with applied vortex–correction. This deficit results from insufficient resolution about the stagnation area at the leading edge. The maximum Mach number only reaches a value of $M_{max} \approx 0.964$, compared with $M_{max} \approx 0.987$ for the fine grid.

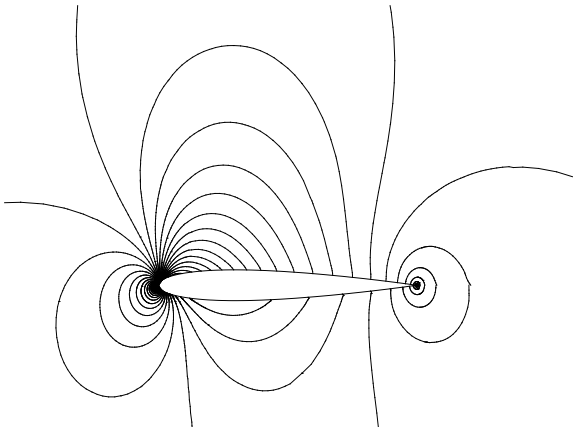


Figure 33 Mach number contours for NACA0012 airfoil, $M_\infty = 0.63$, $\alpha = 2.0°$. $M_{min} = 0.0$, $M_{max} = 0.975$, $\Delta M = 0.025$

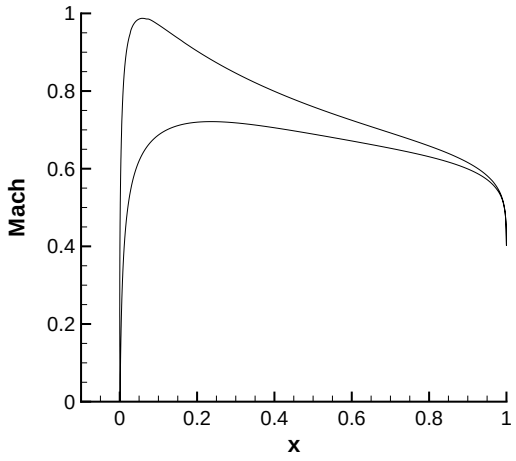


Figure 34 Mach distribution for NACA0012 airfoil, $M_\infty = 0.63$, $\alpha = 2.0°$

Table 4 Aerodynamic coefficients for NACA0012 airfoil, $M = 0.63$, $\alpha = 2.0°$

	$\varepsilon = 5.0 \cdot 10^{-2}$		$\varepsilon = 3.0 \cdot 10^{-3}$	
vortex correction	**no**	**yes**	**no**	**yes**
number of cells	5638	5962	17035	17083
C_L	0.3138246	0.3266950	0.3223484	0.3329005
1-$C_L/C_{L,pot}$ [%]	5.7583783	1.8933933	3.1986786	0.0298798
M_{max}	0.9553167	0.9641491	0.9787976	0.9872640

6.2 Transonic SFB 401 Cruise Configuration

We revisit the inviscid flow around the SFB 401 cruise configuration at $M_\infty = 0.85$, $\alpha = 0°$. In Section 5.2.3, this test case has been investigated using a regular, non-adaptive structured grid. In the following study, the fully adaptive algorithm is utilized. Computations are initialized on a grid consisting of four blocks, each with a resolution of 16x16 cells, see Figure 35. Adaptation is carried out each time the residual decreased to $2.5 \cdot 10^{-4}$, based on the initial residual, measured in the L_1 norm of the density. 14 adaptation cycles have been conducted, with a maximum refinement level of $L_{max} = 9$. Adaptation is applied to the set of primitive variables. The threshold value for the multiscale analysis is $\varepsilon = 8 \cdot 10^{-2}$.

Figure 36 and Figure 37 present computational grids and corresponding pressure distributions for three different stages of adaptation. Namely, for grid no. 1, 4 and 14. Figure 38 shows the corresponding pressure distribution on the airfoil surface for grid no. 1 and grid no. 14. On the coarsest mesh, the shock at the trailing edge

of the upper surface can already be identified. On the lower surface, the shock at $x \approx 0.53$ is very smeared while the shock near the leading edge cannot be identified at all. After 3 cycles of adaptation, all three shocks are present (Figure 36, Figure 37). The mesh contains 7405 cells, with a maximum grid level $L = 4$. The locally finest resolution of the grid approximately corresponds to the resolution of a structured, non-adaptive mesh with 256x32 cells, as used in Section 5.2.3. With further adaptation, the shock resolution can still be improved significantly. On grid no. 14, all shocks are highly resolved. The mesh contains 28288 cells, with a maximum grid level $L = 9$. A large number of cells is concentrated within the shock regions. Although the number of grid cells of the finest mesh appears to be rather high, compared with the structured grid used in Section 5.2.3, it has to be emphasized that the shock resolution is greatly improved by the adaptive scheme. Further, to obtain the same solution quality, the utilization of a structured, non-adaptive grid would prospectively require a significantly higher number of cells than the locally adapted grid. In addition, the process of adjusting a structured grid to the shock location requires a detailed a-priori knowledge of the solution to provide adequate grid resolution.

The aerodynamic lift coefficient is $C_L = 0.514092$, measured on the finest mesh (grid no. 14). In Section 5.2.3 we determined a lift coefficient of $C_L = 0.463527$ (FLOWer) and $C_L = 0.497806$ (QUADFLOW), respectively. This reflects a difference of 10% (FLOWer) and 3% (QUADFLOW), compared with the locally adapted solution.

Figure 39 presents the evolution of the number of grid cells for different stages of adaptation. During the initial phase, the mesh size rapidly increases with each adaptation step. With further adaptation, the number of cells converges against a constant. It is important to note, that large regions of the shocks have not been resolved by the highest refinement level possible ($L_{max} = 9$), but have only reached a lower level of adaptation $L < L_{max}$. I.e., convergence of the mesh size has not been enforced by reaching the highest grid level. Even in the presence of shocks, the adaptation process converges within the range of the prescribed threshold value.

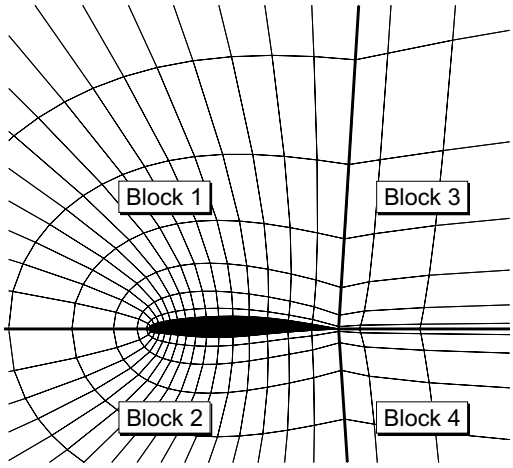
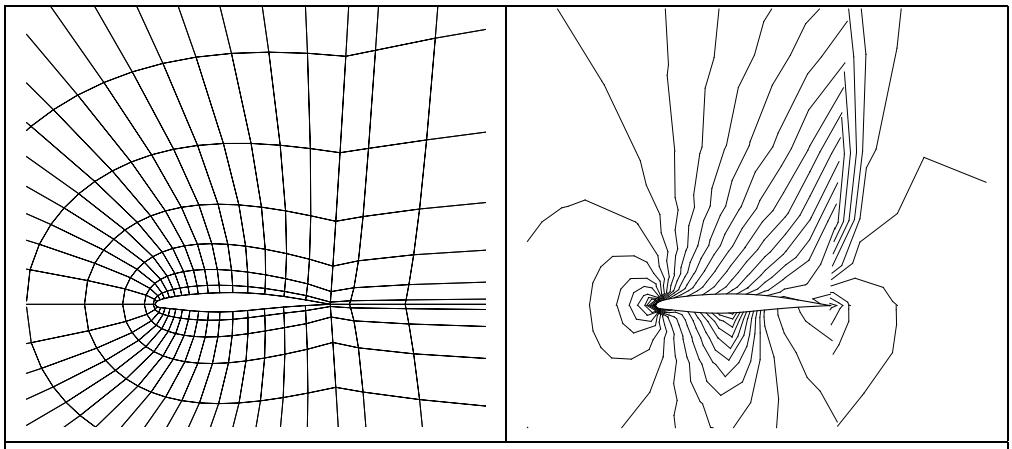
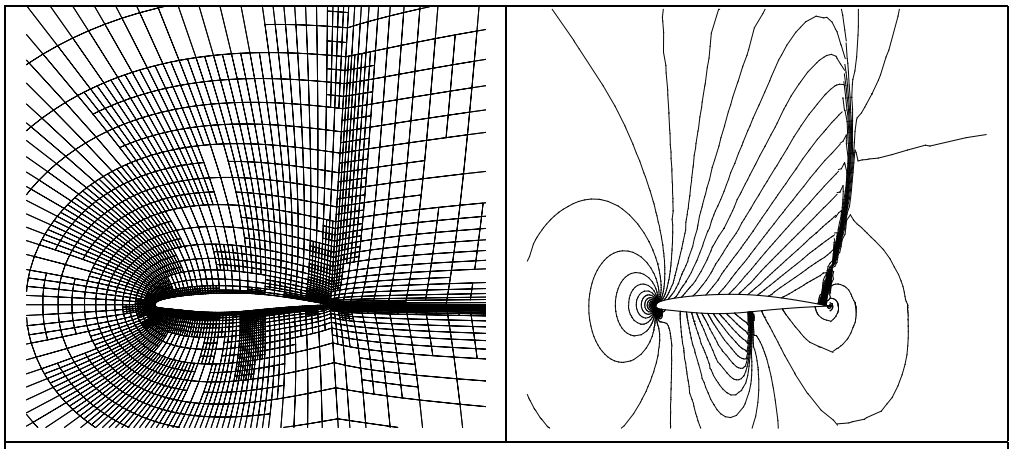


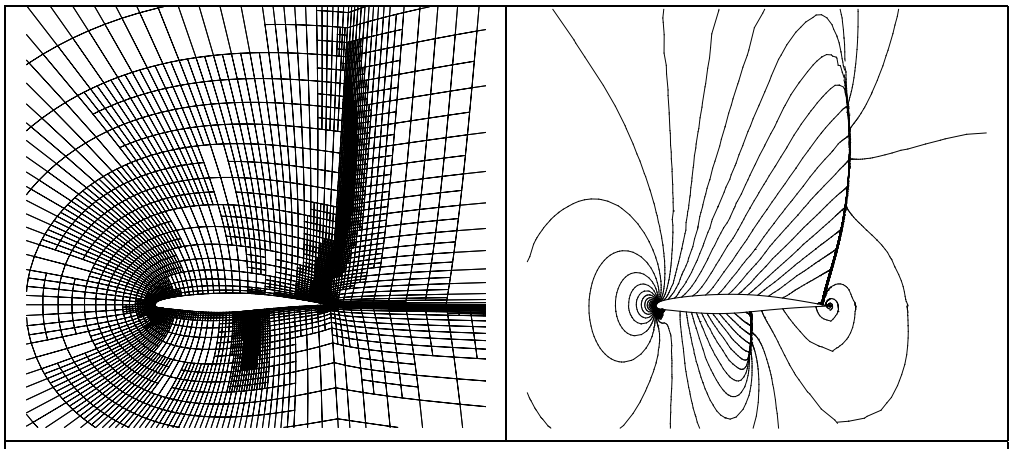
Figure 35 SFB 401 cruise configuration, $M_\infty = 0.85$, $\alpha = 0^\circ$: Initial grid (partial view)



Grid No. 1, number of cells: 1024, $L_{max} = 1$

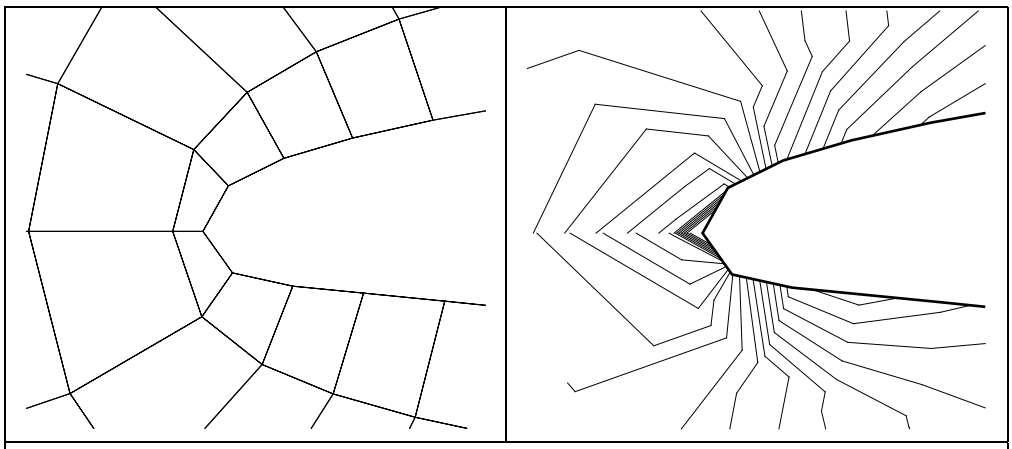


Grid No. 4, number of cells: 7405, $L_{max} = 4$

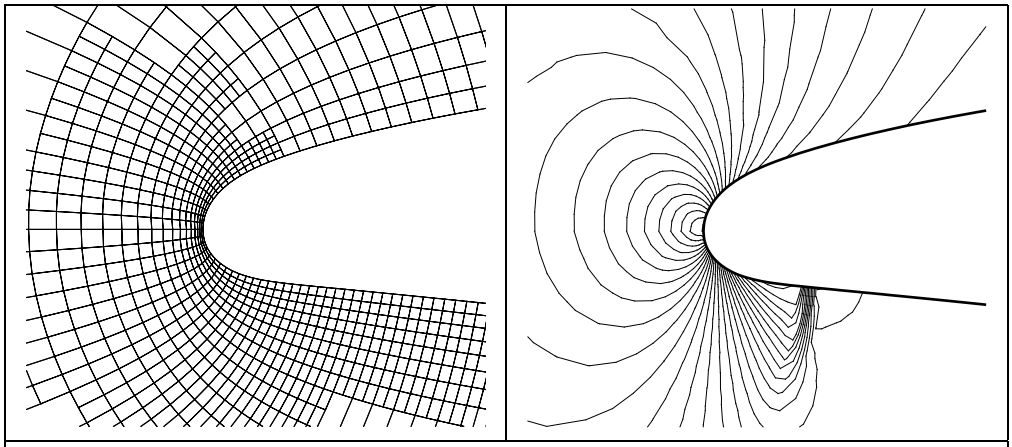


Grid No. 14, number of cells: 28288, $L_{max} = 9$

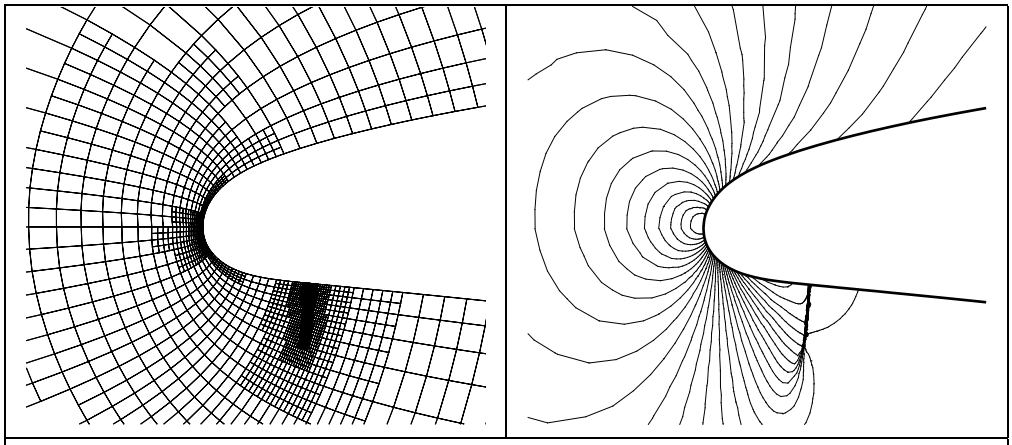
Figure 36 SFB 401 cruise configuration, $M_\infty = 0.85$, $\alpha = 0^\circ$. *Left Figure*: Computational grid. *Right Figure*: c_p distribution, $c_{p,min} = -1.35$, $c_{p,max} = 1.2$, $\Delta c_p = 0.075$



Grid No. 1, number of cells: 1024, $L_{max} = 1$



Grid No. 4, number of cells: 7405, $L_{max} = 4$



Grid No. 14, number of cells: 28288, $L_{max} = 9$

Figure 37 SFB 401 cruise configuration: Detailed view of stagnation area, $M_\infty = 0.85$, $\alpha = 0^\circ$. *Left Figure*: Computational grid. *Right Figure*: c_p distribution, $c_{p,min} = -1.35$, $c_{p,max} = 1.2$, $\Delta c_p = 0.075$

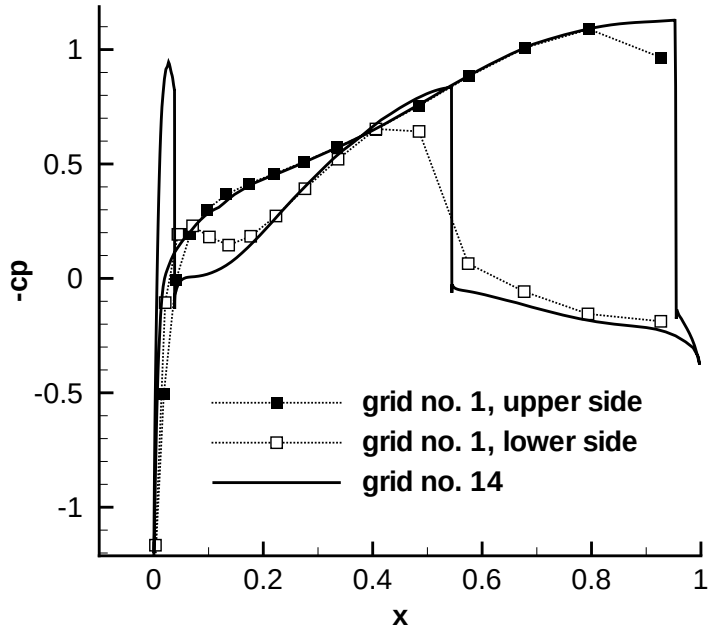


Figure 38 Pressure distribution for SFB 401 cruise configuration, $M_\infty = 0.85$, $\alpha = 0^\circ$

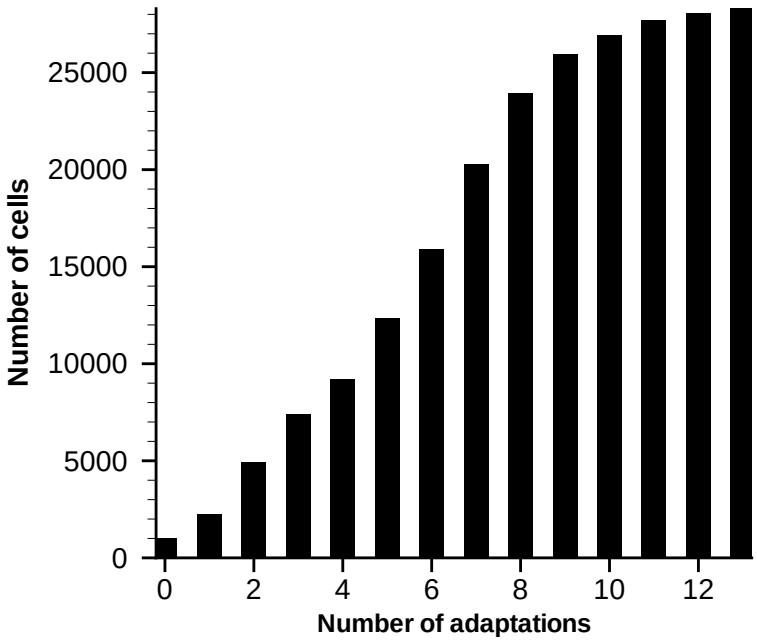


Figure 39 Number of grid cells during adaptation process of SFB 401 cruise configuration, $M_\infty = 0.85$, $\alpha = 0^\circ$

6.3 Transonic NACA0012 airfoil

The next test case concerns the transonic flow over a NACA0012 airfoil at $M_\infty = 0.95$, $\alpha = 0°$ (AGARD reference test case 03 [65]). The far-field boundary is located about 20 chord lengths away from the airfoil. Standard characteristic boundary conditions are applied at the far-field, where the incoming Riemann invariant is set by free stream quantities.

Computations are initialized on a structured grid consisting of 4 blocks with a resolution of 20x20 cells each. 13 cycles of adaptation have been performed with a maximum refinement level of $L_{max} = 8$. Adaptation has been carried out on the set of primitive variables, each time the density residual decreased to 10^{-4}. The threshold value for the multiscale analysis is $\varepsilon = 4 \cdot 10^{-2}$.

The flow pattern downstream of the trailing edge is characterized by a complex shock configuration. Two oblique shocks are formed at the trailing edge. The remaining supersonic region behind the oblique shocks is closed by a further normal shock. This configuration is often related to as so-called *fish-tail*. Figure 40 presents the locally adapted grid and the corresponding Mach distribution in the vicinity of the airfoil, after 13 cycles of adaptation. The grid consists of 55084 cells. All three shocks are highly resolved by the adaptive grid. The position of the normal shock is located at $x \approx 2.1721$ chord length behind the trailing edge of the airfoil.

Figure 41 presents a total view of the shock configuration. The oblique shocks extend about 10 to 12 chord lengths into the flow domain. The adaptive grid provides high resolution over the complete extent of the shocks. Such a high shock resolution is not feasible using standard structured grids. Discretization of the shock region between $x \in [1, 5]$, $y \in [-10, 10]$, by a uniform structured mesh according to a refinement level $L = 8$ equals about $29.5 \cdot 10^6$ grid cells. A uniform discretization of the complete flow domain according to $L = 8$ would result in about 10^8 cells.

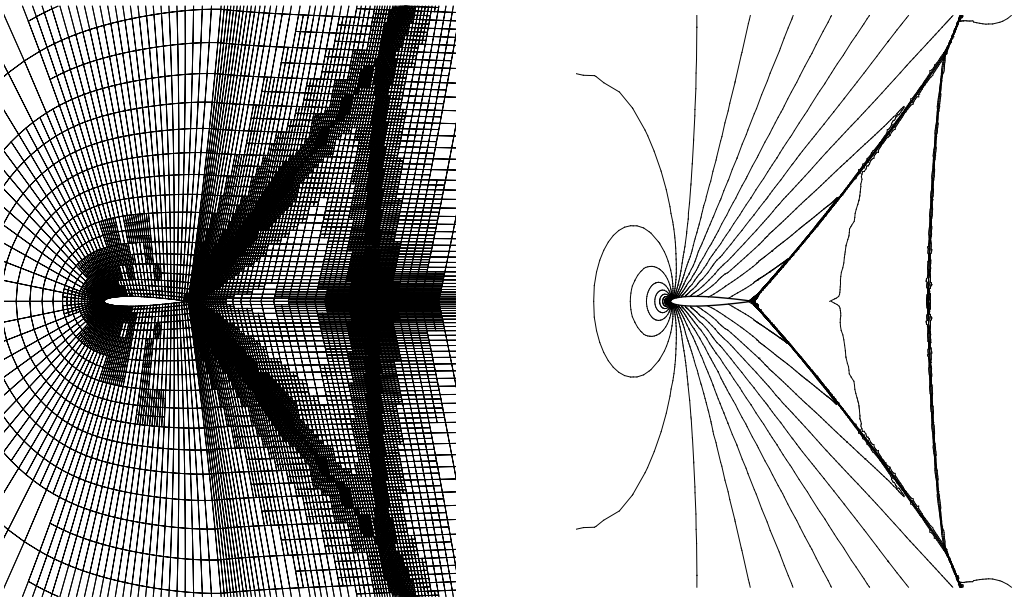


Figure 40 Partial view of NACA0012 airfoil, $M = 0.95$, $\alpha = 0.0°$. *Left Figure*: Computational grid. *Right Figure*: $Mach$ distribution, $M_{min} = 0.0$, $M_{max} = 1.45$, $\Delta M = 0.05$

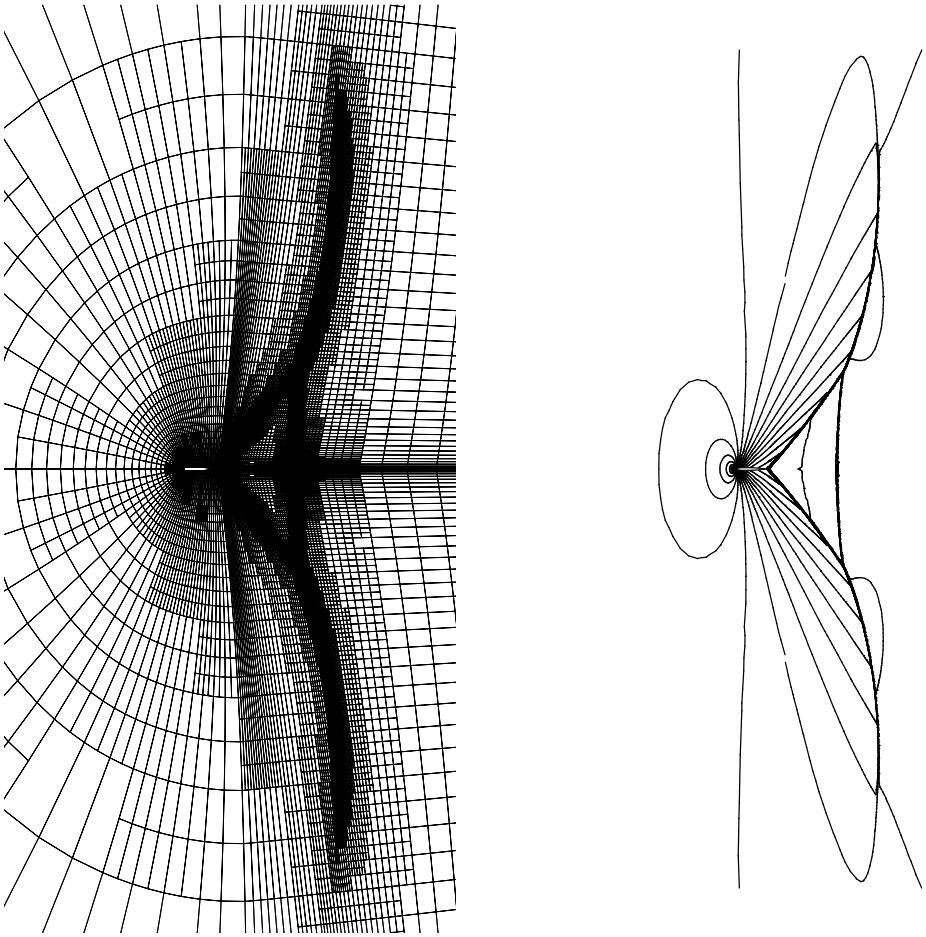


Figure 41 Total view of NACA0012 airfoil, $M = 0.95$, $\alpha = 0.0°$. *Left Figure*: Computational grid. *Right Figure*: $Mach$ distribution, $M_{min} = 0.0$, $M_{max} = 1.45$, $\Delta M = 0.05$

6.4 Laminar Boundary Layer using Adaptation

In this section, we will demonstrate the capability of the multiresolution scheme to automatically resolve laminar boundary layers. The application of the multiscale analysis to viscous flows is of particular interest, since the method has originally been developed in the context of hyperbolic systems of partial differential equations. The following issues are of essential interest for the successful application of the adaptive algorithm to viscous flows:

1. detection and adequate resolution of the boundary layer
2. appropriate choice of the first grid spacing normal to the wall
3. convergence of the multiscale analysis in presence of steep velocity gradients

All three issues will be addressed here.

We reconsider the laminar flow over a flat plate extending on the x-axis in the domain $x \geq 0$. Flow conditions are equivalent to that used in Section 5.2.2, where a regular structured, non-adaptive grid has been employed for validation of the basic scheme. The initial grid consists of 20×8 cells (160 cells all together), with 12 cells located on the plate itself, see Figure 42. For simplicity, the mesh has been generated by coarsening the structured grid employed in Section 5.2.2. The first grid spacing normal to the wall is approximately 10^{-2}, which equals $y^+ \approx 10$ at $Re = 10^4$. At the trailing edge of the plate, about four cells are located within the boundary layer with thickness δ, see Figure 44(a).

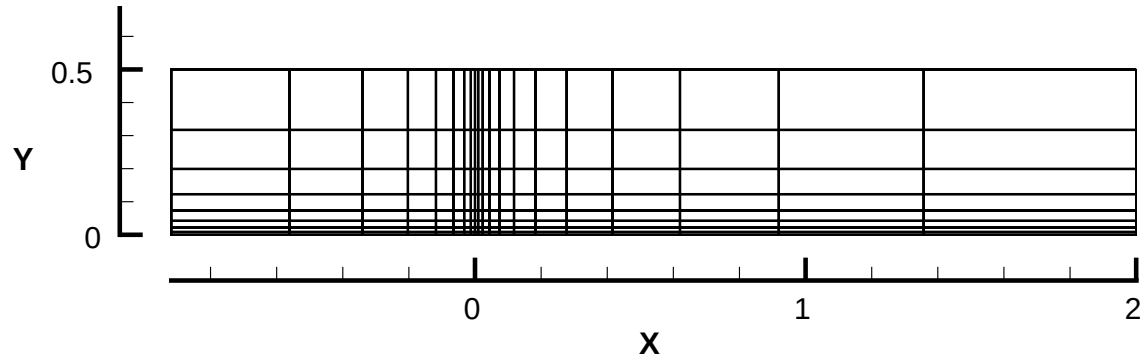


Figure 42 Initial grid for laminar flow over a flat plate. Plate extension: $x \in [0, 2]$. Grid resolution: 20×8 cells with 12 cells located on the plate

10 cycles of adaptation have been performed. Adaptation is carried out on the set of primitive variables, each time the density residual decreased to 10^{-6}, based on its initial value. The highest refinement level permitted is $L_{max} = 8$. A threshold value of $\varepsilon = 8 \cdot 10^{-3}$ is chosen for the multiscale analysis. Figure 43 shows the locally adapted grid after 10 cycles of adaptation. The grid consists of 3595 cells, which is about 36% of the number of cells being used for the structured grid employed in Section 5.2.2. The boundary layer is automatically detected by the present adapta-

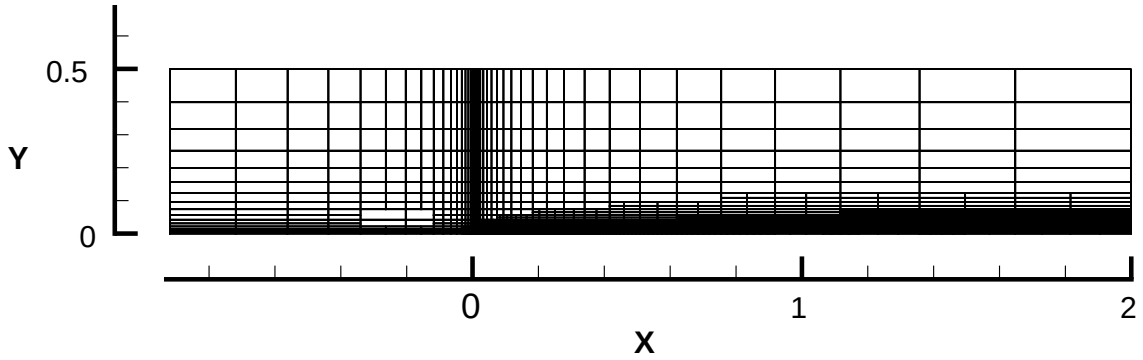
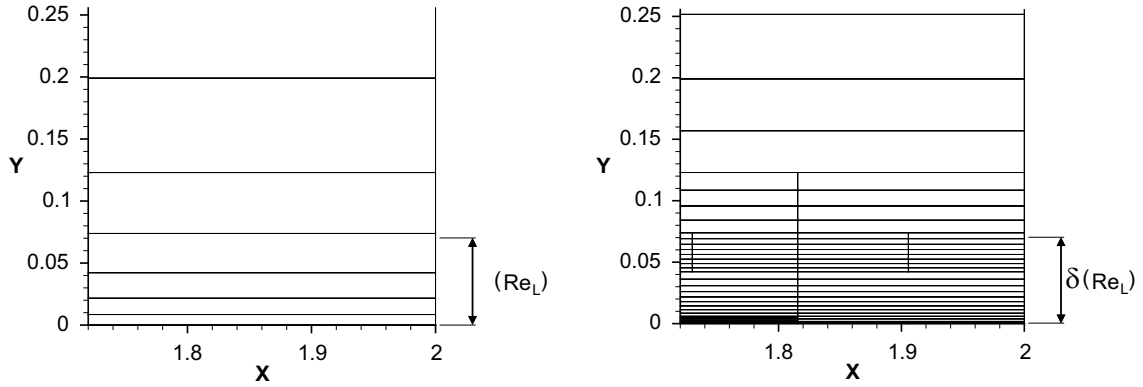


Figure 43 Locally adapted grid for laminar flow over a flat plate. Plate extension: $x \in [0, 2]$. Grid resolution: 3595 cells



(a) initial grid **(b)** adapted grid

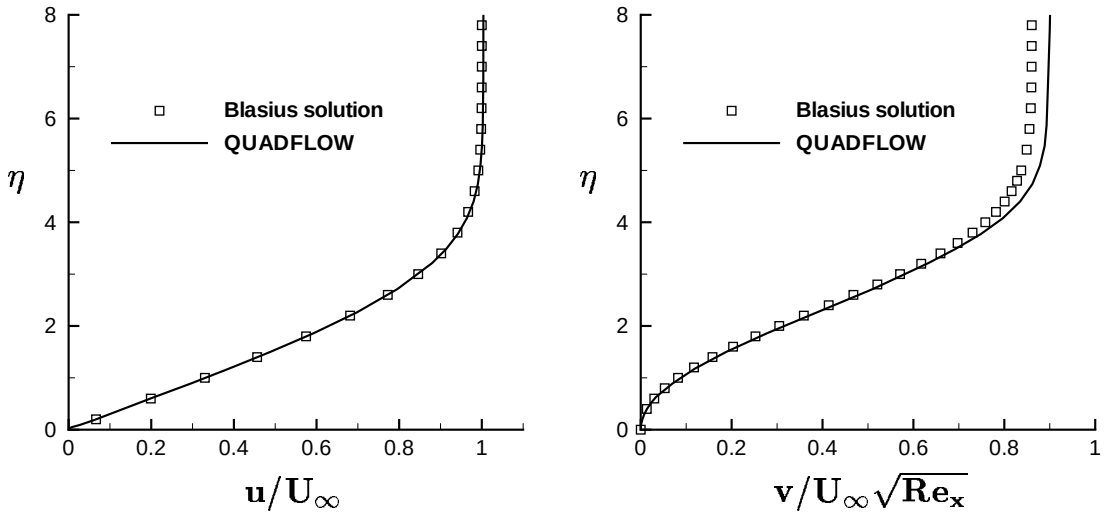
Figure 44 Partial view of trailing edge of flat plate. δ = boundary layer thickness according to Blasius solution

tion criteria. The detailed view of the trailing edge (Figure 44(b)) shows that after reaching the border of the boundary layer ($y > \delta$), the mesh resolution is immediately decreased. The boundary layer is resolved by about 25 cells, at the trailing edge. The first spacing normal to the wall is about 10^{-3}, which equals $y^+ \approx 1$. It is important to note, that the resolution of the first grid spacing corresponds to a refinement level of $L = 4$. I.e., the choice of the first grid spacing has not been enforced by reaching the highest refinement level possible ($L_{max} = 8$). The highest grid level has only been reached in vicinity of the stagnation point at the leading edge of the plate. This fact indicates that even in the presence of steep gradients within the boundary layer, the multiscale analysis converges within the range of the prescribed threshold parameter. The number of grid cells remains bounded during the adaptation process.

Figure 45(a) and Figure 45(b) present a comparison between the computed u-velocity and v-velocity profiles with the Blasius solution at $Re_x = 9188.75$, respectively. Both velocity components are in good agreement with the theoretical solution. In accordance with the results obtained in Section 5.2.2, the v-velocity is slightly over-

predicted for $\eta > 4$ by the numerical scheme. We conclude that this discrepancy is independent of the grid resolution.

Figure 46 shows evolution of the skin friction coefficient along the plate (*please note*: the graph has been truncated for $Re < 1000$). The computed skin friction agrees well with the Blasius solution.



(a) u-velocity profile **(b)** v-velocity profile

Figure 45 Boundary layer profiles at $Re_x = 9188.75$ for the laminar flow over an isothermal flat plate after 10 cycles of adaptation ($M_\infty = 0.2$, $Re_\infty = 10^3$, $T_w = T_\infty$)

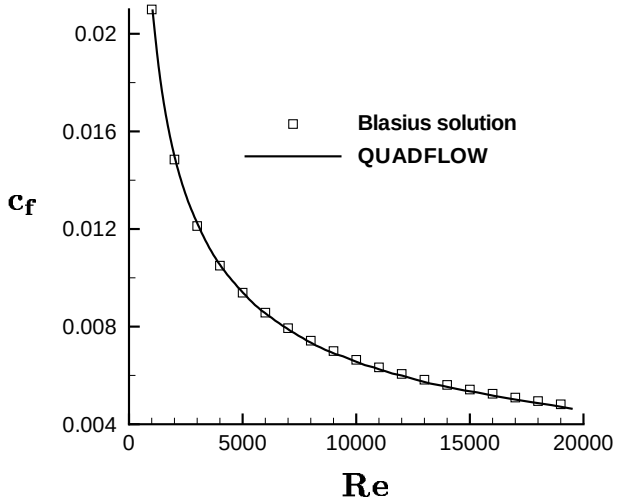


Figure 46 Distribution of the skin friction coefficient for the laminar flow over an isothermal flat plate as function of Reynolds number after 10 cycles of adaptation ($M_\infty = 0.2$, $Re_\infty = 10^3$, $T_w = T_\infty$)

References

[1] R. Abgrall, *Multiresolution analysis on unstructured meshes: Applications to CFD*, in Experimentation, modeling and computation in flow, turbulence and combustion, eds. Chetverushkin and al., John Wiley & Sons, 1997

[2] B. Bihari, A. Harten, *Multiresolution schemes for the numerical solution of 2–D conservation laws I*, SIAM J. Sci. Comput., 18, no. 2 (1997), 315–354

[3] J.M. Carnicer, W. Dahmen, J.M. Peña, *Local decomposition of refinable spaces and wavelets*, Appl. Comput. Harmon. Anal., 3 (1999), 127-153

[4] G. Chiavassa, R. Donat, *Numerical Experiments with Point Value Multiresolution for 2D Compressible Flow*, Technical Report, GrAN-99-4, University of Valencia, 1999

[5] B. Cockburn, F. Coquel, P. LeFloch, *An error estimate for finite volume methods for multidimensional conservation laws*, Math. Comp., 63 (1994), 77-103

[6] B. Cockburn, F. Coquel, P. LeFloch, *An error estimate for higher-order accurate finite volume methods for multidimensional conservation laws*, to appear in Math. Comp.

[7] A. Cohen, N. Dyn, S.M. Kaber, M. Postel, *Multiresolution finite volume schemes on triangles*, J. Comp. Phys., 161 (2000), 264-286

[8] A. Cohen, I. Daubechies, J. Feauveau, *Bi–orthogonal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 45 (1992), 485–560

[9] A. Cohen, S.M. Kaber, S. Müller, M. Postel, *Fully adaptive multiresolution finite volume schemes for conservation laws*, Technical Report 00009, 2000, Laboratoire d'Analyse Numérique, Paris VI, to appear in Math. Comp.

[10] C.M. Dafermos, *Hyperbolic Conservation Laws in Continuum Physics*, Springer Verlag, Berlin, Grundlehren der Mathematischen Wissenschaften, 2000

[11] W. Dahmen, *Some remarks on multiscale transformations, stability and biorthogonality*, in Wavelets, Images and Surface Fitting, eds. P.J. Laurent, A. Le Méhauté, L.L. Schumaker, A.K. Peters, Wellesley, 157–188, 1994

[12] W. Dahmen, B. Gottschlich–Müller, S. Müller, *Multiresolution schemes for conservation laws*, Num. Math., 88, No. 3 (2001), 399-443

[13] W. Dahmen, A. Kunoth, K. Urban, *Biorthogonal spline-wavelets on the interval – stability and moment conditions*, Appl. Comput. Harmon. Anal., 6 (1999), 132-196

[14] B. Gottschlich–Müller, *Multiscale Schemes for Conservation Laws*, PhD thesis, RWTH Aachen, Shaker–Verlag, 1998

[15] Gottschlich–Müller, B., Müller, S., *Application of Multiscale Techniques to Hyperbolic Conservation Laws*, in: Computational Mathematics, Lecture Notes in Pure & Applied Mathematics, **202**, eds. Z. Chen, Y. Li, Ch. A. Micchelli, Y. Xu, Harry–Dekker–Verlag, New York, 1998, 113–138

[16] Gottschlich–Müller, B., Müller, S., *Adaptive Finite Volume Schemes for Conservation Laws based on Local Multiresolution Techniques*, in: Hyperbolic Problems: Theory, Numerics, Applications, eds. M. Fey and R. Jeltsch, Birkhäuser, 1999, 385–394

[17] Harten, A., *Adaptive multiresolution schemes for shock computations*, J. Comp. Phys., 115 (1994), 319–338

[18] Harten, A., *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure Appl. Math. 12, vol. 48 (1995), 1305–1342

[19] D. Kröner, M. Ohlberger, *A posteriori error estimates for upwind finite volume schemes for nonlinear conservation laws in multi dimensions*, Math. Comp., 69 (1999), 25-39

[20] S.N. Kruzhkov, *First order quasilinear equations with several space variables*, Math. USSR Sb., 10 (1970), 217–243 (In Russian)

[21] N.N. Kuznetsov, *The weak solution of the Cauchy problem for a multi–dimensional quasilinear equation*, Mat. Zametki, 2 (1967), 401–410 (In Russian)

[22] Müller, S., *Erweiterung von ENO–Verfahren auf zwei Raumdimensionen und Anwendung auf hypersonische Stauspunktprobleme*, PhD thesis, RWTH Aachen, Shaker–Verlag, 1993

[23] Müller, S., *Adaptive Multiscale Schemes for Conservation Laws*, Habilitation thesis, RWTH Aachen, accepted

[24] S. Müller, A. Voss, *A Manual for the Template Class Library* `igpm_t_lib`, IGPM–Report 197, RWTH Aachen, 2000

[25] F. Schröder–Pander, T. Sonar, O. Friedrich, *Generalized Multiresolution analysis on unstructured grids*, Num. Math., DOI 10.1007/s002110000196, 2000

[26] *Grid Generation, Finite Elements and Geometric Design*, eds. B. Hamann, R.F. Sarraga, special issue, CAGD 12 (1995)

[27] K.H. Brakhage, *High Quality Mesh Generation and Sparse Representation Using B-Splines*, in Proceedings of the 7th International Conference on Numerical Grid Generation in Computational Field Simulations, eds. B.K. Soni, J. Häuser, J.F. Thompson, P. Eiseman, 753-762, Chateau Whistler Resort, British Columbia, 2000

[28] K.H. Brakhage, S. Müller, *Algebraic-Hyperbolic Grid Generation with Precise Control of Intersection of Angles*, Int. J. Numer. Math. Fluids 33(2000), 89-123

[29] K.H. Brakhage, Ein menügesteuertes intelligentes System zur zwei- und dreidimensionalen Computergeometrie, VDI Verlag, series 20: Rechnergestütze Verfahren, no. 26, 1990

[30] K.H. Brakhage, CAG — Computer Aided Geometry, user manual (in German), Institut für Geometrie und Praktische Mathematik, 1986-1999

[31] C. de Boor, *A practical Guide to Splines*, Springer-Verlag, New York, 1978

[32] S.A. Coons, *Surface Patches and B-Spline Curves*, in Computer Aided Geometric Design, eds. R.E. Barnhill, R.F. Riesenfeld, Academic Press, 1974

[33] M. Eck, J. Hadenfeld, *Local Energy Fairing of B-Spline Curves*, Computing Supplementum 10, Geometric Modelling, 129-147, Springer, 1995

[34] M. Eck, J. Hadenfeld, *Knot Removal for B-Spline Curves*, CAGD 12(1995), 259-282

[35] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, 2nd edition, Academic Press, 1990

[36] G. Farin, *Commutativity of Coons and Tensor Product Operations*, Rocky Mountain Journal of Mathematics 22(1992), 541-546

[37] G. Farin, G. Rein, N.S. Sapidis, A.J. Worsey, *Fairing cubic B-Spline curves*, CAGD 4(1987), 91-103

[38] W.J. Gordon, C.A. Hall, *Construction of Curvilinear Coordinate Systems and Applications to Grid Generation*, International Journal for Numerical Methods in Engineering, 7(1973), 461-477

[39] C.A. Hall, W.W. Meyer, *Optimal Error Bounds for Cubic Spline Interpolation*, Journal of Approximation Theory 16(1976), 105-122

[40] J. Hoschek, D. Lasser, *Grundlagen der geometrischen Datenverarbeitung*, 2nd edition, B.G. Teubner, Stuttgart, 1992

[41] P. Knupp, S. Steinberg, *Fundamentals of Grid Generation*, CRC Press (1994)

[42] J.A. Sethian, *Curvature Flow and Entropy Condition Applied to Grid Generation*, J. Comp. Phys. 115(1994), 440-454

[43] S.P. Spekreijse, *Elliptic Grid Generation Based on Laplace Equations and Algebraic Transformations*, J. Comp. Phys. 118(1995), 38-61

[44] S.P. Spekreijse, J.W. Boerstoel, *Multiblock Grid Generation. Part 1: Elliptic Grid Generation Methods for Structured Grids*, in Computational Fluid Dynamics, VKI Lecture Series 1996-06, ed. H. Deconinck, von Karman Institut for Fluid Dynamics, 1-48, 1996

[45] S.P. Spekreijse, J.W. Boerstoel. *Multiblock Grid Generation. Part 2: Multiblock Aspects*, in Computational Fluid Dynamics, VKI Lecture Series 1996-06, ed. H. Deconinck, von Karman Institut for Fluid Dynamics, 1-39, 1996

[46] P. Batten, M.A. Leschziner, U.C. Goldberg, *Average-State Jacobians and Implicit Methods for Compressible Viscous and Turbulent Flows*, J. Comp. Phys., 137, 1997, 38–78

[47] P.L. Roe, *Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes*, J. Comp. Phys, 43, 1981, 357–372

[48] V. Venkatakrishnan, *Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters*, J. Comp. Phys., 118, 1995, 120–130

[49] W.J. Coirer, *An Adaptively Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, Ph.D. Thesis, University of Michigan, 1994

[50] M. Delanaye, M.J. Aftosmis, M.J. Berger, Y. Liu, T.H. Pulliam, *Automatic Hybrid–Cartesian Grid Generatin for High–Reynolds Number Flows around Complex Geometries*, AIAA Paper 99-0777, 1999

[51] D.G. Holmes, S.D. Connell, *Solution of the 2D Navier-Stokes Equations on Unstructured Adaptive Grids*, AIAA Paper 89-1932, 1989

[52] W.K. Anderson, D.L. Bonhaus, *An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids*, Computers & Fluids, vol 23, no 1, 1994, 1-21

[53] P. Geuzaine, *An Implicit Upwind Finite Volume Method for Compressible Turbulent Flows on Unstructured Meshes*, PhD Thesis, Universitè de Liège, 1999

[54] C.H. Hirsch, *Numerical Computation of Internal and External Flows: Computational Methods for Inviscid and Viscous Flows*, volume 2, Wiley, 1988

[55] J.L. Thomas, M.D. Salas, *Far–Field Boundary Conditions for Transonic Lifting Solutions to the Euler Equations*, AIAA J., 24, 1986, 1074–1080

[56] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996

[57] I. Grotowsky, J. Ballmann, *Efficient Time Integration of Navier-Stokes Equations*, Computers & Fluids, vol 28, no 2, 1999, 243–263

[58] K.J. Vanden, P.D. Orkwis, *Comparison of Numerical and Analytical Jacobians*, AIAA Journal, vol 34, no 6, 1996

[59] S. Balay, W. Gropp, L.C. McInnes, B.F. Smith, *PETSc 2.0 Users Manual*, Tech. Rep. ANL-95/11 - Revision 2.0.28, Argonne National Laboratory, 2000, `http://www-fp.mcs.anl.gov/petsc/`

[60] F. Bramkamp, J. Ballmann, S. Müller, *Development of a Flow Solver Employing Local Adaptation Based on Multiscale Analysis on B-Spline Grids*, in: Proceedings of 8th Annual Conf. of the CFD Society of Canada, 2000, 113-118

[61] F. Bramkamp, J. Ballmann, *Solution of the Euler Equations on Locally Adaptive B-Spline Grids*, NNFM, 2000, to appear

[62] F. Ringleb, *Exakte Lösung der Differentialgleichung einer adiabatischen Gasströmung*, ZAMM, 20(4), 1940, 185–198

[63] L.G. Loitsianski, *Laminare Grenzschichten*, Akademie Verlag, Berlin, 1967

[64] K. Becker, N. Kroll, C.C. Rossow, F. Thiele, *Numerical Flow Calculations for Complete Aircraft - the Megaflow Project*, DGLR-JT98-225, DGLR Jahrbuch 1998, 355–364

[65] *Test Cases for Inviscid Flow Field Methods*, AGARD Advisory Report No. 211, 1985