Inaugural – Dissertation
zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht–Karls–Universität
Heidelberg

vorgelegt von

Diplom–Mathematiker Ralf Hartmann

aus Heidelberg

Tag der mündlichen Prüfung: 15. Juli 2002

Adaptive Finite Element Methods for the Compressible Euler Equations

Gutachter: Prof. Rolf Rannacher, Heidelberg
Prof. Endre Süli, Oxford

Contents

Chapter 1

Introduction

In many applications there is particular interest in specific quantities of the solution. These include physically relevant quantities like, for example in aerodynamics, the drag or lift of an airfoil, the pressure difference between the leading and trailing edge of the airfoil or single density or pressure values on the profile of the airfoil. While traditionally these quantities are measured in wind tunnel experiments, nowadays these experiments are increasingly replaced by numerical simulations aiming to predict these quantities, see the recent "First AIAA Drag Prediction Workshop" [42, 44] for example.

Multi-dimensional compressible flows are modelled by nonlinear conservation laws whose solutions exhibit a wide range of localised structures, such as shock waves, contact discontinuities and rarefaction waves. The accurate numerical resolution of these features necessitates the use of locally refined, adaptive computational meshes. The majority of adaptive CFD algorithms will refine or adjust the computational mesh according to an *ad hoc* criterion, such as a large gradient or curvature information in one of the field variables. Although this intuitive approach has had some success, it generally neither yields meshes that are efficient for computing the quantities of interest nor gives efficient and reliable error estimates of the computed quantities. Indeed, the design and implementation of efficient adaptive finite element/finite volume methods and the computation of reliable values for physically relevant quantities is one of the main challenges in the field of computational partial differential equations.

In this thesis we introduce a discontinuous Galerkin method for the numerical solution of hyperbolic conversation laws, as for example the compressible Euler equations of gas dynamics. Based on this finite element method, we develop an adaptive algorithm for the efficient computation of physically relevant quantities of the solution. This includes *a posteriori* error estimation of the error in the computed quantity as well as adaptive mesh design specifically tailored to the efficient computation of this quantity. We illustrate this approach by several different hyperbolic problems in combination with various different target quantities, including the efficient computation of drag and lift coefficients of airfoils immersed in inviscid compressible gas flows.

Discretisation For the discretisation of the hyperbolic problems we employ the *discontinuous Galerkin finite element method*, see the survey article Cockburn *et al.* [15] and the references cited therein. The discontinuous Galerkin method, DG method for short, is locally conservative and may be regarded as a higher–order generalisation of the *finite volume method*. While the lowest order discontinuous Galerkin method, using piecewise constant trial and test functions, coincides with the basic finite volume method, the order of the DG method can easily be increased by replacing the trial and test functions of the finite element method by polynomials of higher degree p, yielding so–called DG(p) methods. This way, higher order discretisations are obtained without the use of any recovery or similar techniques commonly employed in finite volume methods.

We enhance the discontinuous Galerkin discretisation by *consistent* artificial viscosity terms, similar to those in [35], in order to avoid spurious oscillations of the numerical solution in the vicinity of sharp features like shocks and contact discontinuities. Furthermore, we employ higher order boundary approximations at curved reflective boundaries to avoid unphysical solutions, cf. [6]. Finally, we solve the resulting nonlinear problems by Newton iteration methods.

***A posteriori* error estimation and adaptivity** The main purpose of this work is to consider the *a posteriori* error estimation in terms of physically relevant quantities $J(u)$ of the solution, and adaptive mesh design for the efficient computation of these quantities, cf. [5], the survey articles [20, 7] and the references cited therein. In particular, by employing a duality argument we derive the error representation formula

$$J(u) - J(u_h) \equiv \sum_{K \in T_h} \eta_K \,, \tag{1.1}$$

where η_K on each element K of the triangulation T_h consists of the finite element residuals depending only on the numerical solution u_h multiplied by local weighting terms involving the solution z of a certain dual problem. These weights provide valuable information concerning the global transport of the error and the interaction of the error in different components of the solution.

Since the solution z to the dual problem is usually unknown analytically it may be desirable to eliminate it from the error representation (1.1); indeed, by employing the Cauchy–Schwarz inequality, together with standard results from approximation theory, the local terms η_K, $K \in T_h$, may be bounded from above in terms of powers of the mesh function h and Sobolev seminorms of z. Then, z may be completely eliminated from the *a posteriori* estimate by bounding norms of z by suitable norms of the data for the dual problem by employing well–posedness results. The resulting Type II (cf. [34]) *a posteriori* error bound then only involves the computable finite element residual, the discretisation parameter h, interpolation constants and the stability factor for the dual problem. This type of bounds are in the spirit of the ones derived by C. Johnson and his co-workers. However, as we shall see the elimination of the weighting terms

involving the dual solution z may adversely affect the efficiency of the resulting adaptive algorithm leading to uneconomical mesh design. Indeed, we shall see that already by the first step *en route* to deriving a Type II *a posteriori* estimate, the application of the triangle inequality on the right–hand side of (1.1), may lead to large overestimation of the true error. This is because all cancellation effects that occur in the error representation (1.1) through the summation of the local terms η_K are suppressed. Therefore, we refrain from bounding the terms on the right–hand side of (1.1); in particular, we shall not employ the triangle inequality and thereby not eliminate the dual solution z from our *a posteriori* error bound by exploiting standard approximation results and the strong stability of the dual problem. Instead, the dual problem will be approximated numerically as part of the error estimation process. However, the cost of this additional calculation is in general relatively cheap in comparison to the cost of determining the numerical approximation to the original (primal) system of *nonlinear* partial differential equations, since the dual problem comprises of a system of *linear* partial differential equations.

We shall show in a variety of numerical experiments that the *approximate error representation*

$$\tilde{\mathcal{E}}_\Omega \equiv \sum_{K \in T_h} \tilde{\eta}_K, \tag{1.2}$$

originating from (1.1) by replacing the exact dual solution z by a numerical approximation, is very close to the true error and thus provides a sharp and reliable error estimate of the error in the quantity of interest. Furthermore, we employ the computed local indicators $|\tilde{\eta}_K|$, also referred to as *weighted indicators*, for adaptive mesh refinement resulting in meshes that are specifically tailored to the cost-efficient computation of the quantity of interest. We compare the efficiency of these meshes with meshes produced by so–called *ad hoc* indicators that simply rely on the residual or smoothness information of the solution.

In the following, we show a typical example of adaptive mesh design by considering a supersonic flow past an airfoil, see Section 4.5.3 for more details. Figure 1.2(a) shows the mesh produced using an *ad hoc* indicator η_K^{adhoc} that depends on the finite element residuals, only. This mesh is largely concentrated in the vicinity of both shocks of the flow; the bow shock in front of the airfoil and the trailing shock. We note that other *ad hoc* indicators relying, for example, on large gradients or curvature information of the numerical solution, produce meshes fairly similar in character.

Let us assume that there is particular interest in the point value of the pressure at the leading edge of the airfoil, see Figure 1.1.

Figure 1.1: Pressure value at leading edge of airfoil.

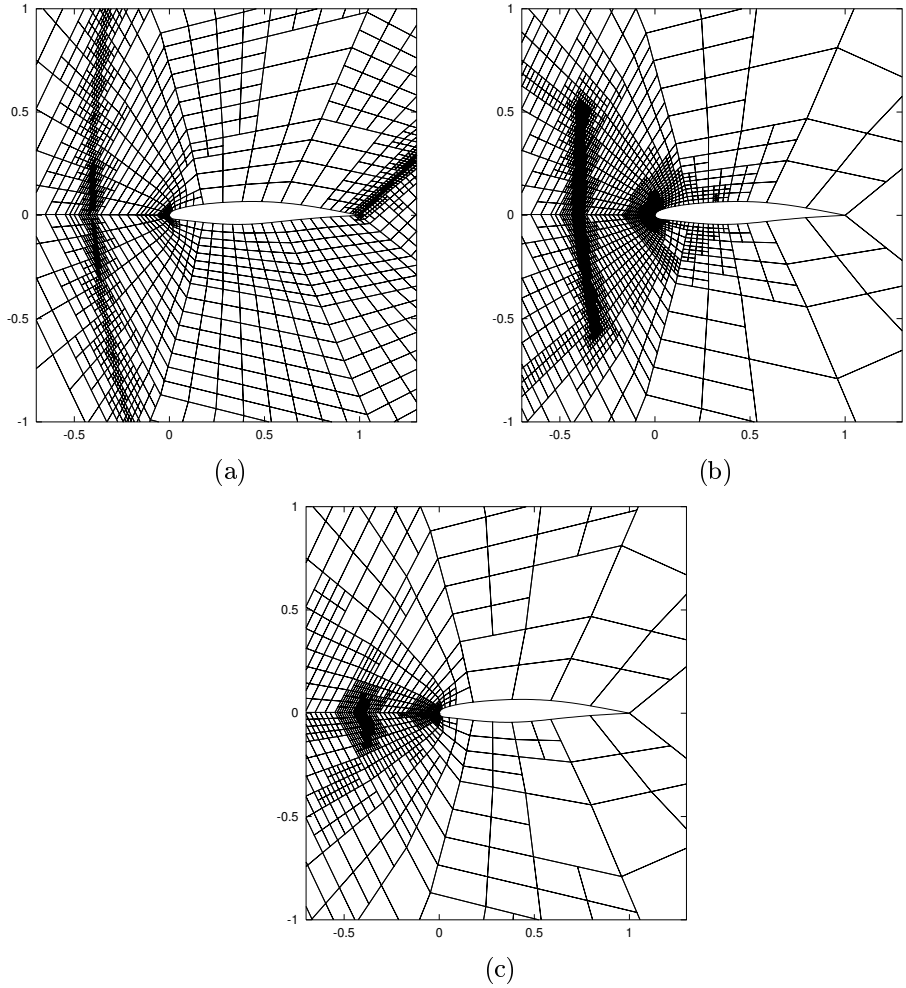


Figure 1.2: Supersonic flow around airfoil. Mesh constructed using (a) an *ad hoc* indicator with 13719 elements ($|J(u) - J(u_h)| = 3.542 \times 10^{-2}$), (b) a modified *ad hoc* indicator with 9516 elements ($|J(u) - J(u_h)| = 7.924 \times 10^{-3}$), and (c) the weighted indicator with 1803 elements ($|J(u) - J(u_h)| = 3.042 \times 10^{-3}$).

Then, it is obvious that the mesh in Figure 1.2(a) is *not* efficient for accurately computing this quantity of interest. Indeed, due to physical reasons, the supersonic flow downstream of the airfoil does not affect the value at the leading edge. Also, regions that are placed sufficiently far above or below the airfoil are believed to not significantly affect the accuracy of the quantity of interest. Therefore, we consider an alternative *ad hoc* error indicator based on

a modification of η_K^{adhoc},

$$\eta_K^{\text{adhoc,C}} = \begin{cases} \eta_K^{\text{adhoc}}, & \text{if centroid}(K) \in C, \\ 0, & \text{otherwise,} \end{cases}$$

whereby only elements in a neighbourhood of a cone-shaped region C upstream of the point of interest are marked for refinement, inhibiting the refinement of the region downstream and the regions sufficiently far above and below the airfoil, see mesh in Figure 1.2(b). However, it is *a priori* not clear which parts of the bow shock in front of airfoil will influence the target quantity. The angle β of the cone C should not be chosen too small as otherwise the lack of resolution of the bow shock will impact on the computed value of the pressure at the leading edge of the airfoil; on the other hand choosing β too large may lead to over–refinement.

In contrast, the weighted error indicator $|\tilde{\eta}_K|$ provides all the necessary information in order to decide which regions of the shock should be refined, and by what extent. Indeed, though the mesh produced using the modified *ad hoc* indicator $\eta_K^{\text{adhoc,C}}$ is significantly more efficient than the mesh in Figure 1.2(a) produced using the (unmodified) *ad hoc* indicator, the mesh produced using the weighted indicator $|\tilde{\eta}_K|$ is even more efficient, for more details about this example, see Section 4.5.3.

Pre-Publications

We note that in agreement with the advisor some of the numerical results presented in this work have already been published or submitted, see [24, 26, 27, 30, 31]. Some of the most recent results are published in joint work with Paul Houston, Leicester. However, it is emphasised that all test problems presented here originate from the author's own work. All numerical data and results included in this work were produced using a code implemented by the author based on the deal.II library [3].

Acknowledgments

I wish to thank Prof. Rolf Rannacher for suggesting this interesting subject and for supporting this work, Wolfgang Bangerth and Dr. Guido Kanschat for their helpful comments and Dr. Paul Houston for a very fruitful cooperation.

I acknowledge the financial support of the DFG Priority Research Program "Analysis and Numerics of Conservation Laws" and the SFB 359 "Reactive Flows, Diffusion and Transport" at the IWR, University of Heidelberg.

Chapter 2

Finite Element approximation to hyperbolic equations

In this chapter we describe the discretisation of the finite element method we employ for solving nonlinear hyperbolic equations. As already indicated in the introduction we particularly consider a *discontinuous Galerkin* approximation to stationary hyperbolic problems with special emphasis on the 2D compressible Euler equations. This chapter deals with the discretisation, the solution of the numerical problems and the convergence of numerical solutions.

In Sections 2.1 and 2.2 we introduce a model problem of nonlinear hyperbolic conservation laws and we formulate its discontinuous Galerkin finite element approximation. After quoting some *a priori* error estimation results for the discontinuous Galerkin discretisation of scalar hyperbolic problems in Section 2.3, we present a numerical example in Section 2.4 indicating that the DG method applied to the 2D compressible Euler equations gives an order $O(h^{p+1})$ of convergence on smooth solutions when piecewise polynomials of degree p are employed. Then, in Section 2.5, we compare the discontinuous Galerkin (DG) method and the streamline diffusion (SD) method with respect to accuracy and conservation properties of the methods.

In Section 2.6 we consider boundary conditions for the 2D compressible Euler equation and introduce sub- and supersonic inflow and outflow boundary conditions as well as reflective boundary conditions for the DG method. Then, in Section 2.7, we define a consistent shock-capturing term that is added to the DG discretisation in order to avoid spurious oscillations in the vicinity of discontinuities of the solution. From the consistency of the DG method we deduce the so–called Galerkin orthogonality property of the finite element method.

In Section 2.8 we explain which solvers we employ for solving the linear and nonlinear problems arising from DG discretisations of nonlinear hyperbolic equations. Then, in Section 2.9 we introduce mapping functions of higher order

polynomial degree that allow the discretisation of curved boundaries by a higher order boundary approximation. It is shown that this must be employed on a curved reflective boundary in order to avoid unphysical numerical solutions.

Finally in Sections 2.10 and 2.11 we discuss the mesh generation for airfoil geometries and show the numerical solution of a transonic flow around the BAC3-11 airfoil on a sequence of adaptively refined meshes.

2.1 Model problem

Given a final time $T > 0$, we consider the following system of conservation equations,

$$\begin{aligned} \partial_t u + \sum_{i=1}^{d} \partial_{x_i} F_i(u) = 0 \qquad & \text{in } (0,T] \times \Omega, \\ u(0,\cdot) = u_0(\cdot) \qquad & \text{in } \Omega, \end{aligned} \tag{2.1}$$

where Ω is a bounded connected domain in $\mathbb{R}^d$, $d \geq 1$, $u = (u_1, \ldots, u_m)^T$, $F = (F_1(u), \ldots, F_d(u))$ and $F_i : \mathbb{R}^m \to \mathbb{R}^m$, $i = 1, \ldots, d$, are continuously differentiable. In particular, we will be concerned with the solution of the *stationary* system of conservation laws,

$$\nabla \cdot F(u) = 0 \quad \text{in } \Omega, \tag{2.2}$$

subject to appropriate boundary conditions described below. We say that (2.1) is hyperbolic, if the matrix

$$B(u,\nu) := \sum_{i=1}^{d} \nu_i A_i(u) \tag{2.3}$$

has m real eigenvalues and a complete set of linearly independent eigenvectors for all vectors $\nu = (\nu_1, \ldots, \nu_d) \in \mathbb{R}^d$. Here, $A_i(u)$ denotes the Jacobi matrix of the flux $F_i(u)$, i.e.

$$A_i(u) := F_i'(u), \qquad i = 1, \ldots, d.$$

An important example of such a model problem is represented by the Euler equations of compressible gas dynamics. In two space–dimensions, the state vector is $u = (\rho, \rho v_1, \rho v_2, e)^T$, where ρ, $v = (v_1, v_2)^T$ and e represent the density, the velocity vector and the total energy per unit volume, respectively. Furthermore, the fluxes are defined by

$$F_1(u) = \begin{pmatrix} \varrho v_1 \\ \varrho v_1^2 + p \\ \varrho v_1 v_2 \\ v_1(e+p) \end{pmatrix} \quad \text{and} \quad F_2(u) = \begin{pmatrix} \varrho v_2 \\ \varrho v_1 v_2 \\ \varrho v_2^2 + p \\ v_2(e+p) \end{pmatrix}. \tag{2.4}$$

Here, p denotes the pressure subject to the equation of state of an ideal gas

$$p = (\gamma - 1)\left(e - \tfrac{1}{2}\rho v^2\right), \tag{2.5}$$

where γ is the ratio of specific heats which, for dry air, is $\gamma \approx 1.4$.

The system of conservation laws (2.1) must be supplemented by appropriate boundary conditions; for example at inflow/outflow boundaries, we require that

$$B^-(u,n)\,(u-g) = 0,$$

where n denotes the unit outward normal vector to the boundary Γ and g is a (given) vector function. Here, $B^-(u,n)$ denotes the negative part of $B(u,n)$,

$$B^-(u,n) = P\Lambda^- P^{-1}, \tag{2.6}$$

where $P = [r_1, \ldots, r_m]$ denotes the $m \times m$ matrix of eigenvectors of $B(u,n)$ and $\Lambda^- = \mathrm{diag}(\min(\lambda_i, 0))$ the $m \times m$ diagonal matrix of the negative eigenvalues of $B(u,n)$ with $Br_i = \lambda_i r_i$, $i = 1, \ldots, d$.

2.2 Discontinuous Galerkin discretisation

For practical computations we first need to discretise the problem. In particular, here we consider the *discontinuous Galerkin* finite element discretisation.

First, we begin by introducing some notation. Suppose that T_h is a subdivision of Ω into disjunctive open element domains K such that $\bar{\Omega} = \cup_{K \in T_h} \bar{K}$. Here, h denotes the piecewise constant mesh function defined by $h|_K \equiv h_K = \mathrm{diam}(K)$ for all $K \in T_h$. Let us assume that each $K \in T_h$ is an image of a fixed reference element $\hat{K}$, that is, $K = \sigma_K(\hat{K})$ for all $K \in T_h$. Here, we shall only consider the case when $\hat{K}$ is the open unit hypercube in $\mathbb{R}^d$. Furthermore the mapping σ_K of the reference element $\hat{K}$ to the element K in real space is assumed to be bijective and smooth, with the eigenvalues of its Jacobian matrix being bounded from below and above. For elements in the interior of the domain, $\partial K \cap \Gamma = \emptyset$, the mapping σ_K is given by a d-linear function; for elements on the boundary $\partial K \cap \Gamma \neq \emptyset$, it might be necessary to employ mappings that include polynomials of higher degree, see Section 2.9 for more details. On the reference element $\hat{K}$ we define spaces of polynomials of degree $p \geq 0$ as follows:

$$Q_p = \mathrm{span}\left\{\hat{x}^\alpha : 0 \leq \alpha_i \leq p,\ 0 \leq i \leq d\right\},$$

where here α denotes a multi-index and $x^\alpha = \prod_{i=1}^d x_i^{\alpha_i}$. Now, we introduce the finite element space V_h^p consisting of discontinuous vector–valued polynomial functions of degree $p \geq 0$, defined by

$$V_h^p = \{v_h \in [L_2(\Omega)]^m : v_h|_K \circ \sigma_K \in [Q_p]^m\}. \tag{2.7}$$

Suppose that $v \in H^1(K)$ for each $K \in T_h$. For each $K \in T_h$ and each $x \in \partial K$ we denote by v_K^+ the interior trace of v on ∂K and by v_K^- the outer

trace on ∂K. The inner trace v_K^+ of v on ∂K is the trace taken from within K. The outer trace v_K^- on ∂K is taken from within the neighbouring element $K'_K(x)$ for $x \in \partial K \setminus \Gamma$. Here, $K'_K(x)$ is the (with the exception of a set of $d-1$–dimensional measure zero) unique element $K'_K(x) \neq K$, depending on the choice of x, such that $x \in \partial K'_K(x)$. For $x \in \partial K \cap \Gamma$ the outer trace is set to be $v_K^- := g$ with g an appropriate boundary function, see Section 2.6.

Since below it will be always clear from the context which element K in the subdivision T_h the quantities v_K^+ and v_K^- correspond to, for the sake of notational simplicity, we shall suppress the letter K in the subscript and write, respectively, v^+ and v^- instead.

To formulate the discontinuous Galerkin method, we first introduce a weak formulation of (2.2). To this end, we multiply the conservation law (2.2) by an arbitrary smooth function v and integrate by parts over an element K in the mesh T_h; thereby, we get

$$-\int_K F(u) \cdot \nabla v \, dx + \int_{\partial K} F(u) \cdot n \, v \, ds = 0, \tag{2.8}$$

where $n|_{\partial K}$ denotes the unit outward normal vector to ∂K.

To discretise (2.8), we replace the analytical solution u by the Galerkin finite element approximation u_h and the test function v by v_h, where u_h and v_h both belong to the finite element space V_h^p. In addition, since the numerical solution u_h is discontinuous between element interfaces, we must replace the flux $F(u) \cdot n$ by a *numerical flux* function $H(u_h^+, u_h^-, n)$, which depends on both the inner– and outer–trace of u_h on ∂K, $K \in T_h$, and the unit outward normal n to ∂K. Thereby, summing over the elements K in the mesh T_h, yields the discontinuous Galerkin finite element discretisation of (2.2) as follows: find $u_h \in V_h^p$ such that

$$\sum_{K \in T_h} \left\{ -\int_K F(u_h) \cdot \nabla v_h \, dx + \int_{\partial K} H(u_h^+, u_h^-, n) \, v_h^+ \, ds \right\} = 0 \quad \forall v_h \in V_h^p, \tag{2.9}$$

cf. [4, 14, 16, 18], for example. This scheme is called discontinuous Galerkin method of degree p, or in short notation "DG(p) method".

For elements $K \in T_h$ whose boundaries intersect that of the computational domain Ω, we replace u_h^- by appropriate boundary conditions on the portion of ∂K for which $\partial K \cap \Gamma \neq \emptyset$. For more details about boundary conditions, see Section 2.6.

We remark that the replacement of the flux $F(u) \cdot n$ by the numerical flux function $H(u_h^+, u_h^-, n)$ on the boundary of element K, K in T_h, corresponds to the weak imposition of the boundary data, cf. [4, 33]. Furthermore, we emphasise that the choice of the numerical flux function is *independent* of the finite element space employed. Indeed, the numerical flux $H(\cdot, \cdot, \cdot)$ may be chosen to be any two–point monotone Lipschitz function which satisfies the following two conditions:

(i) $H(\cdot,\cdot,\cdot)|_{\partial K}$ is consistent with the flux $F(\cdot)\cdot n$ for each K in T_h; i.e.

$$H(v,v,n)|_{\partial K} = F(v)\cdot n \quad \forall K \in T_h;$$

(ii) $H(\cdot,\cdot,\cdot)$ is conservative, i.e. given any two neighbouring elements K and K' from the finite element partition T_h, at each point $x \in \partial K \cap \partial K' \neq \emptyset$, noting that $n_{K'} = -n$, we have that

$$H(v,w,n) = -H(w,v,-n).$$

There are several numerical flux functions satisfying these conditions, such as the Godunov, Engquist–Osher, Lax–Friedrichs, Roe or the Vijayasundaram flux, for example; cf. Kröner [39] and Toro [54], and the references cited therein. As examples, here we consider two different numerical fluxes: the (local) Lax–Friedrichs flux and the Vijayasundaram flux.

The (local) Lax–Friedrichs flux $H_{LF}(\cdot,\cdot,\cdot)$, is defined by

$$H_{LF}(u^+,u^-,n)|_{\partial K} = \tfrac{1}{2}\left(F(u^+)\cdot n + F(u^-)\cdot n + \alpha\left(u^+ - u^-\right)\right), \qquad (2.10)$$

for $K \in T_h$, where α is the maximum over u^+ and u^-,

$$\alpha = \max_{v=u^+,u^-}\{|\lambda(B(v,n))|\},$$

of the largest eigenvalue (in absolute value) $|\lambda(B)|$ of the matrix

$$B(u_h,n) = F'(u_h)\cdot n \equiv \sum_{i=0}^{d} A_i(u_h)\,(n)_i\,.$$

Here, A_i, $i = 0,\ldots,d$, are the Jacobi matrices defined in (2.1); $(n)_i$ denotes the ith component of the unit outward normal vector n to element K, $K \in T_h$, for $i = 0,\ldots,d$.

The Vijayasundaram flux $H_V(\cdot,\cdot,\cdot)$, is defined by

$$H_V(u^+,u^-,n)|_{\partial K} = B^+(\bar{u},n)u^+ + B^-(\bar{u},n)u^- \quad \text{for } K \in T_h, \qquad (2.11)$$

where $B^+(\bar{u},n)$ and $B^-(\bar{u},n)$ denote the positive and negative parts, cf. (2.6), of the matrix $B(\bar{u},n)$, respectively, evaluated at some average state $\bar{u}$ between u^+ and u^-.

Remark 2.1 *We note that discontinuous Galerkin discretisations are similar to finite Volume schemes, especially in the use of numerical fluxes. In fact, the basic finite Volume scheme exactly corresponds to the DG(0) method, i.e. to the discontinuous Galerkin method using piecewise constants. Consequently, the DG(p) methods with $p > 0$ can be regarded as the "natural" generalization of finite Volume methods to higher order methods. Indeed, by simply increasing the degree p of the discrete function space V_h^p in the discretisation (2.9) we gain DG methods of corresponding higher orders. This, we demonstrate by a numerical example of the 2D compressible Euler equations in Section 2.4.*

2.2.1 Scalar hyperbolic equation

Originally, the discontinuous Galerkin method was introduced by Reed and Hill [45] and analysed by LeSaint and Raviart [41] for the scalar hyperbolic problem

$$\begin{aligned} \beta \cdot \nabla u + bu &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_-, \end{aligned} \tag{2.12}$$

with constant vector $\beta \in R^2$ and a real number $b > 0$. This linear advection equation is used as simple model for neutron transport. The discontinuous Galerkin discretisation of this problem is given by

$$\sum_{K \in T_h} \left\{ \int_K (\beta \cdot \nabla u_h + b u_h) v_h \, dx - \int_{\partial K_-} \beta \cdot n \, [u_h] v_h^+ \, ds \right\} = \sum_{K \in T_h} \left\{ \int_K f v_h \, dx \right\}, \tag{2.13}$$

with the jump $[u_h] = u_h^+ - u_h^-$, the inflow boundary of the element,

$$\partial K_- = \{x \in \partial K, \beta \cdot n < 0\},$$

the outflow boundary $\partial K_+ = \Gamma \setminus \partial K_-$, and inflow boundary values

$$u_h^-(x) = g(x), \qquad x \in \Gamma_-,$$

see also Johnson and Pitkäranta [38], for example. At first sight, the discontinuous Galerkin discretisation given by (2.9) looks different to the original DG discretisation (2.13). However, by defining the numerical flux for the linear advection equation to be

$$H(u_h^+, u_h^-, n)(x) = \begin{cases} \beta \cdot n \, u_h^-, & \text{for } (\beta \cdot n)(x) < 0, \text{ i.e. } x \in \partial K_-, \\ \beta \cdot n \, u_h^+, & \text{for } (\beta \cdot n)(x) \geq 0, \text{ i.e. } x \in \partial K_+, \end{cases} \tag{2.14}$$

we see that after integration by parts equation (2.9) reduces to (2.13) with $b = 0$ and $f = 0$. We recall that the numerical flux replaces the flux term $F(u) \cdot n$ on the boundary of the elements. The normal flux of the linear advection equation is given by $F(u) \cdot n = \beta \cdot nu$, and the numerical flux H, see (2.14), switches between the interior and the outer traces, u_h^+ and u_h^-, respectively, depending on $x \in \partial K$ being in the outflow or the inflow part of the boundary of the element. There are many generalisations of this numerical flux to systems of equations, but, like the Lax-Friedrichs flux or the Vijayasundaram flux, they simply reduce to (2.14) when applied to the linear advection equation.

2.3 Short review of *a priori* error estimates

In the following we quote an *a priori* error estimate of the discontinuous Galerkin method for the scalar hyperbolic boundary value problem given in (2.12):

Let $b \in C(\bar{\Omega})$, $f \in L^2(\Omega)$, and $g \in L^2(\Gamma_-)$ on the inflow boundary $\Gamma_- := \{x \in \Gamma, a(x) \cdot n(x) < 0\}$, then the weak formulation of (2.12) given by,

$$\int_\Omega (\beta \cdot \nabla u + bu)v \, dx = \int_\Omega fv \, dx, \qquad v \in L^2(\Omega),$$
$$\int_{\Gamma_-} \beta \cdot n \, uw \, ds = \int_{\Gamma_-} \beta \cdot n \, gw \, ds, \qquad w \in L^2(\Gamma_-),$$

has a unique (weak) solution $u \in L^2(\Omega)$ with $\beta \cdot \nabla u \in L^2(\Omega)$.

We recall the discontinuous Galerkin discretisation (2.13) to this problem and write it as follows: find $u_h \in V_h^p$ such that

$$a(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h^p, \tag{2.15}$$

where here $a(\cdot,\cdot)$ and $l(\cdot)$ are given by

$$a(w,v) = \sum_{K \in T_h} \left\{ \int_K (\beta \cdot \nabla w + bw)v \, dx \right.$$
$$\left. - \int_{\partial K_- \setminus \Gamma_-} \beta \cdot n \, [w]v^+ \, ds - \int_{\partial K_- \cap \Gamma_-} \beta \cdot n \, w^+ v^+ \, ds \right\}, \tag{2.16}$$

with $[w] = w^+ - w^-$, and

$$l(w) = \sum_{K \in T_h} \left\{ \int_K fv \, dx - \int_{\partial K_- \cap \Gamma_-} \beta \cdot n \, gv^+ \, ds \right\}.$$

Furthermore, we define the norms $\|v\|_K \equiv \|v\|_{L^2(K)} = \left(\int_K |v|^2 \, dx\right)^{\frac{1}{2}}$ for $K \in T_h$, and $\|v\|_e = \left(\int_e |\beta \cdot n||v|^2 \, dx\right)^{\frac{1}{2}}$ for $e \subset \partial K$. Then the following lemma can be obtained by performing integration by parts.

Lemma 2.2 *Let $a(\cdot,\cdot)$ be the bilinear form defined in* (2.16)*, then the following identity holds*

$$a(v,v) = \sum_{K \in T_h} \left\{ \int_K \left(b - \tfrac{1}{2}\nabla \cdot \beta\right) v^2 \, dx + \tfrac{1}{2}\|v^+\|^2_{\partial K_- \cap \Gamma_-} \right.$$
$$\left. + \tfrac{1}{2}\|v^+ - v^-\|^2_{\partial K_- \setminus \Gamma_-} + \tfrac{1}{2}\|v^+\|^2_{\partial K_+ \cap \Gamma_+} \right\}. \tag{2.17}$$

An implication of this Lemma is a stability result quoted in the following Lemma 2.3 that is a special case of Lemma 2.4 in [33] with $\delta = 0$, i.e. without streamline diffusion stabilisation.

Lemma 2.3 (Stability) *Suppose that there exists a positive constant c_0 such that*

$$c(x) := b(x) - \tfrac{1}{2}\nabla \cdot \beta(x) \geq c_0, \quad x \in \bar{\Omega}. \tag{2.18}$$

Then the discrete solution $u_h \in V_h^p$ of (2.15) obeys the bound

$$\sum_{K\in T_h} \left\{ c_0 \|u_h\|_K^2 + \|u_h^+ - u_h^-\|_{\partial K_- \setminus \Gamma_-}^2 + \|u_h^+\|_{\partial K_+ \cap \Gamma_+}^2 + \tfrac{1}{2}\|u_h^+\|_{\partial K_- \cap \Gamma_-}^2 \right\} \leq \tfrac{1}{c_0}\|f\|_\Omega^2 + 2\|g\|_{\Gamma_-}^2 .$$

Lemma 2.3 implies the uniqueness of the solution u_h of the discontinuous Galerkin method (2.15); furthermore, since (2.15) is a linear problem over the finite-dimensional space V_h^p, the existence of the solution u_h follows from its uniqueness.

Stimulated by the identity in Lemma (2.2) and definition (2.18), we define the DG-Norm $|||\cdot|||$ by

$$|||v||| := \sum_{K\in T_h} \left\{ \|cv\|_K^2 + \tfrac{1}{2}\|v^+\|_{\partial K_- \cap \Gamma_-}^2 + \tfrac{1}{2}\|v^+ - v^-\|_{\partial K_- \setminus \Gamma_-}^2 + \tfrac{1}{2}\|v^+\|_{\partial K_+ \cap \Gamma_+}^2 \right\}.$$

In terms of this norm the following convergence result for the discontinuous Galerkin method applied to the scalar hyperbolic equation (2.12) can be shown, that is (in a simplified version) quoted from [33].

Theorem 2.4 (Convergence rate of the DG method) *Let T_h consist of shape-regular quadrilateral elements. For all $K \in T_h$ let $u|_K \in H^{k_K+1}(K)$, $k_K \geq 0$. Then, for the discrete solution $u_h \in V_h^p$ of the discontinuous Galerkin problem (2.15) and for $0 \leq s_K \leq min(p, k_K)$ the following estimate holds:*

$$|||u - u_h|||^2 \leq C \sum_{K\in T_h} h_K^{2s_K+1} |u|_{s_K+1,K}^2, \tag{2.19}$$

where C is independent of s_K and h_K.

This result was first proven by Johnson and Pitkaränta in [38] for general triangulations. However, this result indicates half an order of convergence in the L^2-norm less than expected from a trial space V_h^p of order $p+1$. Also numerical results show a full $O(h^{p+1})$ order of convergence on virtually all meshes, see also Section 2.4. In fact, Richter obtained in [46] the full order of convergence in the L^2-norm for some structured two-dimensional but non-Cartesian meshes. But on the other hand, Peterson confirmed in [43] by considering so–called Peterson meshes, that actually $O(h^{p+1/2})$ is the optimal order of convergence in the L^2-norm on general meshes.

2.4 Convergence of the discontinuous Galerkin method for the 2D Euler equations

For nonlinear hyperbolic systems like the 2D compressible Euler equations there are virtually no *a priori* error estimates available. Therefore, the order of convergence of these problems can be obtained through numerical tests, only. In

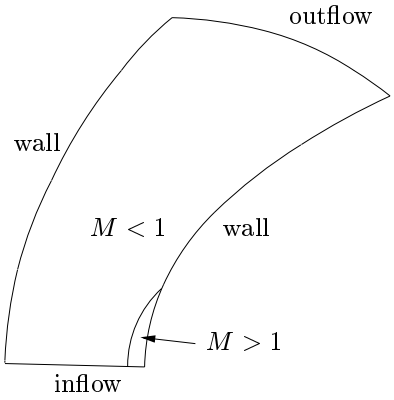


Figure 2.1: Geometry for Ringleb's flow; M denotes the Mach number.

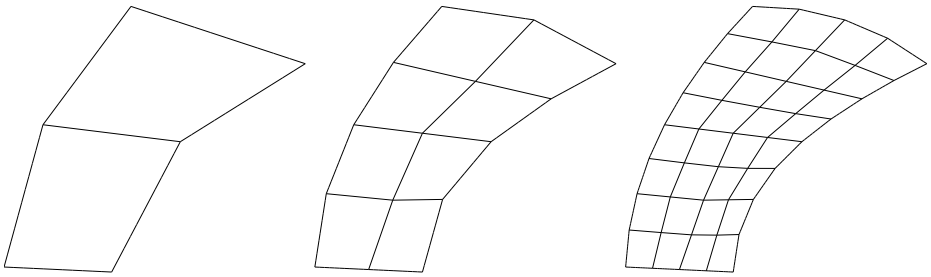


Figure 2.2: Ringleb flow problem: Meshes with 2, 8, 32 elements.

the following, we shall show a numerical example indicating that the discontinuous Galerkin method applied to the steady 2D compressible Euler equation exhibits a full $O(h^{p+1})$ order of convergence on smooth solutions.

To this end, we consider the solution of the 2D compressible Euler equations to the Ringleb flow problem, that is one of the few non-trivial problems of the 2D Euler equations for which a (smooth) analytical solution is known. For this case the analytical solution may be obtained using the hodograph transformation, see [13] or Appendix B.5. This problem represents a transonic flow in a channel, see Figure 2.1, with inflow and outflow boundaries given by the lower and upper boundaries of the domain, and non–absorbing, i.e. *reflective* boundaries with normal velocity $v \cdot n = 0$, on the left and right boundary.

The solution to this flow problem is smooth but it is transonic with a small supersonic region near the lower right corner. Furthermore, this problem includes curved boundaries and the mesh consists of arbitrary quadrilaterals; see the first three globally refined meshes in Figure 2.2.

We note that low order boundary approximations of *reflective* boundaries reduce the order of convergence, cf. Section 2.9.2. To suppress this effect, here

# el.	# dofs	L^2-error	Rate	Order
2	32	8.48e-02	-	-
8	128	2.39e-02	3.54	1.82
32	512	6.72e-03	3.56	1.83
128	2048	1.81e-03	3.72	1.90
512	8192	4.75e-04	3.80	1.93
2048	32768	1.23e-04	3.87	1.95
8192	131072	3.14e-05	3.92	1.97

Table 2.1: Ringleb flow problem: DG(1) is of order $O(h^2)$.

# el.	# dofs	L^2-error	Rate	Order
2	72	2.06e-02	-	-
8	288	2.75e-03	7.52	2.91
32	1152	3.85e-04	7.13	2.83
128	4608	5.18e-05	7.43	2.89
512	18432	6.61e-06	7.83	2.97
2048	73728	8.22e-07	8.05	3.01

Table 2.2: Ringleb flow problem: DG(2) is of order $O(h^3)$.

we impose the boundary condition,

$$B^-(u,n)\,(u-g) = 0, \quad x \in \partial\Omega, \tag{2.20}$$

on the *whole* boundary $\partial\Omega$ of the domain, where g is the boundary value function taken from the exact solution to the Ringleb flow problem. This boundary condition represents an inflow boundary condition for characteristic variables on inflow parts (with respect to the corresponding characteristics) of the boundary.

In the following, we compute the numerical solutions to this problem on globally refined meshes and evaluate the L^2-error of the solutions. The results of this test are presented in Tables 2.1- 2.4. They show the number of elements of the globally refined meshes, the number of degrees of freedom, the L^2-error of the numerical solution and two additional columns including the rate and the order of the convergence. These tables clearly show a $O(h^{p+1})$ order of convergence of the L^2-error $\|u-u_h\|$ of the numerical solution u_h. Finally Figure 2.3 shows the L^2-error of the DG(p), $0 \le p \le 4$ methods plotted against the number of degrees of freedom. The resulting $O(h^{p+1})$ order of convergence is optimal for trial and test functions of polynomial degree p. This again, as discussed in Section 2.3 for the scalar case, is half an order more than expected for the scalar linear case on general meshes. We re-emphasise that, although the discontinuous Galerkin method is based on ideas of upwinding, it is *not* restricted to convergence of first order but allows arbitrarily high orders of convergence (for smooth solutions) depending on the order of the discrete function space V_h^p employed.

# el.	# dofs	L^2-error	Rate	Order
2	128	5.85e-03	-	-
8	512	3.58e-04	16.37	4.03
32	2048	2.84e-05	12.60	3.66
128	8192	1.92e-06	14.78	3.89
512	32768	1.23e-07	15.61	3.96
2048	131072	7.81e-09	15.75	3.98

Table 2.3: Ringleb flow problem: DG(3) is of order $O(h^4)$.

# el.	# dofs	L^2-error	Rate	Order
2	200	2.92e-03	-	-
8	800	6.92e-05	42.13	5.40
32	3200	2.25e-06	30.81	4.95
128	12800	7.77e-08	28.94	4.86
512	51200	2.56e-09	30.32	4.92
2048	204800	8.21e-11	31.21	4.96

Table 2.4: Ringleb flow problem: DG(4) is of order $O(h^5)$.

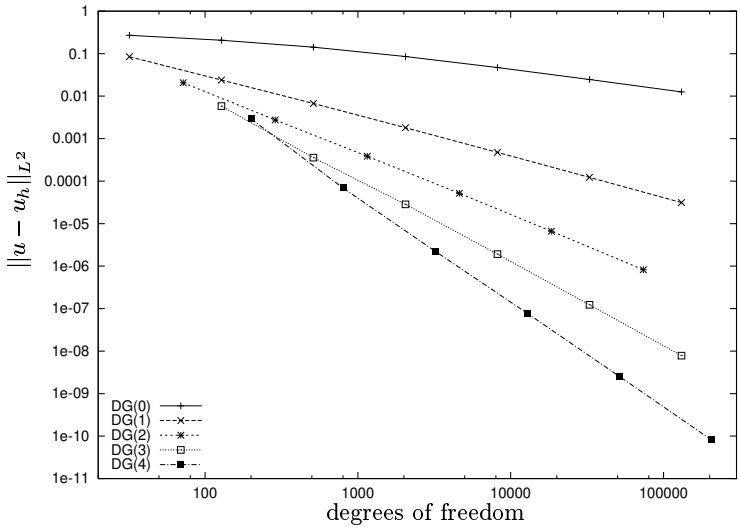


Figure 2.3: Ringleb flow problem: L^2-error of DG(p), $0 \leq p \leq 4$, solutions.

2.5 Comparison of the SD and DG method

This work deals with the discretisation of hyperbolic problems by the *discontinuous Galerkin* finite element method, "DG method" for short. This method uses trial and test spaces that include *discontinuous* discrete functions; given that solutions to hyperbolic problems typically include discontinuities like shocks and contact discontinuities, this seems to be a natural choice. However, there are several other finite element methods for solving hyperbolic problems, one of the most prominent ones is the streamline diffusion method, "SD method" for short. This method employs continuous trial and test spaces and is based on the standard Galerkin discretisation that, for stability reasons, is enhanced by an artificial viscosity term that acts in streamline direction, only.

This section is devoted to the comparison of the (continuous) SD method and the DG method. We shall compare these methods with respect to two major topics: first, the accuracy of the methods in approximating smooth or discontinuous solutions, and second, the conservation (of flux) properties inherent in these methods, that are essential for approximating solutions to hyperbolic conservation laws.

For simplicity, we restrict this comparison to the simplest possible hyperbolic problem, the linear advection equation given in (2.12) with $b = 0$, i.e.

$$\begin{aligned} \beta \cdot \nabla u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_-, \end{aligned}$$

with constant advection direction $\beta \in \mathbb{R}^2$. From (2.13) we recall the discontinuous Galerkin discretisation of this equation: find $u_h \in V_h^p$ such that

$$\sum_{K \in T_h} \left\{ \int_K \beta \cdot \nabla u_h v_h \, dx - \int_{\partial K_-} \beta \cdot n \, [u_h] v_h^+ \, ds \right\} = \sum_{K \in T_h} \int_K f v_h \, dx \quad \forall v_h \in V_h^p, \tag{2.21}$$

where $[u_h] = u_h^+ - u_h^-$ on $\partial K_- \setminus \Gamma_-$ and $[u_h] = u_h^+ - g$ on Γ_-. Furthermore, V_h^p denotes the discrete function space of discontinuous piecewise polynomial functions of degree $p \geq 0$ defined in (2.7). Additionally, we define the finite element function space $V_h^{c,p}$ of *continuous* piecewise polynomial functions of degree $p \geq 1$,

$$V_h^{c,p} = \{ v_h \in V_h^p : v_h \text{ continuous in } \Omega \}, \tag{2.22}$$

and introduce the streamline diffusion method (see [36] for example) for the linear advection equation (2.5): find $u_h \in V_h^{c,p}$ such that

$$\begin{aligned} \sum_{K \in T_h} \int_K \beta \cdot \nabla u_h (v_h + \delta_K \beta \cdot \nabla v_h) \, dx - \int_{\Gamma_-} \beta \cdot n \, [u_h] v_h^+ \, ds \\ = \sum_{K \in T_h} \int_K f (v_h + \delta_K \beta \cdot \nabla v_h) \, dx \quad \forall v_h \in V_h^{c,p}, \end{aligned} \tag{2.23}$$

where δ_K denotes a stabilisation parameter given by $\delta_K = C_\delta h_K$ and C_δ is a fixed positive constant that is set to $C_\delta = 1$ in the numerical tests below.

The order of convergence of the SD method is known to be $O(h^{p+1/2})$, see [37] for example. Thereby, both methods, the DG method and the SD method, are of the same order of convergence. Nevertheless, there are differences in the accuracy of the two methods. This will be tested numerically in the following subsection. In the subsection thereafter we discuss the conservation properties of the two methods and draw some conclusions in the last subsection.

2.5.1 Accuracy

In the following, we numerically test the accuracy of the DG and the SD method given in (2.21) and (2.23), respectively. To this end, we consider the linear advection equation (2.5) on the unit square $\Omega = [0,1]^2$ with $\beta = \frac{1}{5}(4,3)$ and zero right hand side, $f = 0$. First, we consider a smooth boundary value function g that yields the following *smooth* solution to the linear advection equation (2.5)

$$u(x) = \exp\left(x_2 - \tfrac{\beta_2}{\beta_1} x_1\right).$$

In Figures 2.4(a) and (b) we show the $L^2(\Omega)$-error of the numerical solution plotted against the number of elements and the number of degrees of freedom. First we note that the methods considered show an order of convergence of exactly $O(h^{p+1})$ if piecewise polynomials of degree p are employed. Furthermore, we see that the streamline diffusion method and the DG method of given degree p and the same number of elements are both of approximately the same accuracy. Given that the DG method involves more degrees of freedom per element than the SD method this shows that for smooth solutions these additional degrees of freedom do not result in a better accuracy. Indeed, in Figure 2.4 (b) we clearly see that for a given number of degrees of freedom the SD method is more accurate than the DG method. Only on the coarsest meshes, the SD method (with $C_\delta = 1$) is slightly less accurate than the DG method. But, numerical tests show that even on the coarsest meshes the DG method is not more accurate than the SD method, provided C_δ is sufficiently small.

As second example, we consider a discontinuous boundary value function

$$g(x) = \begin{cases} 1 & \text{for } x \in \{0\} \times [0,0], \\ 0 & \text{otherwise,} \end{cases}$$

that results in the following *discontinuous* solution to equation (2.5)

$$u(x) = \begin{cases} 1 & \text{for } x_2\beta_1 \geq x_1\beta_2, \\ 0 & \text{otherwise.} \end{cases}$$

In Figure 2.5, we show the $L^2(\Omega)$-error of the numerical solution plotted against the number of degrees of freedom. First, we see that for this discontinuous solution all methods of degree $p \geq 1$ are of approximately the same accuracy. Furthermore, we see that on globally refined meshes the streamline diffusion

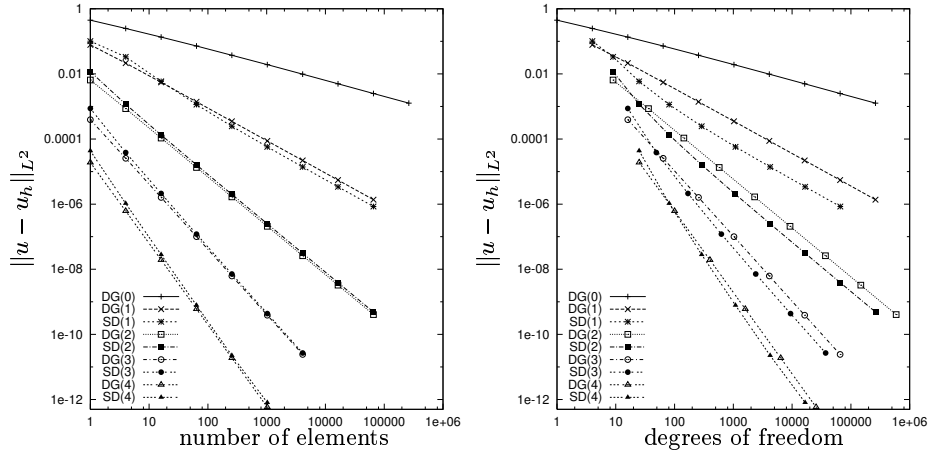


Figure 2.4: *Smooth* solution to the linear advection equation: $L^2(\Omega)$-error of DG(p), $0 \leq p \leq 4$, and SD(p), $1 \leq p \leq 4$, numerical solutions. (a) Error vs. number of elements; (b) Error vs. number of degrees of freedom.

method of a given degree p and a given number of degrees of freedom is slightly more accurate than the corresponding discontinuous Galerkin method, whereas on locally refined meshes they show approximately the same accuracy. We note that in the smooth parts of the solution the SD method is again more accurate in terms of degrees of freedom than the DG method. This is the same effect that appears in the previous smooth solution example. However, this effect is superposed by the fact that the DG method is slightly more accurate in the neighborhood of discontinuities. On locally refined meshes, the mentioned effect is totally covered as here the number of elements located in the smooth parts of the solution are significantly less than those located in the neighborhood of the discontinuity. Finally, in Figure 2.6 we show several numerical solutions on globally refined mesh: the DG(1) solution on 64 and 256 elements, the DG(0) solution on 1024 elements and the SD(1) solution on 256 elements; in Figure 2.8 we show the DG(1) and the SD(1) solution on 1081 elements on a locally refined mesh.

2.5.2 Global and local conservation property

Now, we check the SD and the DG method for inherent global and local conservation properties. In the following, we show that the SD method implies a global conservation (of flux) property, whereas the DG method implies both, a global *and* a local conservation property.

First, we consider the discrete solution u_h to the SD method (2.23). In (2.23) we set $v_h \equiv 1$ and the stabilisation term $\delta_K \beta \cdot \nabla v_h$ vanishes. Integration by

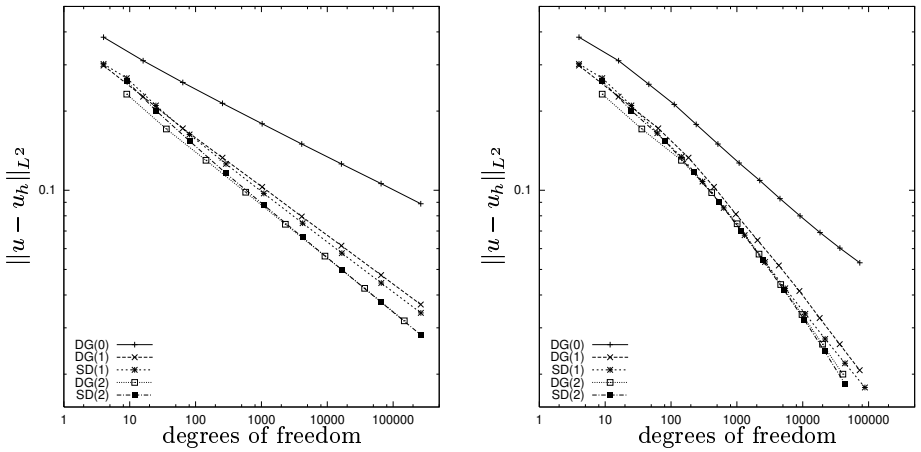


Figure 2.5: *Discontinuous* solution to the linear advection equation: $L^2(\Omega)$-error of DG(p), $0 \leq p \leq 2$, and SD(p), $p = 1, 2$, numerical solutions. (a) Global refinement; (b) Local refinement.

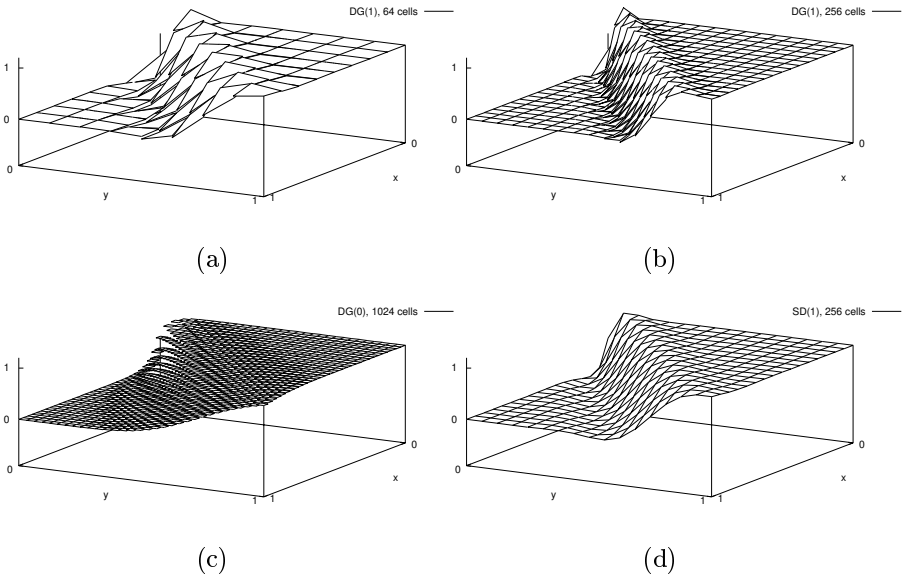


Figure 2.6: *Discontinuous* solution to the linear advection equation: (a) DG(1) on 64 elements with 256 degrees of freedom (DoFs); (b) DG(1) on 256 elements with 1024 DoFs; (c) DG(0) on 1024 elements with 1024 DoFs and (d) SD(1) on 256 elements with 289 DoFs.

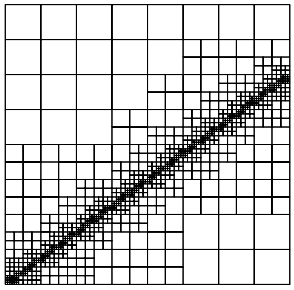
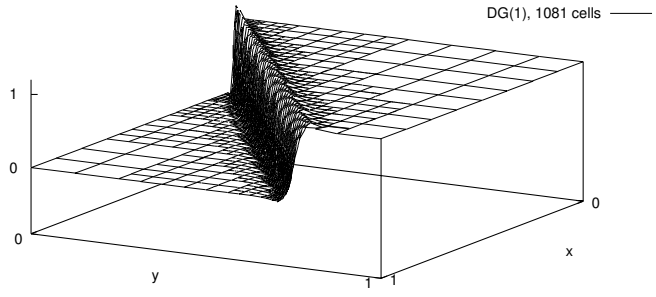
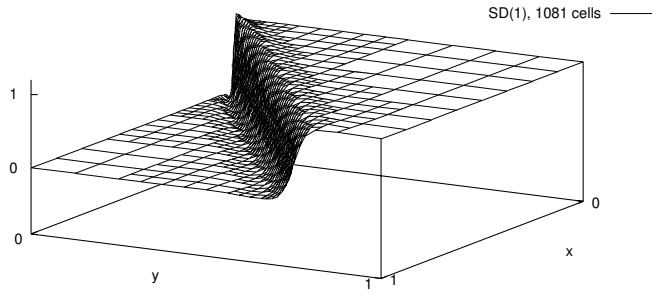


Figure 2.7: Locally refined mesh with 1081 elements for the *discontinuous* solution to the linear advection equation.



(a)



(b)

Figure 2.8: *Discontinuous* solution to the linear advection equation on locally refined mesh with 1081 elements: (a) DG(1) and (b) SD(1) numerical solution.

parts gives the following *global* convervation property

$$\int_{\Gamma_-} \beta \cdot n \, g \, ds + \int_{\Gamma_+} \beta \cdot n \, u_h^+ \, ds = \int_\Omega f \, dx. \tag{2.24}$$

The global conservation property of the DG method (2.21) can be shown analogously. For proofing the *local* conservation property of the DG method on a fixed element K we recall (2.21) and choose

$$v_h(x) = \begin{cases} 1, & \forall x \in K, \\ 0, & \text{otherwise}, \end{cases}$$

a possible choice of test function as $v_h \in V_h^d$. Again, by integration by parts we deduce following *local* conservation property

$$\int_{\partial K_-} \beta \cdot n \, u_h^- \, ds + \int_{\partial K_+} \beta \cdot n \, u_h^+ \, ds = \int_K f \, dx, \quad \forall K \in T_h, \tag{2.25}$$

for the discrete solution u_h to the DG method (2.21).

With zero right hand side, $f = 0$, this property reduces to

$$\int_{\partial K_-} \beta \cdot n \, u_h^- \, ds + \int_{\partial K_+} \beta \cdot n \, u_h^+ \, ds = 0, \quad \forall K \in T_h, \tag{2.26}$$

meaning that on each element K the outflow, $\int_{\partial K_+} \beta \cdot n \, u_h^+ \, ds$, is equal to the inflow, $\int_{\partial K_-} \beta \cdot n \, u_h^- \, ds$.

Next, we introduce a value $d_K := d_K(u_h)$ on each element K, with

$$d_K(u_h) := \begin{cases} \left| \int_{\partial K_+} \beta \cdot n \; u_h^+ \; ds \right| & \text{for } u_h^-|_{\partial K_-} \equiv 0, \\ \left| \dfrac{\int_{\partial K_-} \beta \cdot n \; u_h^- \; ds + \int_{\partial K_+} \beta \cdot n \; u_h^+ \; ds}{\int_{\partial K_-} \beta \cdot n \; u_h^- \; ds} \right| & \text{otherwise}, \end{cases} \tag{2.27}$$

that indicates by what extent the *local* conservation property (2.26) is violated. For the DG method, this value is zero, $\max_K d_K = 0$, due to the local conservation property (2.26) of the DG method, but it is generally nonzero for methods without an inherent local conservation property. Analogously, we define for a (non–homogeneous) boundary value function g the value $d_\Omega := d_\Omega(u_h)$, with

$$d_\Omega(u_h) := \left| \left(\int_{\Gamma_-} \beta \cdot n \; g \; ds + \int_{\Gamma_+} \beta \cdot n \; u_h^+ \; ds \right) / \int_{\Gamma_-} \beta \cdot n \; g \; ds \right|, \tag{2.28}$$

that indicates by what extend the *global* conservation property (2.24) is violated.

In the following, we numerically test the indicators d_Ω and $\max_K d_K$ of the global and local conservation property, respectively. They are computed for the SD(1) and the DG(1) method in combination with both, smooth and discontinuous solutions to the linear advection equation (2.5), see Tables 2.5 and 2.6, respectively. Each of these tables includes the values d_Ω and $\max_K d_K$ evaluated

	SD method			DG method	
# el.	d_Ω	$\max_K d_K$		d_Ω	$\max_K d_K$
16	-5.82e-14	4.50e-03	-	1.88e-16	2.66e-16
64	-6.64e-15	1.10e-03	4.10	5.47e-16	2.70e-16
256	2.05e-15	2.75e-04	4.00	7.44e-16	4.61e-16
1024	5.29e-16	6.84e-05	4.02	1.45e-15	4.46e-16
4096	-1.48e-15	1.71e-05	4.00	2.24e-15	2.15e-15

Table 2.5: Conservation properties for a *smooth* solution.

	SD method			DG method	
# el.	d_Ω	$\max_K d_K$		d_Ω	$\max_K d_K$
16	-1.87e-14	1.08e-01	-	6.00e-17	2.38e-16
64	-9.39e-14	1.31e-01	0.83	6.30e-16	3.16e-16
256	-2.43e-14	1.31e-01	1.00	1.16e-15	3.97e-16
1024	8.08e-15	1.31e-01	1.00	1.36e-15	3.97e-16
4096	1.33e-14	1.31e-01	1.00	1.28e-15	4.76e-15

Table 2.6: Conservation properties for a *discontinuous* solution.

for the numerical solutions on a sequence of globally refined meshes for both, the SD and the DG method. Additionally, the tables show the convergence rates of the $\max_K d_K$ values of the SD method. In agreement with the theory above, we see that the DG method involves a global *and* local conservation property, whereas the SD method shows a global conservation property, only. Furthermore, for the SD method on smooth solutions, $\max_K d_K$ converges to zero with the order h^2; i.e. the SD method exhibits a local conservation property in the limit $h \to 0$, only. Finally, we see that for discontinuous solutions the SD method shows no local conservation property, indicated by the fact that $\max_K d_K \to C > 0$ as $h \to 0$. Further numerical tests show that the maximum values of d_K are placed in the neighborhood of the discontinuity of the solution. Outside a neighborhood of the discontinuity the SD method shows a local conservation property in the limit $h \to 0$ like for smooth solutions.

We note that $d_K(u) = 0$ holds when u is the exact solution of problem 2.5. Thereby, given an element K with $u_h^-|_{\partial K} \not\equiv 0$, we have

$$\begin{aligned} d_K = d_K(u_h) &= d_K(u_h) - \frac{\int_{\partial K_-} \beta \cdot n \; u \, ds}{\int_{\partial K_-} \beta \cdot n \; u_h^- \, ds} d_K(u) \\ &= \frac{\int_{\partial K_-} \beta \cdot n \; (u_h^- - u) \, ds + \int_{\partial K_+} \beta \cdot n \; (u_h^+ - u) \, ds}{\int_{\partial K_-} \beta \cdot n \; u_h^- \, ds} \leq c \| u - u_h \|_{L^\infty}. \end{aligned}$$

In agreement with the results in Table 2.5, this verifies that the order of convergence of $\max_K d_K$ is at least the order of convergence of u_h in the L^∞-norm.

2.5.3 Conclusion

First we summarise the results of this section.

- – On *smooth* solutions the DG method requires the same number of elements for achieving the same accuracy as the SD method. But, in terms of degrees of freedom, the DG method is less accurate than the SD method as the former uses more degrees of freedom per element.
  – In contrast, when applied to problems with *discontinuous* solutions, the SD and DG methods are approximately of the same accuracy for the same number of degrees of freedom.
- The DG method implies a global and a local conservation property, whereas the SD method implies a global conservation property, only. A local conservation property of the SD method is given in the limit of $h \to 0$ and in smooth parts of the solution, only.

Furthermore, we note that for the DG discretisation of the linear advection equation (2.5) only one Block-Gauss-Seidel step is required for solving the linear problem when an appropriate numbering of the elements is employed, see Section 2.8.1. In comparison to that there are several BiCGstab or GMRES steps needed for solving the linear system arising from the SD discretisation.

The SD method (2.23) includes the stabilisation parameter C_δ. As the numerical solution depends on the specific choice of this parameter a determination of an optimal value of C_δ is necessary. In comparison to that, the DG method (2.21) is parameter-free.

Finally, we note that for the DG method the degree of polynomial basis functions can easily be chosen independently on each element. This leads to so–called *hp–methods* where the actual degree on each element is chosen according to the smoothness of the solution. These methods are much easier to implement for discontinuous than for continuous finite element methods since the support of discontinuous basis functions consist of a single element only, and a change of the polynomial degree in one element does not affect the discretisation in neighboring elements. These hp–methods will not be covered in this work, instead we refer to [48], [33] and [34], for example.

2.6 Boundary conditions

For the linear advection equation (2.12) the boundary Γ of the domain can easily be separated into an inflow and an outflow boundary part, Γ_- and Γ_+, respectively, depending on whether the characteristics are entering or leaving the domain, i.e. $\beta \cdot n < 0$ or $\beta \cdot n > 0$, respectively. In contrast to that, for systems of equations, there are in general several distinct characteristic directions. This makes the imposition of boundary conditions much more complicated.

For the 2D Euler equations the numerical flux function H replaces the term

$$F(u) \cdot n = F'(u) \cdot n\, u = B(u, n)u,$$

where here we used the definition (2.3) of the matrix $B(u,n)$ and employed the homogeneity property $F_i(u) = F_i'(u)u$, $i = 1,2$, of the 2D Euler equations, see [50] for example. Furthermore, there are three characteristic directions corresponding to the three different eigenvalues of the matrix $B(u,n)$,

$$\lambda_1 = v \cdot n - c, \qquad \lambda_2 = \lambda_3 = v \cdot n, \qquad \lambda_4 = v \cdot n + c, \tag{2.29}$$

with $c := \sqrt{\frac{\gamma p}{\varrho}}$ denoting the sound speed, see also Appendix (A.2). According to the sign of the eigenvalues, we consider four different types of boundary

1. Supersonic inflow: $\lambda_i < 0, \quad i = 1, \dots, 4,$
2. Subsonic inflow: $\lambda_i < 0, \quad i = 1, \dots, 3, \quad \lambda_4 > 0,$
3. Subsonic outflow: $\lambda_i > 0, \quad i = 2, \dots, 4, \quad \lambda_1 < 0,$
4. Supersonic outflow: $\lambda_i > 0, \quad i = 1, \dots, 4.$

Additionally, we consider *reflective boundaries*, i.e. solid walls.

In the following, we itemise the discretisation of all mentioned boundary types. To that end, let the boundary Γ of the domain be subdivided into the following distinct parts: the (sub- and supersonic) inflow boundary Γ_{i}, the supersonic outflow boundary $\Gamma_{\mathrm{o,sup}}$, the subsonic outflow boundary $\Gamma_{\mathrm{o,sub}}$ and the reflective boundary Γ_{refl},

$$\Gamma = \Gamma_{\mathrm{i}} \cup \Gamma_{\mathrm{o,sup}} \cup \Gamma_{\mathrm{o,sub}} \cup \Gamma_{\mathrm{refl}}. \tag{2.30}$$

First, we recall the discontinuous Galerkin discretisation, cf. (2.9): find $u_h \in V_h^p$ such that

$$\sum_{K \in T_h} \left\{ - \int_K F(u_h) \cdot \nabla v_h \, dx + \int_{\partial K} H(u_h^+, u_h^-, n) \, v_h^+ \, ds \right\} = 0 \quad \forall v_h \in V_h^p.$$

After splitting the face terms into interior face terms and boundary face terms, $\sum_K \int_{\partial K} = \sum_K \int_{\partial K \setminus \Gamma} \cup \sum_K \int_{\partial K \cap \Gamma}$, we now define $a_\Gamma(u,v)$ to include the boundary face terms as follows

$$a_\Gamma(u,v) = \sum_{K \in T_h} \int_{\partial K \cap \Gamma} H(u^+, u^-, n) \, v^+ \, ds. \tag{2.31}$$

This boundary term consists of several parts

$$a_\Gamma(u,v) = a_{\Gamma_{\mathrm{i}}}(u,v) + a_{\Gamma_{\mathrm{o,sup}}}(u,v) + a_{\Gamma_{\mathrm{o,sub}}}(u,v) + a_{\Gamma_{\mathrm{refl}}}(u,v), \tag{2.32}$$

according to the subdivision (2.30) of the boundary of the domain. Depending on the respective boundary part, these boundary terms are given as follows:

- On the *inflow boundary* Γ_{i} the outer trace u^- is replaced by the given boundary function g

$$a_{\Gamma_{\mathrm{i}}}(u,v) := \sum_{K \in T_h} \int_{\partial K \cap \Gamma_{\mathrm{i}}} H(u^+, g, n) \, v^+ \, ds.$$

- On the *supersonic outflow boundary* $\Gamma_{\mathrm{o,sup}}$ the outer trace u^- is replaced by the inner trace u^+

$$a_{\Gamma_{\mathrm{o,sup}}}(u,v) := \sum_{K\in T_h} \int_{\partial K\cap\Gamma_{\mathrm{o,sup}}} H(u^+,u^+,n)\, v^+\, ds.$$

- On the *subsonic outflow boundary* $\Gamma_{\mathrm{o,sub}}$ one characteristic variable must be imposed. In applications, subsonic outflow boundary conditions are not given in characteristic variables but in primitive variables. In many cases the pressure $p = p_{\mathrm{o,sub}}$ is prescribed. Thereby, on $\Gamma_{\mathrm{o,sub}}$ the outer trace u^- is replaced by a modified state $u^- = u^-_{\mathrm{o,sub}}(u)$

$$a_{\Gamma_{\mathrm{o,sub}}}(u,v) := \sum_{K\in T_h} \int_{\partial K\cap\Gamma_{\mathrm{o,sub}}} H(u^+,u^-_{\mathrm{o,sub}}(u^+),n)\, v^+\, ds,$$

where $u^-_{\mathrm{o,sub}}(u)$ depends on the inner trace $u = (\varrho, \varrho v_1, \varrho v_2, e)$ and the prescribed pressure $p = p_{\mathrm{o,sub}}$ as follows

$$u^-_{\mathrm{o,sub}}(u) := \begin{pmatrix} \varrho \\ \varrho v_1 \\ \varrho v_2 \\ \frac{p_{\mathrm{o,sub}}}{\gamma-1} + \frac{1}{2}\varrho v^2 \end{pmatrix}. \tag{2.33}$$

- On the *reflective boundary* Γ_{refl} the flux $H(u^+,u^-,n)$ includes a modified state $u^- = u^-_{\mathrm{refl}}(u)$.

$$a_{\Gamma_{\mathrm{refl}}}(u,v) := \sum_{K\in T_h} \int_{\partial K\cap\Gamma_{\mathrm{refl}}} H(u^+,u^-_{\mathrm{refl}}(u^+),n)\, v^+\, ds.$$

Here, $u^-_{\mathrm{refl}}(u)$ originates from u by simply inverting the sign of the normal velocity component of u. With $v = (v_1, v_2)$ denoting the velocity of the state $u = (\varrho, \varrho v_1, \varrho v_2, e)$, the velocity $v^- = (v_1^-, v_2^-)$ of u^- is determined by

$$v^- = (v - 2(v\cdot n)n).$$

Thereby, $u^- = u^-_{\mathrm{refl}}(u)$ is given by the following linear mapping,

$$u^-_{\mathrm{refl}}(u) = u^-_{\mathrm{refl}}{}'(u)u := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-2n_1^2 & -2n_1n_2 & 0 \\ 0 & -2n_1n_2 & 1-2n_2^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} u, \tag{2.34}$$

where $n_i, i = 1,2$, are the components of the unit outward normal vector $n = (n_1, n_2)$. This definition ensures that the normal velocity component of the average state $\bar{u} = \frac{1}{2}(u^+ + u^-)$ vanishes, $\bar{v}\cdot n = \frac{1}{2}(v^+ + v^-)\cdot n = 0$.

Despite the fact that the boundary face terms are slightly different on different parts of the boundary Γ, for simplicity, we will write the scheme in the following as given in (2.9). u_h^- on Γ is meant to represent appropriate boundary values according to the four different cases discussed above.

2.7 Shock–capturing

Even though the fluxes at inter-element boundaries are up-winded through the choice of an appropriate numerical flux function $H(\cdot,\cdot,\cdot)$, spurious oscillations may still be generated by the discontinuous Galerkin finite element method (2.9), in the vicinity of sharp features of the analytical solution, when polynomials of higher degree are used, i.e. when $p > 0$. In order to compute physically relevant solutions which do not exhibit such oscillatory behaviour, the basic discontinuous Galerkin scheme (2.9) must be enhanced by the addition of some form of nonlinear dissipation mechanism which does not adversely affect the formal order of accuracy of the scheme. Indeed, much research activity has focused on the development of appropriate stabilisation devices; for example, Cockburn *et al.* [14, 18] have proposed the use of local projection/limiting techniques for the Runge–Kutta discontinuous Galerkin finite element method.

The *a posteriori* error analysis presented in the following chapter is based on a hyperbolic duality argument employing the so–called Galerkin orthogonality property of the finite element method, see (2.39) below. Therefore, any stabilisation technique used to enhance the numerical performance of the standard discontinuous Galerkin method (2.9) should not violate this orthogonality property. To this end, we add an artificial viscosity term (also referred to as a 'shock–capturing' term) to the scheme (2.9) which depends on both the mesh function h and the finite element residual. The dependence of the shock–capturing term on the residual of the underlying conservation law ensures the consistency of the resulting finite element method for smooth analytical solutions u. Furthermore, we note that this shock–capturing term is isotropic; i.e. it includes both diffusion in the direction of the streamlines, as well as cross–wind diffusion.

Accordingly, we consider the following discontinuous Galerkin finite element discretisation of (2.2): find $u_h \in V_h^p$ such that

$$a(u_h, v_h) = 0 \quad \forall v_h \in V_h^p, \tag{2.35}$$

where $a(\cdot,\cdot)$ denotes a semi-linear form that is nonlinear in its first argument and linear in its second and that is defined by

$$a(u_h, v_h) = \sum_{K \in T_h} \left\{ -\int_K F(u_h) \cdot \nabla v_h \, dx + \int_{\partial K} H(u_h^+, u_h^-, n)\, v_h^+ \, ds \right. \\ \left. + \int_K \varepsilon \nabla u_h \cdot \nabla v_h \, dx \right\}. \tag{2.36}$$

Here, ε denotes the artificial viscosity defined by

$$\varepsilon = C_\varepsilon \, h^{2-\beta} \, |\nabla \cdot F(u_h)|, \tag{2.37}$$

where C_ε and $0 < \beta < 1/2$ are positive constants. The definition of ε in (2.37) represents a slight modification of the artificial viscosity model introduced and

analysed by Jaffre *et al.* [35]; here, we have simplified the definition of ε to only include the *interior* residual term $|\nabla \cdot F(u_h)|$. We note that in [35], the discontinuous Galerkin finite element method (2.35), with the addition of streamline–diffusion stabilisation and the numerical flux function $H(\cdot,\cdot,\cdot)$ defined as the (global) Lax–Friedrichs flux, was shown to be convergent for any approximating polynomial of degree $p \geq 0$ on general meshes for multidimensional scalar problems, whereas its solutions were shown to converge to the 'entropy solution' for multidimensional scalar problems.

The addition of the artificial viscosity term with the constant ε introduces two parameters into the resulting discontinuous Galerkin scheme; namely, C_ε and β. An automatic procedure for determining the correct level of artificial viscosity to be added into the scheme would obviously be advantageous. We note that in the context of streamline–diffusion stabilisation, work has been conducted in [10, 22, 29], for example, to automatically determine the optimal size of the streamline–diffusion parameter, thereby rendering the method free of any numerical tuning. Extensions of these techniques to yield parameter–free shock capturing terms are possible; however, this is beyond the scope of this work, and will not be discussed in more detail.

In the following, we demonstrate the effect of the proposed shock–capturing method. To this end, we consider the supersonic flow past a wedge, see Section B.4. The solution to this problem develops an oblique shock originating at the corner. In Figure 2.9, the shock at the outflow boundary is shown in detail for the numerical solution without shock–capturing, i.e. for $C_\varepsilon = 0$, and for small values, $C_\varepsilon = 0.02$ and $C_\varepsilon = 0.04$ with $\beta = 0.1$. We note that in all cases in Figure 2.9 the shock is resolved by 3-4 elements. Hence, for sufficiently small C_ε the shock is not smeared but it is resolved by at most as many elements as for the case when no shock–capturing is employed.

We end this section by noting that after the definition of the semi-linear form a and the shock–capturing terms in (2.36), the discontinuous Galerkin finite element discretisation used in this work for solving hyperbolic problems is now completely defined. As discussed above, the shock–capturing terms include residual terms and, as a consequence, are consistent. This is an essential ingredient to deduce that if the analytical solution u to (2.2) is sufficiently regular (i.e. the semi-linear form a in (2.36) can be evaluated for the exact solution u), then the following consistency condition holds:

$$a(u, v_h) = 0 \quad \forall v_h \in V_h^p. \tag{2.38}$$

This can be verified by inserting the exact solution u into (2.35) and using both, the dependence of the shock-capturing term on the residual and the consistency condition of the numerical flux function $H(\cdot,\cdot,\cdot)$, cf. condition (i) in Section 2.2. Thereby the *Galerkin orthogonality property* of the finite element method (2.35) is expressed as follows:

$$a(u, v_h) - a(u_h, v_h) = 0 \quad \forall v_h \in V_h^p. \tag{2.39}$$

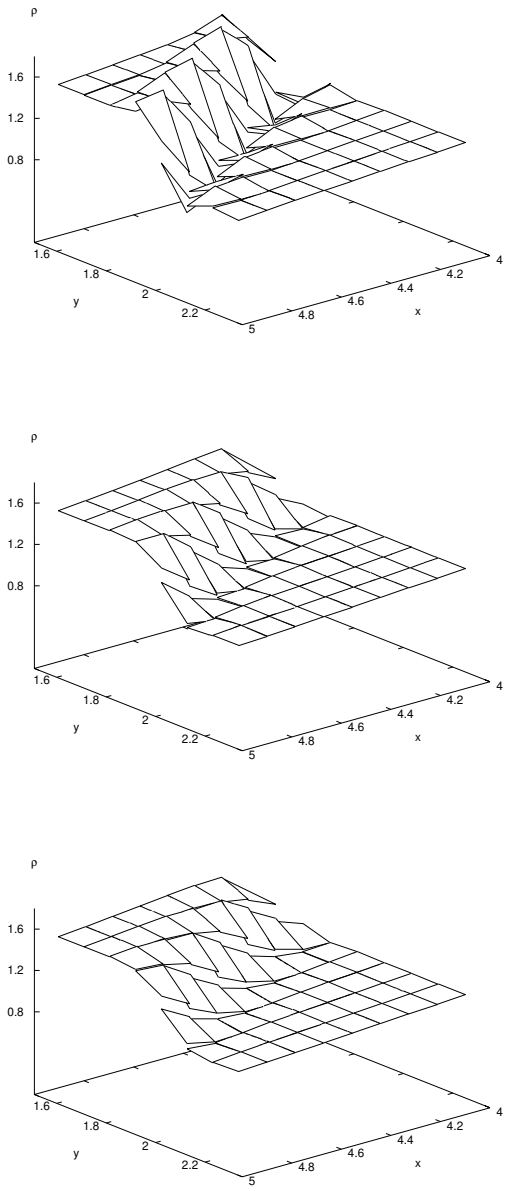
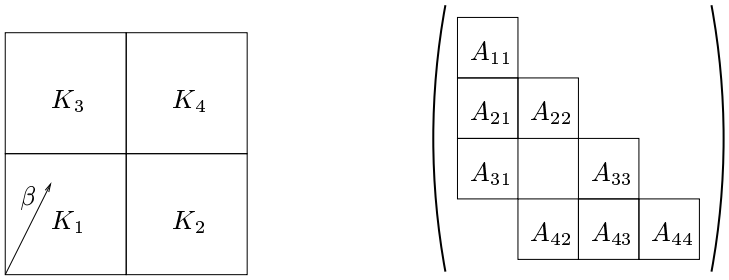


Figure 2.9: Compression corner problem: Shocks for $C_\varepsilon = 0$, $C_\varepsilon = 0.02$ and $C_\varepsilon = 0.04$.



(a) (b)

Figure 2.10: (b) Matrix structure resulting from the DG discretisation of a linear hyperbolic equation on a mesh (a) with 4 elements, ordered downstream.

2.8 Solving the discrete problems

For the case of a nonlinear problem (2.2) the discretised problem (2.35) is nonlinear and may be solved by a Newton iteration method or by a pseudo time iteration, for example. Both will be described in Section 2.8.3. The linear discrete problems, either arising from discretisation of linear problems (2.2), or occurring in each of the nonlinear iteration steps, are solved by GMRES iteration or simply by a Richardson iteration, both with block-Gauss-Seidel or block-SSOR preconditioning. This will be discussed and numerically tested in the following two subsections 2.8.1 and 2.8.2.

2.8.1 Solving the linear equations

In the case of scalar linear hyperbolic equations and an ordering of the elements from upstream to downstream, the system matrix becomes a lower block-diagonal matrix. The block structure of the matrix originates from the discontinuity of the test functions of the discontinuous Galerkin scheme (2.9) that allows for subdividing the global problem in several local problems as described in the following: By choosing $v_h|_{K_i} \equiv \phi_{i,k}$ for a fixed i and $v_h|_{K_j} \equiv 0$ for all $j \neq i$, equation (2.35) reduces to the local problem

$$-\int_{K_i} F(u_h) \cdot \nabla \phi_{i,k}\, dx + \int_{\partial K_i} H(u_h^+, u_h^-, n)\, \phi_{i,k}^+\, ds + \int_{K_i} \varepsilon \nabla u_h \cdot \nabla \phi_{i,k}\, dx = 0. \tag{2.40}$$

Here, $\phi_{i,k}$, $k = 1, \ldots, l$, represent the basis functions of the discrete space V_h^p with $\mathrm{supp}(\phi_{i,k}) \subset K_i$, and $l := m(p+1)^2$ denoting the number of degrees of freedom per element of discrete functions $v_h \in V_h^p$. As an example, Figures 2.10(a) and 2.10(b) show a simple mesh and the corresponding system matrix. The mesh consists of 4 elements K_i, $i = 1, \ldots, 4$, that are ordered in direction of vector β. The matrix consists of blocks each of the size $l \times l$, where blocks without a label denote zero blocks. A diagonal block A_{ii}, $i = 1, \ldots, 4$,

includes the interior terms of the element K_i; and a sub-diagonal block A_{ij} originates from the element boundary term including u_h^+ in element K_i and u_h^- in the neighbouring element K_j. This system can simply be solved by a single block-Gauss-Seidel step. This corresponds to the fact that for linear hyperbolic equations the local problems (2.40) on all elements can be solved successively in an upstream-downstream ordering of the elements.

When the equation is nonlinear a lower block diagonal matrix can in general not be attained through any ordering of the elements. Rather, there are some blocks above the block-diagonal of the matrix, even though possibly much less than below the diagonal. Numerical experiments show that for this case several block Gauss-Seidel steps within a defect correction algorithm must be employed for solving the system. This is equivalent to employing Richardson iteration in combination with a block-Gauss-Seidel preconditioning. In the general case of *nonlinear systems* like the compressible Euler equations, we have to distinguish between supersonic and subsonic flows. While for supersonic flows the information is transported only in direction of the flow field, making the block-Gauss-Seidel an appropriate preconditioner, for subsonic flows some information is transported upstream of the flow field. Here a block-SSOR preconditioner is employed that accommodates the flows in down- *and* up-stream direction.

In this context, we note that in [19] the convergence of block iterative methods for linear systems arising in the numerical solution of the Euler equations has been proven.

2.8.2 Numerical tests for solving the linear problems

In this section, we numerically test the performance of three different solvers in combination with several different preconditioners. They are tested on linear problems that typically arise from a nonlinear Newton iteration step which is employed for solving the stationary 2D Euler equations. We consider the solvers GMRES, BiCGstab and the Richardson iteration. Here, by Richardson iteration we denote a defect correction iteration, one of the simplest iterative solvers. We combine solvers with standard preconditioners, like Jacobi, SOR (Gauss-Seidel) and SSOR preconditioners as well as block–based preconditioners, block-Jacobi, block-SOR(Gauss-Seidel) and block-SSOR, that use the block structure of the matrices which arise from discontinuous Galerkin discretisations. To obtain a reasonably good approximation of the Newton direction, it is not necessary to reduce the linear residual of a nonlinear Newton step to machine accuracy, but it is sufficient to reduce the linear residual only by a specific factor, say 10^{-3}. For this reason, the following tests measure the number of steps and the time needed to reduce the linear residual by this given factor, only.

As indicated in the last section, we distinguish between subsonic and supersonic flows. The first test suite is based on a transonic and shocked flow through a nozzle. The flow is mainly subsonic; it accelerates from sub- to supersonic flow while passing the throat of the nozzle and it is shocked back to subsonic flow in the diverging part of the nozzle. This problem is solved using the DG(1) method on a globally refined mesh with 5120 degrees of freedom. In Table 2.7

	GMRES		Richardson		BiCGstab	
	iter's	time	iter's	time	iter's	time
b-ssor	59	8.2s	106	11.4s	35	9.2s
b-sor	102	12.0s	183	13.1s	87	15.8s
b-jacobi	205	28.1s	360	15.3s	165	19.0s
ssor	152	24.3s	solver failed		solver failed	
sor	235	39.9s	solver failed		solver failed	
jacobi	324	60.8s	solver failed		solver failed	

Table 2.7: Nozzle problem (transonic): numbers of iterations and time for GMRES, Richardson and BiCGstab solver for different preconditioners.

	GMRES		Richardson		BiCGstab	
	iter's	time	iter's	time	iter's	time
b-ssor	38	17.6s	54	21.1s	26	23.2s
b-sor	45	15.2s	73	18.3s	57	34.3s
b-jacobi	137	48.7s	206	28.6s	solver failed	

Table 2.8: Shock-reflection problem (supersonic): numbers of iterations and time for GMRES, Richardson and BiCGstab solver for different preconditioners.

we collect the numbers of iterations and the time used by each of the linear solvers in combination with each of the preconditioners to reduce the linear residual by the given factor. Here, "solver failed" means that the linear residual *in*creases through the linear iteration of the solver and that after several steps the linear residual exceeded a reasonably high value indicating that the solver will not converge. We see that only the GMRES solver is stable enough to solve the linear problem in conjunction with *non*-block-based preconditioners.

The second test suite is based on the so called "Shock-Reflection" problem, see [39], a purely supersonic problem, that involves a shock that is reflected from a wall. Table 2.8 shows the respective numbers of iterations and time measurements. We omit the rows that include non-block-based preconditioners because each of the linear solvers in combination with these preconditioners failed to solve the problem. First, we see that in each of the tests block-Jacobi preconditioning is the slowest of all block-based preconditionings. Furthermore, we see, by looking at the solvers GMRES and Richardson, that for the transonic problem, see Table 2.7, the solvers are faster in combination with the Block-SSOR preconditioner than with the Block-SOR(Gauss-Seidel) preconditioner. In contrast to that, for the supersonic problem, see Table 2.8, the situation is vice versa, as solving with Block-SOR turns out to be faster than with Block-SSOR preconditioning. This coincides with the consideration that for supersonic problems a "one-directional" preconditioning should be sufficient whereas for subsonic problems information is transported down- *and* upstream, wherefore a

		GMRES		Richardson	
# el.	# DoFs	iter's	time	iter's	time
20	320	15	0.1s	29	0.2s
80	1280	35	0.9s	60	1.2s
320	5120	59	8.2s	106	11.4s
1280	20480	136	102.2s	273	122.5s
5120	81920	180	630.4s	355	654.1s

Table 2.9: Nozzle problem (transonic): numbers of iterations and time for GMRES and Richardson with block-SSOR preconditioning under global refinement.

		GMRES		Richardson	
# el.	# DoFs	iter's	time	iter's	time
16	256	14	0.1s	21	0.1s
64	1024	19	0.3s	29	0.4s
256	4096	29	1.3s	38	2.4s
1024	16384	45	15.2s	72	17.6s
4096	65536	84	119.0s	solver failed	

Table 2.10: Shock-reflection problem (supersonic): numbers of iterations and time for GMRES and Richardson solver with block-Gauss-Seidel preconditioning under global refinement.

symmetric "both-directional" preconditioning, as provided by the Block-SSOR preconditioner, is the most adequate one.

Looking at the results of the BiCGstab solver we notice that the BiCGstab solver is always slower than either the GMRES or the Richardson solver. The BiCGstab solver turns out to be – to some extent – competitive with the other solvers for the case of subsonic problems and symmetric preconditioning, only, but it is slower by far in cases of transonic, hence "purely" unsymmetric, problems and unsymmetric preconditionings.

Finally, we numerically test on globally refined meshes the work (number of iterations and time) required by the Richardson iteration and the GMRES solver for solving the linear systems. We first consider the 'transonic flow through the nozzle' problem. Here, we employ the solver in combination with the block-SSOR preconditioner. The results of the numerical tests are shown in Table 2.9. Analogously, we solve the supersonic shock reflection problem, where here the solvers are preconditioned by the block-Gauss-Seidel preconditioner. The corresponding results are shown in Table 2.10.

We end this section by noting that it could certainly be advantageous to use multigrid methods for solving these linear problem, especially for the case of subsonic and transonic flows. Unfortunately, multigrid methods have not yet been fully implemented in the finite element library the code of this work was

based on. Therefore, we can not give comparison of the above-tested solvers with multigrid methods.

2.8.3 Solving the nonlinear problems

There are several different ways of solving the nonlinear (stationary) equations. The most common method is based on considering the solution of the stationary problem (2.2) as the stationary limit of the unstationary problem

$$\partial_t u + \nabla \cdot F(u) = 0. \tag{2.41}$$

This is then solved by a simple explicit time stepping scheme whereby this iteration in time is performed until the stationary solution is reached. Similarly, an implicit time stepping scheme can be employed. A different approach is to solve the stationary equation (2.2) directly by employing a nonlinear iteration scheme, like a *Newton iteration*. All these methods will be introduced in the following.

Explicit time discretisations

For the time discretisation of (2.41) we first introduce some notation. By $0 < t_0 < t_1 < \ldots < t_N = T$ we define a partition of the time interval $[0, T]$ with time step lengths $k_n = t_n - t_{n-1}$, $n = 1, \ldots, N$ and by $u_h^n \in V_h^p$ we denote a discrete function at the time t_n. Finally, we write $R(\cdot, \cdot)$ for the nonlinear residual of equation (2.35) defined by

$$R(u, v) = -a(u, v). \tag{2.42}$$

Using this notation, the explicit Euler scheme is given by: for $0 \leq n < N$ find $u_h^{n+1} \in V_h^p$ such that

$$(u_h^{n+1}, v_h) = (u_h^n, v_h) + k_n R(u_h^n, v_h) \qquad \forall v_h \in V_h^p, \tag{2.43}$$

and the second order Runge-Kutta scheme by: for $0 < n \leq N$ find $u_h^{n+1} \in V_h^p$ such that

$$\begin{aligned} (u_h^{n+1}, v_h) &= \tfrac{1}{2} \left\{ (u_h^n, v_h) + (\tilde{u}_h^n, v_h) + k_n R(\tilde{u}_h^n, v_h) \right\}, \\ (\tilde{u}_h^n, v_h) &= (u_h^n, v_h) + k_n R(u_h^n, v_h) \qquad \forall v_h \in V_h^p. \end{aligned}$$

For higher order Runge-Kutta time discretisations we refer to [14], for example. We note that for a DG space–discretisation with polynomials of degree p we use a $(p+1)$-th order accurate Runge Kutta explicit time discretisation, see Cockburn and Shu [17].

Newton iteration

The nonlinear *Newton iteration* generates a sequence of iteratives u_h^k by the following method. Given an iterative u_h^k, the update d^k of u_h^k to get to the next iterative

$$u_h^{k+1} = u_h^k + \omega^k d^k$$

no. of	Newton		Explicit Euler		Runge Kutta 2	
el.	steps	sec	steps	sec	steps	sec
20	6	1.6	113	3.9	56	3.6
35	5	2.6	163	10.2	85	10.0
71	5	5.2	265	32.9	136	32.9
137	5	10.5	451	107.2	230	107.0
257	6	24.5	811	358.1	409	356.3

Table 2.11: Comparison of Newton iteration with time iteration schemes (explicit Euler, 2nd order Runge Kutta) for solving the nonlinear equations.

is given by the following problem: find $d_h^k \in V_h^p$ such that

$$a'[u_h^k](d_h^k, v_h) = R(u_h^k, v_h), \quad \forall v_h \in V_h^p. \tag{2.44}$$

Here, w^k denotes a damping parameter and $a'[w](\cdot, v)$ denotes the Fréchet derivative of $u \to a(u, v)$, for $v \in V$ fixed, at some w in V, where V is some suitable chosen function space such that $V_h^p \in V$. We remark that the linearisation $a'[w](\cdot, v)$ of $a(\cdot, v)$ in equation (2.44) is only a *formal* notation, in the sense that $a'[w](\cdot, v)$ may not in general exist. Instead, a suitable approximation to $a'[w](\cdot, v)$ in the direction d_h^k must be determined, for example, by computing appropriate finite difference quotients of $a(\cdot, v)$. For more details about possible approximations to $a'[w](\cdot, v)$, see Section 2.8.4.

Comparison of Newton iteration and explicite time iteration schemes

Here, we numerically test the performance of the Newton iteration for solving nonlinear equations in comparison to explicit time iteration schemes, such as the explicit Euler scheme and the second order Runge Kutta scheme. This test is performed on the 'supersonic flow around a wedge' problem, see Section B.3 or [11], under successive adaptive refinement. After the nonlinear problem is solved up to a nonlinear residual of 10^{-6} in the l_2-norm, the mesh is refined and the numerical solution is interpolated to the refined mesh. This transfered solution serves as start solution of the nonlinear iteration on the refined mesh. All adaptive refinement takes place at the position of the shock whereby the minimal mesh size, $h_{\min}$, is halved in each of the refinement steps. Numerical tests for the test problem considered here, show that to sustain numerical stability of the explicit Euler scheme, the time step size k must satisfy $\frac{k}{h_{\min}} \leq 0.05$, approximately, whereas for the Runge-Kutta scheme this CFL condition can be weakened to $\frac{k}{h_{\min}} \leq 0.1$. Hence, for this test suite the time step size of the explicit Euler scheme needs to be chosen about half of the time step size of the Runge-Kutta scheme. Therefore, approximately twice as many time steps are needed for the explicit Euler scheme than for the Runge-Kutta scheme in order to reduce the nonlinear residual below a prescribed final residual. However, given that each Runge Kutta time step involves two linear systems to be solved,

in comparison to one in each Euler step, both schemes turn out to require approximately the same amount of time. This behaviour is confirmed by Table 2.11 that shows the number of elements of the successively adaptive refined meshes, the number of iteration steps and the required CPU time in seconds on each refinement level, for the Newton iteration, the explicit Euler and the second order Runge Kutta scheme. We see that the numbers of time steps for the time iteration schemes significantly increase (they almost double) on each refinement level, whereas for the Newton iteration the number of iterations stays constant. This results in a large difference in the times required by the iteration schemes under comparison: it can clearly be seen in Table 2.11 that the Newton iteration requires much less time than the time iteration schemes. Finally, we note that the CFL conditions on the time step size given above must be satisfied even when the nonlinear residual is small. Indeed, numerical tests show that the time step size can *not* be increased even when the numerical solution is already close to the stationary solution limit.

Implicit time discretisation

The implicit Euler scheme is given by: for $0 < n \leq N$ find $u_h^{n+1} \in V_h^p$ such that

$$(u_h^{n+1}, v_h) + k_n a(u_h^{n+1}, v_h) = (u_h^n, v_h) \qquad \forall v_h \in V_h^p.$$

For a nonlinear equation (2.41) this time discretisation involves a nonlinear problem in each time step, that in turn can be solved by Newton iteration: for $0 < n \leq N$ and given an iterative $u_h^{n+1,k}$, the update d^k of $u_h^{n,k}$ to get to the next iterative

$$u_h^{n+1,k+1} = u_h^{n+1,k} + \omega^k d^k$$

is given by the following problem: find $d_h^k \in V_h^p$ such that

$$(d_h^k, v_h) + k_n a'[u_h^{n+1,k}](d_h^k, v_h) = (u_h^n, v_h) - (u_h^{n+1,k}, v_h) + k_n R(u_h^{n+1,k}, v_h)$$

holds for all $v_h \in V_h^p$. This method is very time consuming. A numerical test comparing the implicit and the explicit Euler schemes with same time step sizes employed and each nonlinear problem of the implicit Euler scheme solved up to a nonlinear residual of 10^{-6} requires 86.8 seconds versus 3.9 seconds, see first row of Table 2.11, on the coarsest mesh. But, the implicit Euler scheme does not need to satisfy a CFL condition like the explicit Euler scheme. This allows arbitrarily large time steps, whereas in the limit the implicit Euler scheme reduces to the Newton iteration (2.44).

2.8.4 Approximations to the Jacobian matrix of the scheme

As discussed in the last subsection, we employ Newton's iteration for solving the nonlinear problems. For that, we introduced the Fréchet derivative $a'[w](u, v)$ of $u \to a(u, v)$ that is needed in each Newton's iteration step defined in (2.44).

We note that exactly the same "Jacobian matrix" $a'[w](u,v)$ will occur again in the context of the dual problem in Chapter 3. The explicit form of this Jacobian matrix will be described in the following.

To that end, we first recall definition (2.36) of the semi-linear form $a(\cdot,\cdot)$ and write it as follows

$$a(u,v) = \sum_{K\in T_h} \left\{ -\int_K F(u)\cdot\nabla v\,dx + \int_{\partial K\setminus\Gamma} H(u^+,u^-,n)\,v_h^+\,ds \right\} + a_\Gamma(u,v) + a_{\rm sc}(u,v), \quad (2.45)$$

where $a_\Gamma(u,v)$ was defined in (2.31) and includes the boundary face terms,

$$a_\Gamma(u,v) = \sum_{K\in T_h} \int_{\partial K\cap\Gamma} H(u^+,u^-,n)\,v^+\,ds,$$

where here $u^-|_\Gamma$ denotes an appropriate boundary function as described in Section 2.6. Furthermore $a_{\rm sc}(u,v)$ includes the shock–capturing terms,

$$a_{\rm sc}(u,v) = \sum_{K\in T_h} \int_K \varepsilon\nabla u\cdot\nabla v\,dx, \quad (2.46)$$

where $\varepsilon\equiv\varepsilon(u) = C_\varepsilon\,h^{2-\beta}\,|\nabla\cdot F(u)|$ according to the definition in (2.37).

In general, the numerical fluxes $H(u^+,u^-,n)$ are not differentiable with respect to u^+ and u^-. As a consequence, $a(u,v)$ is in general not differentiable with respect to u. Therefore, $a'[w](u,v)$ must be replaced by an appropriate approximation that may be written in the general form

$$\tilde a'[w](u,v) = \sum_{K\in T_h} \left\{ -\int_K (F'(w)u)\cdot\nabla v\,dx + \int_{\partial K\setminus\Gamma} \left(\tilde H'_{u^+}(w^+,w^-,n)u^+ + \tilde H'_{u^-}(w^+,w^-,n)u^-\right) v\,ds \right\} + \tilde a'_\Gamma[w](u,v) + \tilde a'_{\rm sc}[w](u,v),$$

where $w\to\tilde H'_{u^+}(w^+,w^-,n)$ and $w\to\tilde H'_{u^-}(w^+,w^-,n)$ denote approximations to the derivatives of H with respect to its first and second arguments, respectively, and $\tilde a'_\Gamma[w](u,v)$ and $\tilde a'_{\rm sc}[w](u,v)$ represent approximations to the derivatives of $u\to a_\Gamma(u,v)$ and $u\to a_{\rm sc}(u,v)$, respectively; see the following two subsections for more details. Furthermore, we give examples of $\tilde H'_{u^+}$ and $\tilde H'_{u^-}$ for different fluxes at the end of this section.

Jacobian matrix of the shock-capturing term

$\tilde{a}'_{\mathrm{sc}}[w](u,v)$ represents an approximation to the derivative of the shock-capturing term (2.46) and is given by

$$\tilde{a}'_{\mathrm{sc}}[w](u,v) = \sum_{K\in T_h} \int_K \left(\varepsilon(w)\nabla u + \tilde{\varepsilon}'[w](u)\nabla w\right)\cdot\nabla v\,dx,$$

where $\tilde{\varepsilon}'[w](u)$ denotes the following approximation to the derivative of $u \to \varepsilon(u)$, cf. (2.37),

$$\tilde{\varepsilon}'[w](u) = C_\varepsilon\, h^{2-\beta}\mathrm{sgn}(\varepsilon(w))\left(F'(w)\cdot\nabla u + F''(w)\cdot wu\right) I.$$

Jacobian matrix of boundary face terms

According to the definition (2.32) of $a_\Gamma(u,v)$, its approximate derivative $\tilde{a}'_\Gamma[w](u,v)$ consists of several parts

$$\tilde{a}'_\Gamma[w](u,v) = \tilde{a}'_{\Gamma_{\mathrm{i}}}[w](u,v) + \tilde{a}'_{\Gamma_{\mathrm{o,sup}}}[w](u,v) + \tilde{a}'_{\Gamma_{\mathrm{o,sub}}}[w](u,v) + \tilde{a}'_{\Gamma_{\mathrm{refl}}}[w](u,v). \tag{2.47}$$

Employing the same approximations $\tilde{H}'_{u^+}$ and $\tilde{H}'_{u^-}$ on boundary faces as on inner faces, the terms on the right hand side of (2.47) are given by:

- On a *inflow boundary* Γ_{i}:

$$\tilde{a}'_{\Gamma_{\mathrm{i}}}[w](u,v) = \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{i}}} \tilde{H}'_{u^+}(w^+,g,n)u^+v^+\,ds.$$

- On a *supersonic outflow boundary* $\Gamma_{\mathrm{o,sup}}$

$$\tilde{a}'_{\Gamma_{\mathrm{o,sup}}}[w](u,v) = \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{o,sup}}} \Big(\tilde{H}'_{u^+}(w^+,w^+,n) + \tilde{H}'_{u^-}(w^+,w^+,n)\Big)\,u^+v^+\,ds. \tag{2.48}$$

- On a *subsonic outflow boundary* $\Gamma_{\mathrm{o,sub}}$

$$\tilde{a}'_{\Gamma_{\mathrm{o,sub}}}[w](u,v) = \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{o,sub}}} \Big(\tilde{H}'_{u^+}(w^+,u^-_{\mathrm{o,sub}}(w^+),n) + \tilde{H}'_{u^-}(w^+,u^-_{\mathrm{o,sub}}(w^+),n)u^{-}_{\mathrm{o,sub}}{}'(w^+)\Big)\,u^+v^+\,ds, \tag{2.49}$$

with $u^-_{\mathrm{o,sub}}(u)$ as defined in (2.33). Writing $u^-_{\mathrm{o,sub}}(u)$ in conservative variables $u = (u_1,u_2,u_3,u_4) = (\varrho,\varrho v_1,\varrho v_2,e)$ gives

$$u^-_{\mathrm{o,sub}}(u) = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \frac{p_{\mathrm{o,sub}}}{\gamma-1} + \frac{1}{2}(u_2^2+u_3^2)/u_1 \end{pmatrix}, \tag{2.50}$$

and consequently

$$u_{\mathrm{o,sub}}^{-\,\prime}(u) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2}(u_2^2+u_3^2)/u_1^2 & u_2/u_1 & u_3/u_1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2}v^2 & v_1 & v_2 & 0 \end{pmatrix}.$$

- On a *reflective boundary* Γ_{refl}

$$\tilde{a}'_{\Gamma_{\mathrm{refl}}}[w](u,v) = \sum_{K\in T_h} \int_{\partial K\cap\Gamma_{\mathrm{refl}}} \Big(\tilde{H}'_{u^+}(w^+, u^-_{\mathrm{refl}}(w^+), n) + \tilde{H}'_{u^-}(w^+, u^-_{\mathrm{refl}}(w^+), n) u^{-\,\prime}_{\mathrm{refl}}(w^+) \Big) u^+ v^+ \, ds,$$

with $u^-_{\mathrm{refl}}(u)$ and $u^{-\,\prime}_{\mathrm{refl}}(u)$ as defined in (2.34).

Approximative derivatives of numerical fluxes

The numerical fluxes $H(u^+, u^-, n)$ generally are not differentiable with respect to u^+ and u^-. In fact, for example the Vijayasundaram flux and the local Lax-Friedrichs flux, like many other fluxes, include non–differentiable terms as 'min'- and 'max'-functions, for example.

1. **Vijayasundaram flux.** In the case of the Vijayasundaram flux

$$H(u^+, u^-, n) = B^+(\bar{u}, n)u^+ + B^-(u^-, n)u^-,$$

as defined in (2.11), these non-differential terms are the matrices $B^-(\bar{u}, n)$ and $B^+(\bar{u}, n)$, that include the negative and positive parts of $B(\bar{u}, n) = F'(\bar{u}) \cdot n$, respectively. Therefore, we neglect the (non–existing) derivative of $B^-(\bar{u}, n)$ and approximate the (non–existing) derivative of $m(u) := B^-(\bar{u}, n)u$ by

$$\tilde{m}'[w](u) = B^-(\bar{w}, n)u. \tag{2.51}$$

For $B^+(\bar{u}, n)$ we employ an analogous approximation.

Applying the approximation (2.51) to the Vijayasundaram flux function (2.11) yields

$$\tilde{H}'_{u^+}(u^+, u^-, n) = B^+(\bar{u}, n) \quad \text{and} \quad \tilde{H}'_{u^-}(u^+, u^-, n) = B^-(\bar{u}, n). \tag{2.52}$$

Consequently, the boundary terms in (2.47) are given by

$$\begin{aligned}
\tilde{a}'_{\Gamma_{\mathrm{i}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{i}}} B^+(w^+,n)u^+v^+\,ds,\\
\tilde{a}'_{\Gamma_{\mathrm{o,sup}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{o,sup}}} B^+(w^+,n)u^+v^+\,ds,\\
\tilde{a}'_{\Gamma_{\mathrm{o,sub}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{o,sub}}} \Big(B^+(w^+,n)\\
&\qquad +B^-(w^+,n){u^-_{\mathrm{o,sub}}}'(w^+)\Big)\,u^+v^+\,ds,\\
\tilde{a}'_{\Gamma_{\mathrm{refl}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{refl}}} \Big(B^+(w^+,n)\\
&\qquad +B^-(w^+,n){u^-_{\mathrm{refl}}}'(w^+)\Big)\,u^+v^+\,ds.
\end{aligned}$$

2. **Lax-Friedrichs flux.** The local Lax-Friedrichs flux

$$H(u^+,u^-,n)=\tfrac{1}{2}\left(F(u^+)\cdot n+F(u^-)\cdot n+\alpha\left(u^+-u^-\right)\right),$$

as defined in (2.10), includes the constant α that is an estimate of the largest eigenvalue (in absolute value) of the matrix $B(u,n)$ evaluated at $u=u^+$ and $u=u^+$. Hence, α includes a 'max'- function and is thus not differentiable. Therefore, we neglect the derivatives of α and use the following approximate derivatives $\tilde{H}'_{u^+}$ and $\tilde{H}'_{u^-}$ of the numerical flux function H

$$\begin{aligned}
\tilde{H}'_{u^+}(u^+,u^-,n)&=\tfrac{1}{2}\left(F'(u^+)\cdot n+\alpha I\right)\quad\text{and}\\
\tilde{H}'_{u^-}(u^+,u^-,n)&=\tfrac{1}{2}\left(F'(u^-)\cdot n-\alpha I\right).
\end{aligned}$$

Consequently, the boundary terms (2.47) are given by

$$\begin{aligned}
\tilde{a}'_{\Gamma_{\mathrm{i}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{i}}} F'(w^+)\cdot n\,u^+v^+\,ds,\\
\tilde{a}'_{\Gamma_{\mathrm{o,sup}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{o,sup}}} F'(w^+)\cdot n\,u^+v^+\,ds,\\
\tilde{a}'_{\Gamma_{\mathrm{o,sub}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{o,sub}}} \tfrac{1}{2}\Big(F'(u^+)\cdot n+\alpha I\\
&\qquad +\left(F'(u^-_{\mathrm{o,sub}}(u^+))\cdot n-\alpha I\right){u^-_{\mathrm{o,sub}}}'(u^+)\Big)u^+v^+\,ds,\\
\tilde{a}'_{\Gamma_{\mathrm{refl}}}[w](u,v) &= \sum_{K\in T_h}\int_{\partial K\cap\Gamma_{\mathrm{refl}}} \tfrac{1}{2}\Big(F'(u^+)\cdot n+\alpha I\\
&\qquad +\left(F'(u^-_{\mathrm{refl}}(u^+))\cdot n-\alpha I\right){u^-_{\mathrm{refl}}}'(u^+)\Big)u^+v^+\,ds.
\end{aligned}$$

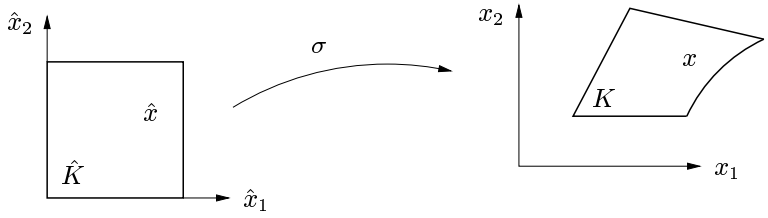


Figure 2.11: Mapping σ of reference element $\hat{K}$ to the element K in real space.

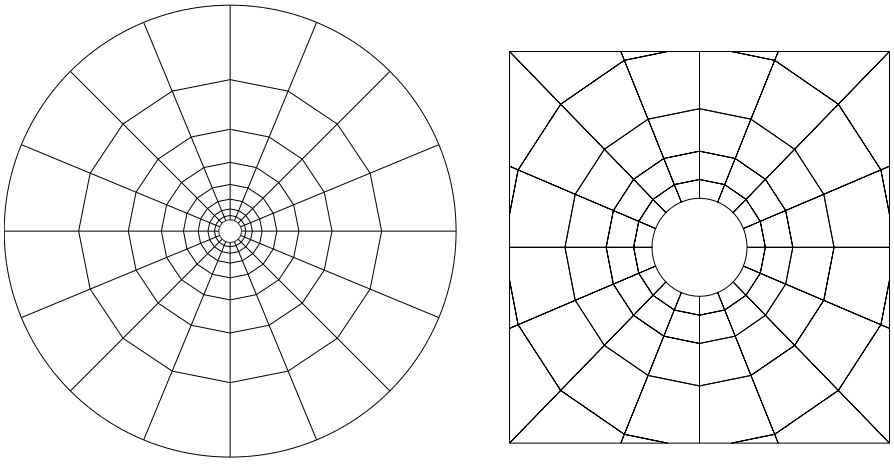
2.9 Higher order boundary approximation

In many applications the domain Ω is not a polygonal domain but includes curved boundaries. For these cases the boundary cannot be represented exactly by the triangulation T_h. Approximating the boundary by a piecewise linear boundary interpolation, i.e. by a polygonal boundary, may in some applications not be sufficient. Then, a higher order boundary approximation, for example by piecewise quadratic or cubic boundary interpolation, must be employed. This necessitates the use of *curved elements* at the boundary of the domain. In finite element methods it is common practice, see [49] for example, to deal with curved elements by employing higher order polynomial mappings σ_K of the reference element $\hat{K}$ to the real element K in real space, see Figure 2.11. This way a "Q_p boundary approximation", that denotes a piecewise polynomial interpolation of the boundary by polynomials of degree p, can be realised by a "Q_p mapping" $\sigma_K \in [Q_p]^d$ polynomial mapping of degree p. For a more detailed description of curved elements and mapping functions of higher polynomial degrees as well as for some implementational details we defer the reader to Appendix C.

In the following, we use quadrilateral elements with straight boundaries in the interior of the domain; these elements are given by the image of a Q_1 mapping of the reference element. Furthermore, we use curved elements at the boundary of the domain, where these curved elements are images of Q_p mappings of the reference element, for given polynomial degrees p stated below. A special case is the approximation of the domain Ω by a polygonal domain. This is equivalent to a Q_1 approximation of the boundary and is realized by employing a Q_1 mapping on *all* elements of the triangulation.

For the discontinuous Galerkin discretisation it is in some cases not sufficient to approximate curved boundaries by polygons. In the following subsection we recompute the "flow around a circle problem" considered by Bassi and Rebay [4]. This example clearly shows that when using discontinuous Galerkin discretisations it is essential to employ a higher order boundary approximation at reflective boundaries in order to avoid unphysical solutions.

In the subsection thereafter, we show by numerical tests that even the order of convergence of discontinuous Galerkin solutions is reduced when reflective boundaries are approximated by polynomials of lower order.



(a) (b)

Figure 2.12: (a) Coarse mesh of the "flow around a circle problem"; (b) Zoom of part $[-4,4]^2$ of the mesh, both for Q_2 boundary approximation.

2.9.1 Flow around a circle

Bassi and Rebay [4] encountered that a polygonal boundary approximation of *curved reflective* boundaries leads to inaccurate and even to unphysical numerical solutions. Furthermore, they showed that 'a higher order boundary approximation gives a dramatic improvement in the accuracy of the numerical approximation'.

In order to illustrate this effect, in the following we recompute the "flow around a circle problem" of the 2D compressible Euler equations, an example that was already considered in [4].

To this end, we consider a computational domain that is defined to be an annulus with inner and outer boundary consisting of circles of radius 1 and 20 units, respectively, see Figures 2.12(a) and 2.12(b) that show the coarse mesh of the computation domain and a zoom of part $[-4,4]^2$ of the mesh. On the farfield boundary we apply horizontal $M = 0.38$ flow that enters the computational domain from the left. The solution to this subsonic flow is known to be symmetric, see [4]. Figures 2.13 and 2.14 show the Mach isolines of DG(1) numerical solutions on four globally refined meshes (128, 512, 2048 and 8192 elements) with Q_1 and Q_2 boundary approximation, respectively. The isolines are plotted for values of the Mach number given by $M_j = 0.038j$ for $j = 0, 1, \ldots$.

The solutions on meshes with Q_1 boundary approximation, see Figures 2.13, are very inaccurate and definitely not symmetric because of a non-physical wake that develops behind the obstacle. The poor behaviour of this non-physical numerical solution does not disappear through refinement, even on very fine meshes a wake develops downstream the circle. Bassi and Rebay showed in [4] that there is a non-physical entropy production at the boundary of the circle

leading to a non-physical "boundary layer" that affects the solution even a distance from the boundary, as seen in the wake downstream of the obstacle.

The behaviour of the numerical solution changes completely when using meshes with Q_2 boundary approximation, see Figure 2.14. For this case, the numerical solution converges nicely to a solution which is symmetric in the upstream-downstream direction. Already after four refinement steps the solution can hardly be distinguished from a symmetric solution, see Figure 2.14 (d). We note that employing a Q_3 boundary approximation results in numerical solutions that cannot be distinguished from the numerical solutions in Figure 2.14. Finally, Figure 2.15 shows the DG(2) numerical solution with Q_2 boundary approximation. It is remarkable that already after one refinement step the solution is accurate and shows a upstream–downstream symmetry. This example shows that employing a higher order boundary approximation leads to a significant improvement in the accuracy of the numerical solution at the boundary as well as in the rest of the domain, especially behind an obstacle.

At the end of this section, we note that the DG(0) method, and hence the corresponding finite volume schemes, shows the same accuracy on meshes with boundary approximation of arbitrary degree. For example, Figures 2.16 and 2.17 show the DG(0) solution with Q_1 and Q_2 boundary approximation, respectively. This might explain why the effect of unphysical solutions as a result of low order boundary approximation has not been encountered earlier for example by the finite volume community.

2.9.2 Reduced order of convergence due to boundary approximation

The effect of low convergence of numerical solutions involving reflective boundaries and even the occurrence of non-physical numerical solutions, are due to a low order boundary approximation of the reflective boundaries. This was first encountered by Bassi and Rebay in [4]. They regarded the non-physical entropy production at the reflective boundary of the "flow around the circle" and the "Ringleb flow" problem and the convergence of the numerical solution in terms of this entropy production.

In this section we consider the convergence of numerical solutions to the "Ringleb flow" problem, see B.5, in terms of the L^2-error of the numerical solutions. In Section 2.4 we numerically showed that the convergence of numerical solutions to a slightly modified Ringleb flow problem is of the optimal order $O(h^{p+1})$. There, we modified the flow problem by imposing the boundary conditions, see (2.20), on the whole boundary of the domain, hence no reflective boundary conditions were active. In the following we want to investigate how the order of convergence is affected when reflective boundary conditions, see Section 2.6, *are* imposed, and how the order of convergence changes when the reflective boundaries are approximated by polynomials of different degrees.

Figure 2.18 shows the convergence of the L^2-error of the DG(1) numerical solution to the Ringleb flow problem when the reflective boundaries are approximated by polynomials of degrees one to four (Q_1-Q_4 boundary approximation).

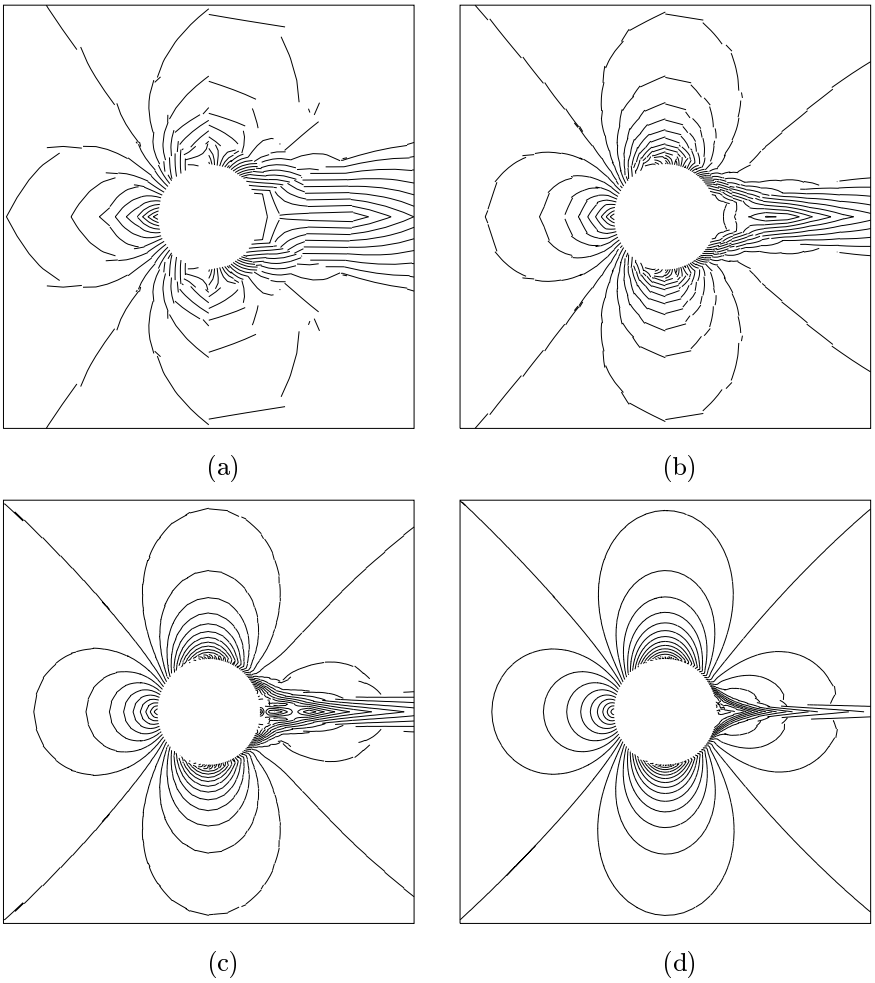


Figure 2.13: (a)-(d) DG(1) solutions with Q_1 boundary approximation on globally refined meshes of 128, 512, 2048 and 8192 elements.

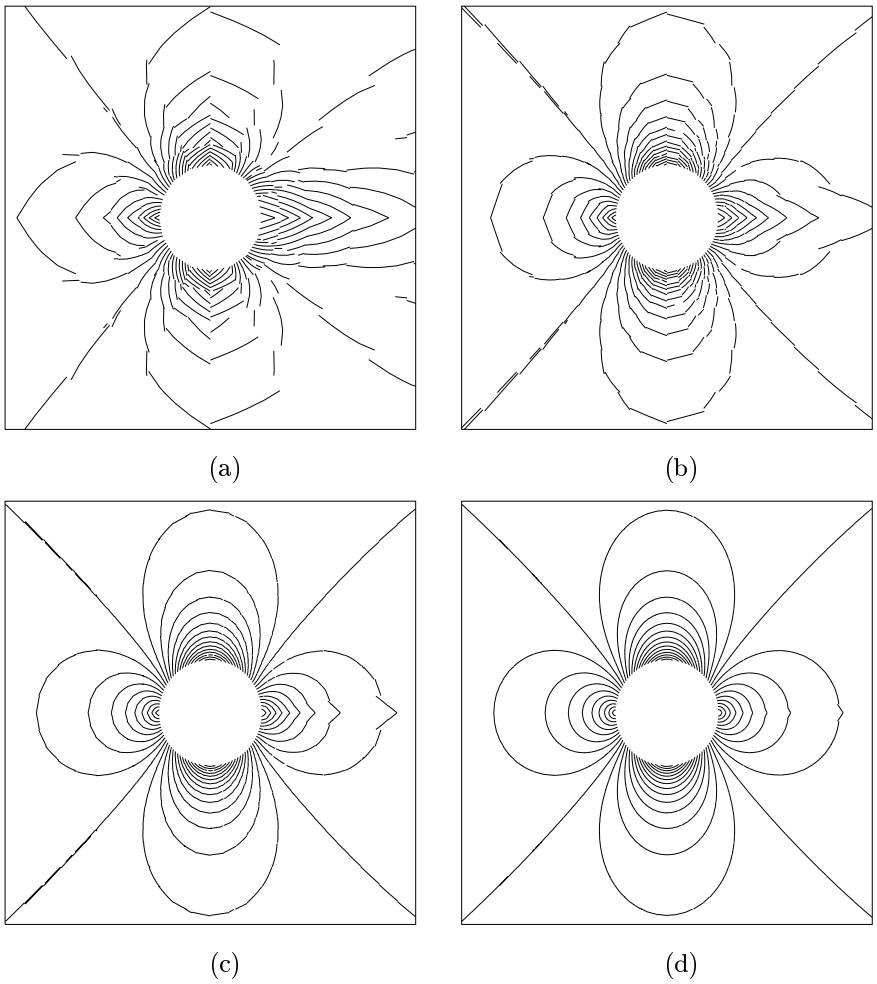
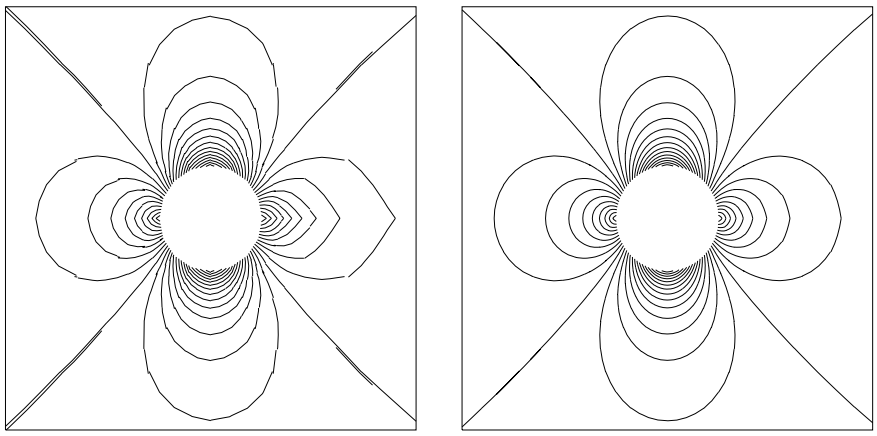


Figure 2.14: (a)-(d) DG(1) solutions with Q_2 boundary approximation on globally refined meshes of 128, 512, 2048 and 8192 elements.



(a) (b)

Figure 2.15: DG(2) solutions with Q_2 boundary approximation on globally refined meshes of (a) 128 and (b) 512 elements.

When the reflective boundaries are approximated by higher order polynomials (Q_2-Q_4 boundary), the order of convergence is exactly $O(h^2)$ and the accuracy of these higher order boundary methods coincide properly. In contrast to that, there is a big gap in accuracy when the reflective boundaries are approximated by piecewise linear functions (Q_1 boundary) compared to higher order boundary approximations. The reason for this dramatic difference in accuracy are the kinks in the reflective boundary when it is approximated by piecewise linear functions, see meshes in Figure 2.2, for example. Each of the kinks affects the flow leading to a non-physical entropy production, cf. [4], right at the boundary. This effect can be clearly seen in Figure 2.19. It shows the density ρ and the density isolines on the base of the plot of DG(1) solutions computed on meshes of, respectively, 32 and 128 elements with Q_1 boundary approximation. At the reflective boundaries the numerical solutions are staggered and crumbled. This leads to a bad approximation of the solution in the neighbourhood of the boundary. For comparison, in Figure 2.20 we show the corresponding solutions on meshes with Q_2 boundary approximation where the numerical solutions are smooth at the boundary. As seen in Figure 2.19 the wobbling behaviour of the numerical solution at the reflective boundary does not vanish while the mesh is refined. Indeed, even on very fine meshes the numerical solution does not change this behaviour; see a zoom of a part of the numerical solutions in Figure 2.21 computed on meshes of 2048 elements with Q_1 and Q_2 boundary approximation, respectively. We note that the solution in Figure 2.21, left, is staggered even several layers of elements away from the boundary. This indicates that the numerical solution is affected from a low order boundary approximation not only in the neighbourhood of this boundary but also in the rest of the domain.

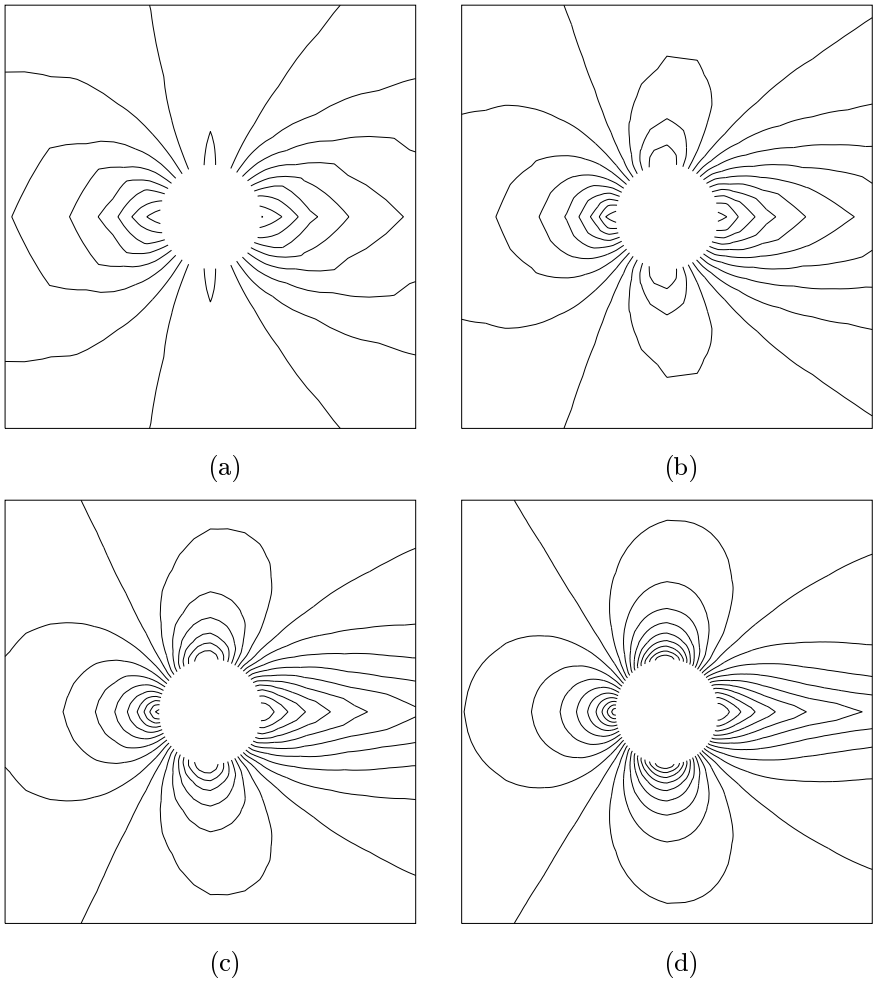


Figure 2.16: (a)-(d) DG(0) solutions with Q_1 boundary approximation on globally refined meshes of 128, 512, 2048 and 8192 elements.

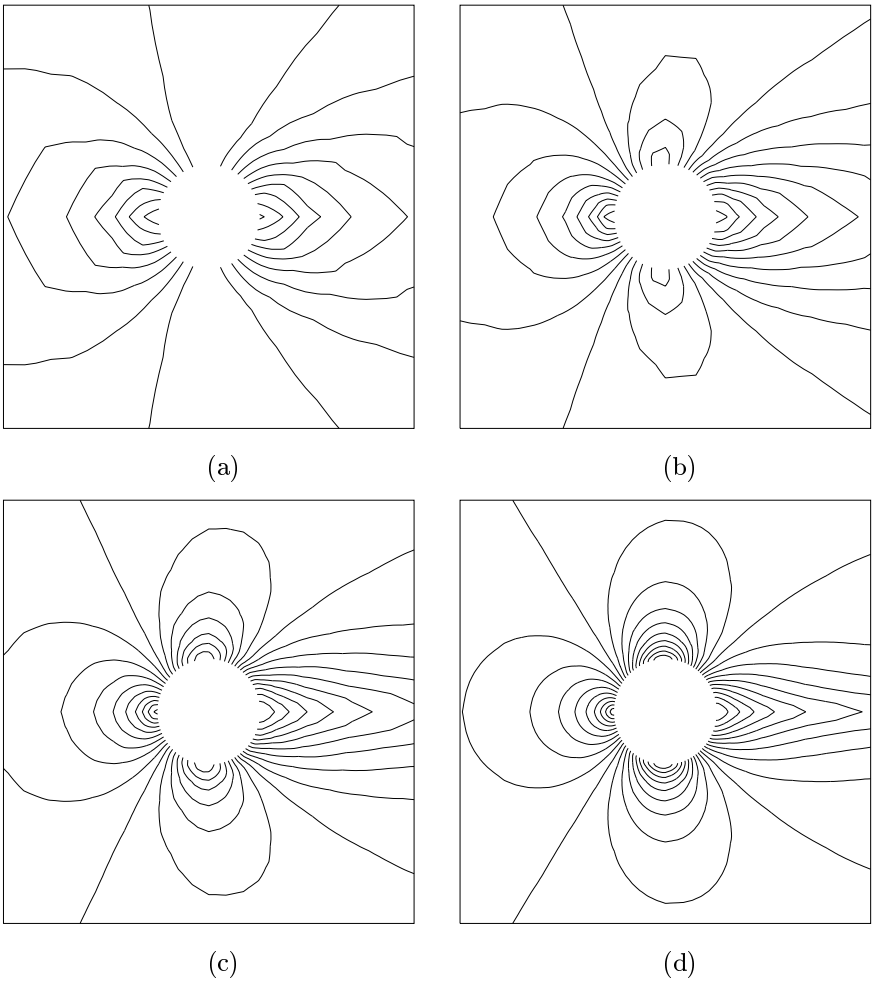


Figure 2.17: (a)-(d) DG(0) solutions with Q_2 boundary approximation on globally refined meshes of 128, 512, 2048 and 8192 elements.

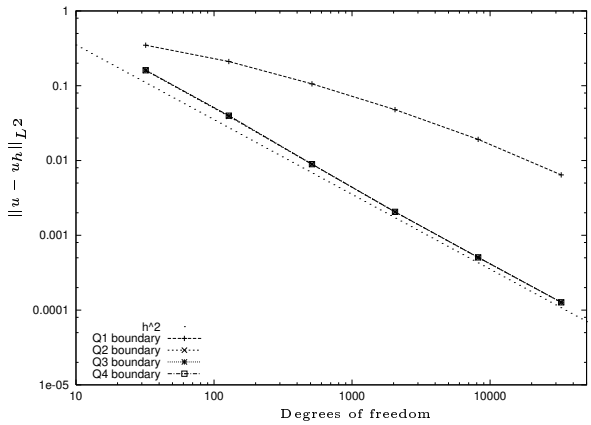


Figure 2.18: Ringleb flow problem: L^2-error of the DG(1) solution for Q_1-Q_3 boundary approximations.

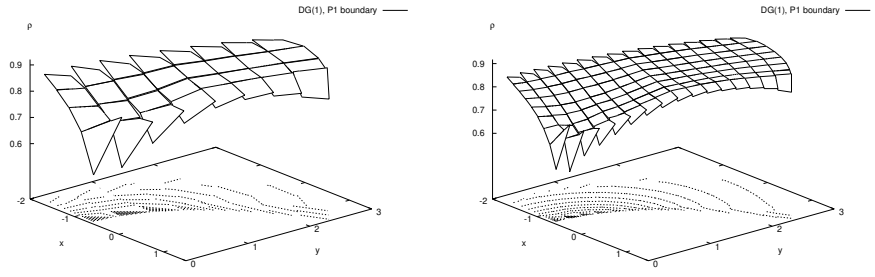


Figure 2.19: Ringleb flow problem: DG(1) for Q_1 boundary approximation.

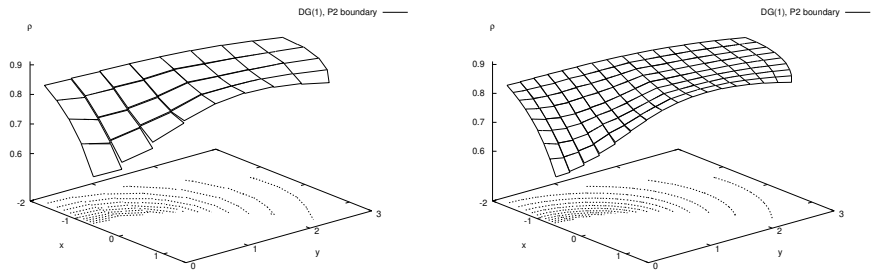


Figure 2.20: Ringleb flow problem: DG(1) for Q_2 boundary approximation.

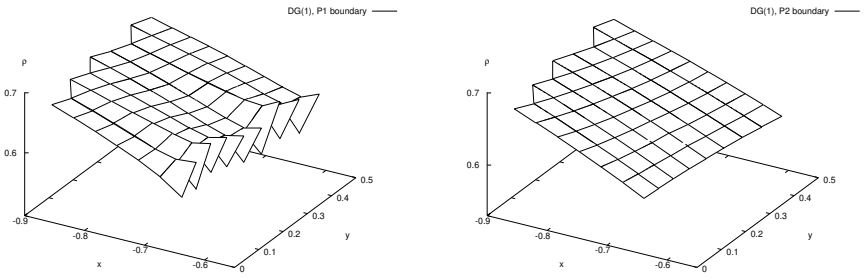


Figure 2.21: Ringleb flow problem: part of DG(1) solution on mesh with Q_1 (left) and Q_2 (right) boundary approximation.

This, indeed, is confirmed by the "flow around a circle" example problem where an unphysical wake develops downstream of the obstacle.

For summarising the results of this section we state that, in fact, the degree of polynomials employed for approximating reflective boundaries does affect the order of convergence of the discontinuous Galerkin method. Using a piecewise linear approximation of the boundary leads to a very inaccurate and unstable solution in the neighbourhood of the boundary and in some cases even unphysical numerical solutions develop. This effect of unphysical solutions as a result of low order boundary approximation for discontinuous Galerkin methods was first reported in Bassi and Rebay, cf. [4]. Here, we showed that for the DG(1) method a super–parametric element using at least Q_2 boundary approximations must be employed to avoid this effect and to ensure an optimal order of convergence in the L^2-norm.

2.10 Mesh generation for airfoil computations

This section is devoted to the generation of meshes. Here, we consider meshes particularly designed for flow computations around obstacles like airfoils. As powerful mesh generators for quadrilateral meshes were not available to the author, in the following the airfoil meshes will be constructed by hand, see [21] for several different ways of mesh generation. These meshes are basically given by an annulus where the inner boundary represents the geometry of the airfoil and the outer boundary is given by a circle. To allow for free flow boundary values applied on the whole outer boundary this circle should be relatively large in comparison to the chord length of the airfoil. Given an airfoil of unit chord length, here we choose the radius of the outer boundary circle to be 10, for example. Figure 2.12(a) shows such an annular mesh with a small circular obstacle in the center.We see that this mesh consists of several layers of elements that encircle the small obstacle. These layers of elements are necessary in order

Figure 2.22: Profile of the NACA0012 airfoil.

to avoid deformed elements.

The construction of basic meshes like shown in Figure 2.12(a) is a rather simple task due to their axial symmetry. The situation is different for obstacles such as airfoils that are not axially symmetric. In that case one possibility is to base the mesh on a basic mesh like given in Figure 2.12(a) and to move grid points on the inner boundary according to the airfoil geometry. In order to avoid deformed elements, again, it is advisable to move grid points of the second layer and so on. This can be done by smoothly mapping the whole basic mesh, denoted by triangulation T_h^b, to a mesh, T_h, whose outer boundary is unchanged and whose inner boundary matches the geometry of the airfoil. In the following we define this mapping that will be based on solutions to the Laplace equation.

Let the index set $\mathcal{J}$ of all points p_j^b, $j \in \mathcal{J}$, of a basic mesh T_h^b be separated into two distinct index subsets $\mathcal{J} = \mathcal{I} \cup \mathcal{B}$, where $\mathcal{I}$ denotes the index set of all interior points and $\mathcal{B}$ the index set of all points on the boundary of the (basic) mesh. Furthermore let $\{p_j\}_{j\in\mathcal{B}}$, be the set of all boundary points of the airfoil mesh given by a bijective mapping of the boundary vertices p_j^b, $j \in \mathcal{B}$, of the basic mesh.

Then, we define the following discrete problem: for $l = 1, 2$ find the discrete function u_l, continuous and bilinear on each element $K \in T_h^b$, such that

$$\begin{array}{rcll} -\Delta u_l(x) & = & 0, & x \in K, K \in T_h^b, \\ u_l(p_j^b) & = & (p_j)_l, & j \in \mathcal{B}. \end{array} \tag{2.53}$$

Then the inner vertices p_j, $j \in \mathcal{B}$, of the airfoil mesh, T_h, are given by

$$(p_j)_l = u_l(p_j^b), \quad j \in \mathcal{I}.$$

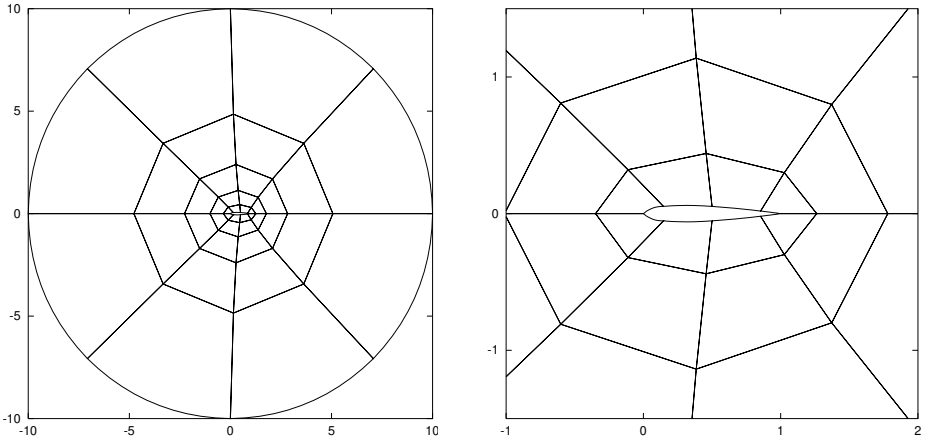
2.10.1 The NACA0012 airfoil

First, we consider the NACA0012 airfoil, see [1] for example; The upper and lower surfaces of this airfoil geometry are specified by the function $g^{\pm}$, where

$$g^{\pm}(s) = \pm 5 \cdot 0.12 \cdot (0.2969 s^{1/2} - 0.126 s - 0.3516 s^2 + 0.2843 s^3 - 0.1015 s^4), \tag{2.54}$$

cf. Figure 2.22. Here, $0 \le s \le l$, where the chord length l of the airfoil is $l \approx 1.00893$; thereby, we write $\hat{g}$ to denote the rescaling of g to yield an airfoil of unit (chord) length.

Figure 2.23 shows an airfoil mesh based on a basic mesh similar to that in Figure 2.12(a). In the zoomed part of the mesh, see Figure 2.23(b), we see that two elements at the trailing edge have very obtuse angles as the aifoil consists of a very acute angle at the trailing edge. We note that the obtuse angle keeps the



(a) (b)

Figure 2.23: Simple coarse mesh for the NACA0012 airfoil. (a) Mesh; (b) Zoom of the mesh; Two elements at the trailing edge have an angle of almost 180°.

same no matter how many times the mesh is refined. As deformed elements like those at the trailing edge are not wanted, in the following we create a modified mesh, see Figure 2.24.

2.10.2 The BAC3-11 airfoil

Here we consider the BAC3-11 airfoil, see Figure 2.25, an unsymmetric airfoil geometry that was originally specified in [2]. For this geometry mesh refinement may lead to very deformed and even to degenerate meshes. This subsection firstly reveals this problem that occurs when the mesh of this specific geometry is refined, and it shows how this difficulty may be overcome by locally modifying the method of mesh refinement.

Usually a quadrilateral element is refined to yield four disjoint sub–elements by connecting with lines the midpoints of opposing faces of the boundary of the element. Also on elements that include a face next to a (possibly curved) boundary of the domain, the refinement is based on the midpoint of that face. In general the midpoint on a curved boundary part is defined to be the point that splits the boundary face into two parts of equal length. This way the curved boundary is discretised by piecewise polynomials separated by equidistantly distributed points where here distance is measured in arc length.

Figure 2.26(a) shows a zoom of the mesh near the leading edge of the BAC3-11 airfoil. This is the mesh as it occurs after three global refinement steps. We see that the points at the boundary of the domain, i.e. on the profile of the airfoil, are equidistantly distributed. But furthermore, we see that the second element next to the airfoil and below the leading edge is strongly deformed in the sense that it includes an inner angle of more than 180°. The reason for this

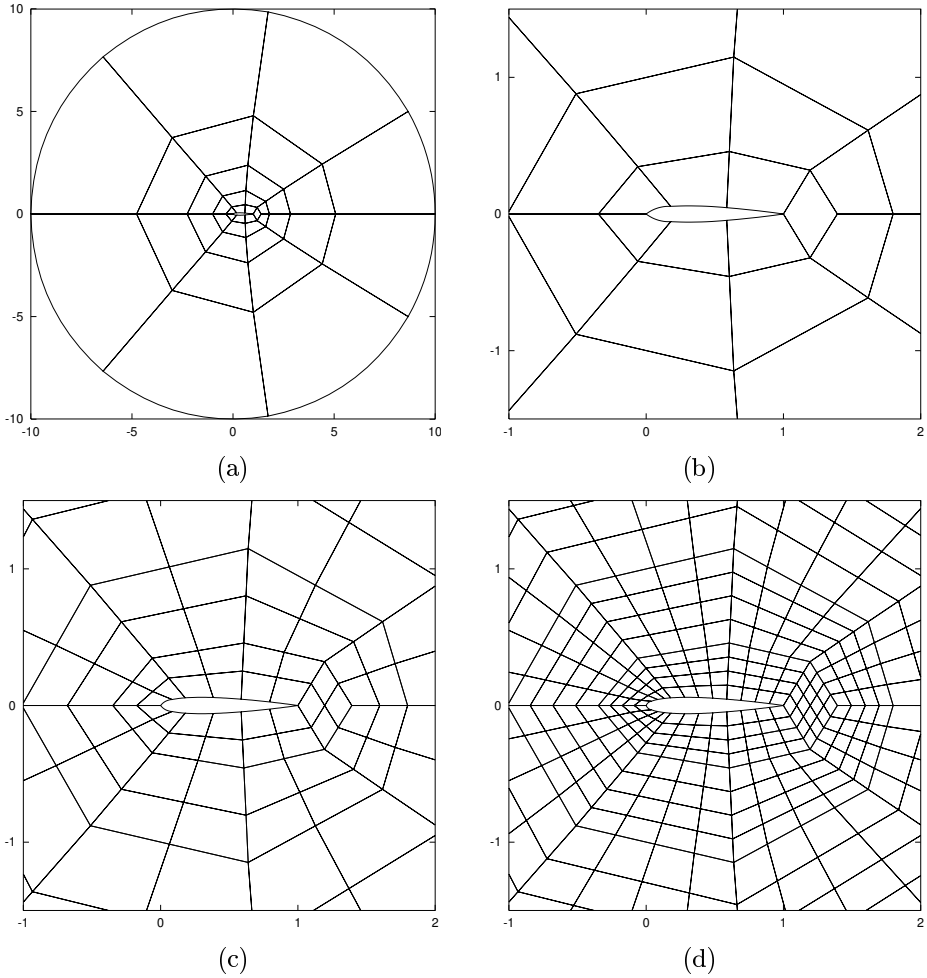


Figure 2.24: Modified coarse mesh for the NACA0012 airfoil. (a) Whole mesh; (b) Zoom into the mesh. At trailing edge: Four elements of the inner layer are now replaced by three elements. (c) Once globally refined; (d) Twice globally refined.

Figure 2.25: Profile of the BAC3-11 airfoil.

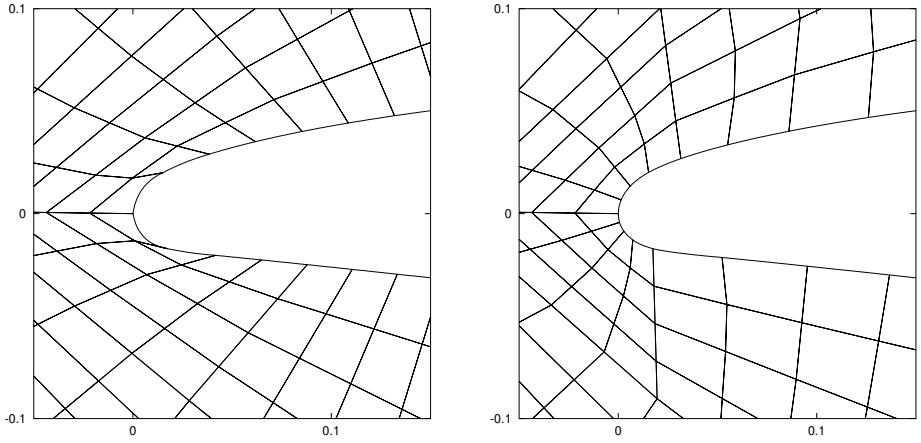


Figure 2.26: Mesh near the leading edge of the BAC3-11 airfoil. (a) Mesh refinement based on face midpoints results in deformed elements near the leading edge; (b) Modified refinement avoids deformation of elements.

distortion in the mesh is given by the strong curvature of this part of the boundary in combination with midpoints that seemingly are not chosen appropriate for this specific case. In order to overcome this difficulty, we modify the mesh refinement of boundary elements next to the leading edge by considering points on the boundary that subdivide the face into two parts with equal lengths measured in the y component, only. This way, we gain a mesh, see Figure 2.26(b), where the points on the boundary are not equally distributed but the mesh does not suffer any more from the flaw described above. Finally, Figure 2.27 shows the coarse mesh for the BAC3-11 airfoil and some globally refined meshes where here the modified refinement described above is employed.

2.11 Numerical example: Flow around the BAC3-11 airfoil

Here, at the end of this chapter, we show the numerical solution of a Mach 0.7 flow around the BAC3-11 airfoil at an angle of attack of 5 degrees. Figure 2.28 shows the Mach number isolines of numerical solutions on a sequence of locally refined meshes shown in Figure 2.29. We employ a refinement indicator that depends on the finite element residuals of the numerical solution. Other indicators depending on gradients or the curvature of the solution produce similar meshes. These indicators do not require the solution of an auxiliary problem (dual problem, cf. Chapter 3) and will be called *ad hoc* indicators, in the following.

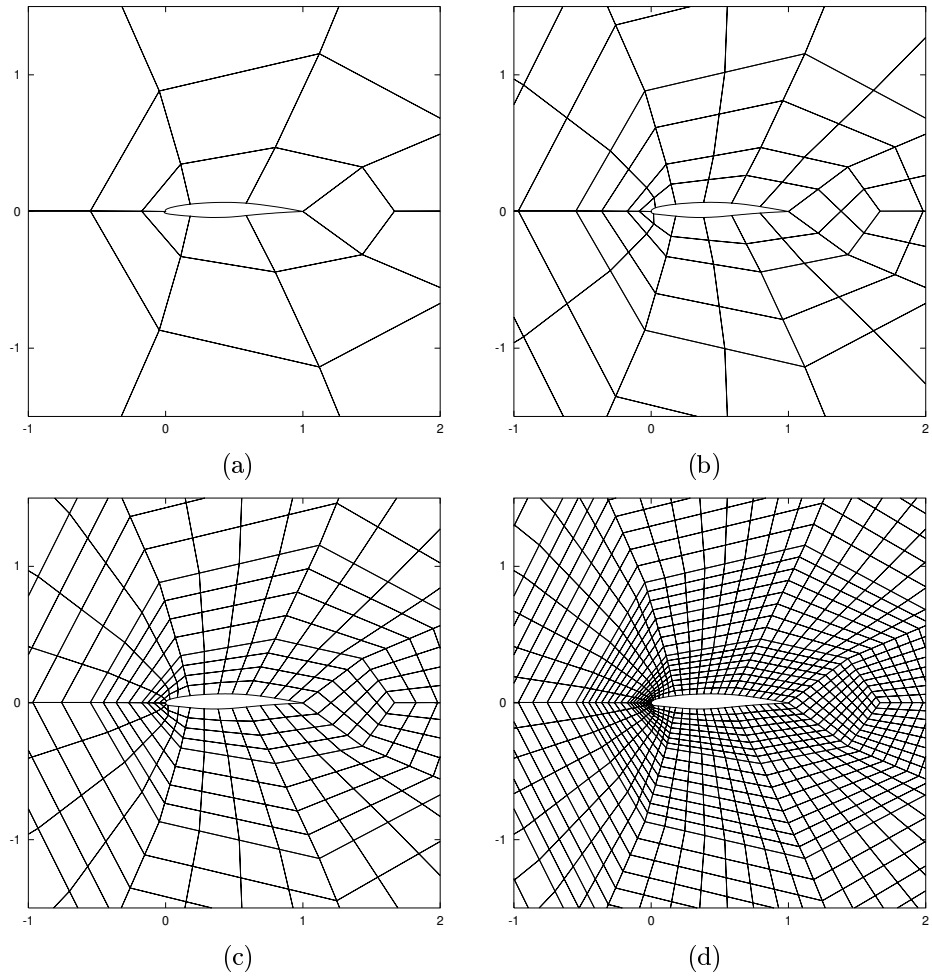


Figure 2.27: Globally refined meshes for the BAC3-11 airfoil.

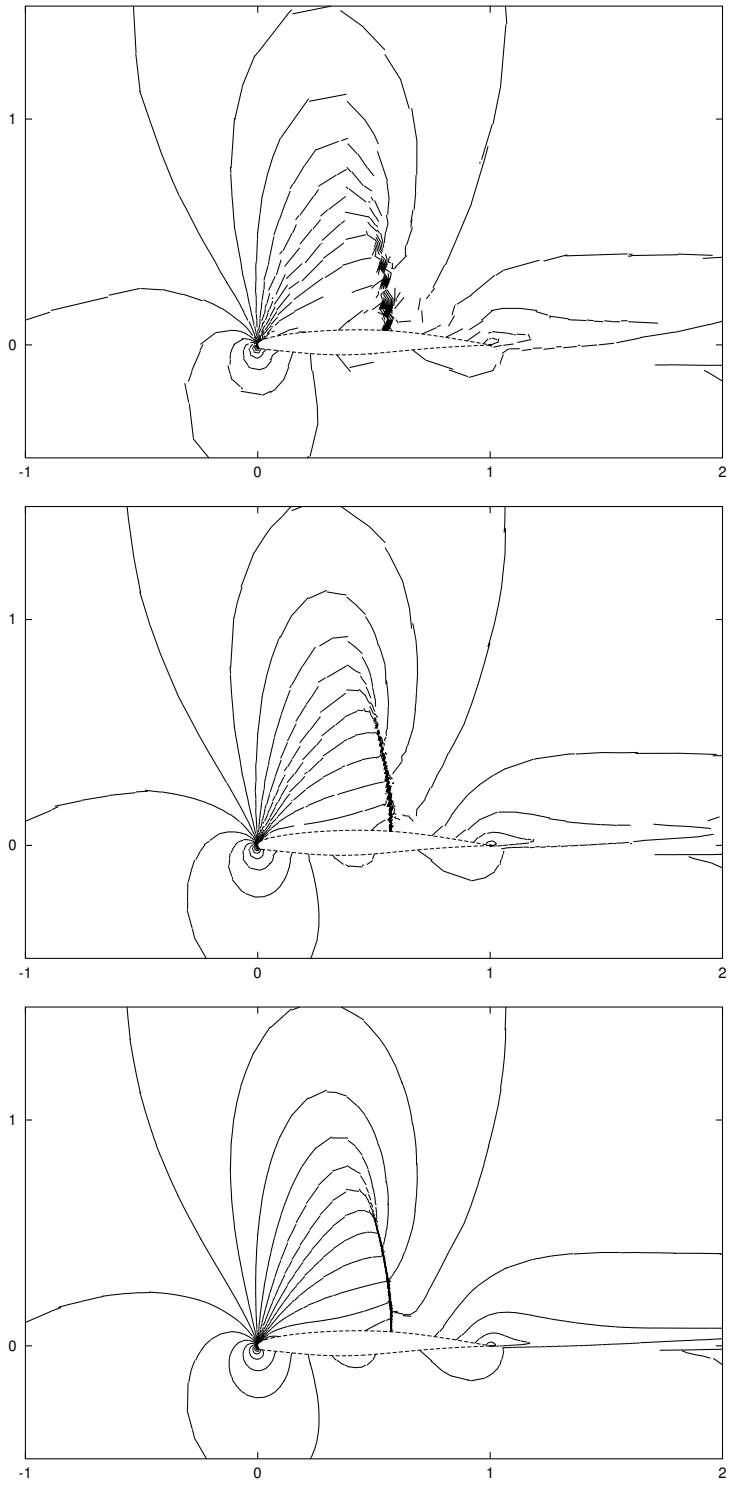


Figure 2.28: $M = 0.7$ flow around BAC3-11 airfoil at $\alpha = 5$ degrees. Mach number isolines $M_i = \frac{i}{16}$. Computed on meshes with 1020, 2913 and 8289 elements.

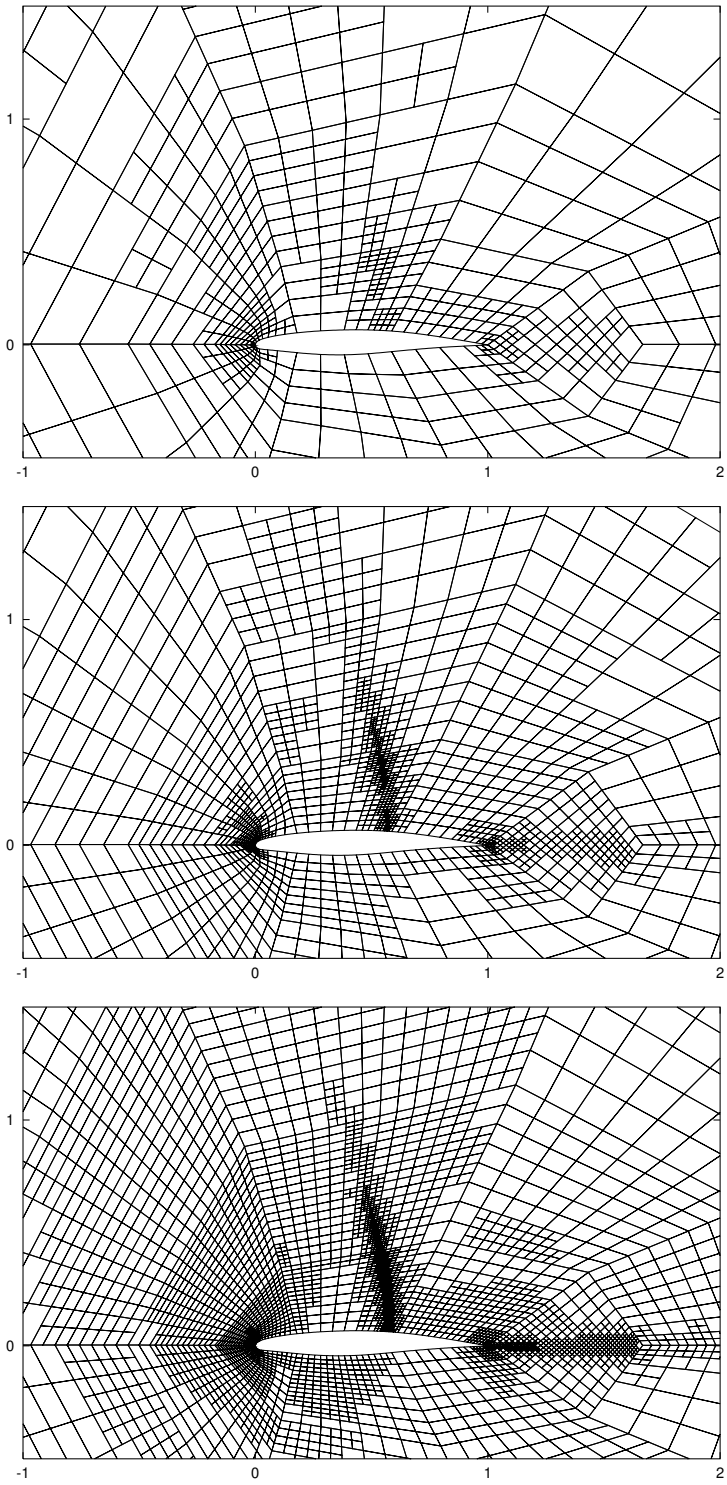


Figure 2.29: $M = 0.7$ flow around BAC3-11 airfoil at $\alpha = 5$ degrees. Meshes with 1020, 2913 and 8289 elements.

Chapter 3

A posteriori error estimation

In many applications there is particular interest in specific quantities of the solution. These include physically relevant quantities like, for example in aerodynamics, the drag or lift of an airfoil, the pressure difference between the leading and trailing edge of the airfoil or single density or pressure values on the profile of the airfoil. For ensuring a specific accuracy of the numerical computation it is essential to provide an *a posteriori* error bound. However, for ensuring a given accuracy in the computed quantity it is not useful to give error estimation in terms of global norms like the global L^1-norm, but it is necessary to provide error estimation in terms of the quantity of interest.

Therefore, in this section, we shall be interested in controlling the error of the numerical solution measured in terms of a given target functional $J(\cdot)$, sometimes also referred to as the error functional. Here, $J(\cdot)$ is the physical quantity of interest and may, for example, represent the drag or lift coefficient or a point value of the solution.

The proceeding error analysis is based on a hyperbolic duality argument using the general approach developed by Johnson *et al.* [20], and Rannacher *et al.* [7], see also [5, 24, 32, 51].

3.1 The dual problem

We begin by first introducing some notation. Assuming that the functional of interest $J(\cdot)$ is differentiable, we write $\bar{J}(\cdot;\cdot)$ to denote the mean value linearisation of $J(\cdot)$ defined by

$$\bar{J}(u, u_h; u - u_h) = J(u) - J(u_h) = \int_0^1 J'[su + (1-s)u_h](u - u_h)\, ds, \qquad (3.1)$$

where $J'[w](\cdot)$ denotes the functional (Fréchet) derivative of $J(\cdot)$ evaluated at some w in V. Here, V is some suitably chosen function space such that $V_h^p \subset V$.

Analogously, we write $L(u, u_h; \cdot, \cdot)$ to denote the mean–value linearisation of the semi-linear form $a(\cdot, \cdot)$ given by

$$\begin{aligned} L(u, u_h; u - u_h, v) &= a(u, v) - a(u_h, v) \\ &= \int_0^1 a'[su + (1-s)u_h](u - u_h, v)\, ds \end{aligned} \tag{3.2}$$

for all v in V. Here, $a'[w](\cdot, v)$ denotes the Fréchet derivative of $u \to a(u, v)$, for $v \in V$ fixed, at some w in V. We remark that the linearisation defined in equation (3.2) is only a *formal* calculation, in the sense that $a'[w](\cdot, v)$ may not in general exist. Instead, it must be replaced by a suitable approximation $\tilde{a}'[w](\cdot, v)$, see Section 2.8.4.

For the purposes of the proceeding a-posteriori error analysis, we *assume* that the linearisation performed in (3.2) is well–defined. Under this hypothesis, we introduce the following *dual* or *adjoint* problem: find $z \in V$ such that

$$L(u, u_h; w, z) = \bar{J}(u, u_h; w) \qquad \forall w \in V. \tag{3.3}$$

In the next section we discuss the well-posedness of the dual problem and provide some concrete examples when $a'[w](\cdot, v)$ may be uniquely determined. Further discussion concerning the numerical approximation of the dual problem is presented in Section 3.4.

3.2 Well-posedness of the dual problem

We assume that the dual problem (3.3) is well-posed and possesses a unique solution. Clearly, the validity of this assumption depends on both the definition of $L(u, u_h; \cdot, \cdot)$ and the choice of the linear functional under consideration and cannot be shown for general problems. Rather, here we present some important examples which are covered by our hypothesis.

3.2.1 Linear hyperbolic problems

Following the discussion in [32, 34] we consider the case of the scalar linear hyperbolic equation (2.12)

$$\begin{aligned} \beta \cdot \nabla u + bu &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_-. \end{aligned} \tag{3.4}$$

In the following we collect some linear target functionals for that the exact dual solution z defined in (3.3) is known to be well–posed:

1. *Outflow normal flux:* Given a weight function $\psi \in L^2(\Gamma_+)$ we consider the weighted normal flux through the outflow boundary Γ_+ that is defined by

$$J(w) = \int_{\Gamma_+} \beta \cdot n w \psi \, ds.$$

Then, z is the unique solution to the following boundary value problem: find $z \in V$ such that

$$\begin{aligned} -\nabla \cdot (\beta z) + bz &= 0 \quad \text{in } \Omega, \\ z &= \psi \quad \text{on } \Gamma_+. \end{aligned}$$

2. *Mean value:* Here, let ψ be a weight function in $L^2(\Omega)$. Then a weighted mean value is given by

$$J(w) = \int_\Omega w\psi \, ds.$$

In this case, z is the solution to following dual problem: find $z \in V$ such that

$$\begin{aligned} -\nabla \cdot (\beta z) + bz &= \psi \quad \text{in } \Omega, \\ z &= 0 \quad \text{on } \Gamma_+. \end{aligned}$$

3. *Point value:* Finally we consider a point value

$$J(w) = w(x_0)$$

at a given point $x_0 \in \Omega$. The existence and uniqueness of a dual solution corresponding to this non–regular target functional can be shown by first considering regularised target functionals $J_\epsilon(w) = \int_\Omega w\psi_{x_0,\epsilon} \, ds$ with $\psi_{x_0,\epsilon} = \epsilon^{-d}\phi((x - x_0)/\epsilon)$ where ϕ denotes a nonnegative function that is zero outside the unit ball. For $\epsilon \to 0$ this results in a unique solution z to the following dual problem

$$\begin{aligned} -\nabla \cdot (\beta z) + bz &= \delta_{x_0} \quad \text{in } \Omega, \\ z &= 0 \quad \text{on } \Gamma_+, \end{aligned}$$

where here, δ_{x_0} denotes a δ-distribution at point x_0 with the property

$$\int_\Omega \delta_{x_0} w \, ds = w(x_0).$$

Analogous well–posedness results for the dual problem also hold for systems of linear hyperbolic conservation laws provided the numerical flux function is chosen appropriately.

3.2.2 Nonlinear one–dimensional scalar hyperbolic problems

The question of well–posedness of the dual problem associated with nonlinear hyperbolic conservation laws is rather more delicate. Let us, for example, consider the following one–dimensional scalar hyperbolic equation

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = 0, \quad -\infty < x < \infty, \quad 0 < t \le T, \tag{3.5}$$

with strictly convex flux function f, i.e. $f'' > 0$, subject to the initial condition

$$u(x,0) = u_0(x), \quad -\infty < x < \infty. \tag{3.6}$$

Here, we suppose that (3.5), (3.6) is approximated by the standard Galerkin finite element method consisting of continuous piecewise polynomials in both space and time, cf. [51]. Furthermore, we suppose that the functional of interest represents the (weighted) mean value in space; i.e.

$$J(w) \equiv M_\psi(w) = \int_{-\infty}^{\infty} w\psi \, dx,$$

where $\psi \in L_2(-\infty, \infty)$ is a given weight function. According to (3.3), the dual problem (in strong form) takes the following form: find z such that

$$\begin{aligned} -\frac{\partial z}{\partial t} - a(x,t)\frac{\partial z}{\partial x} &= 0, \qquad -\infty < x < \infty, \quad 0 \le t < T, \\ z(x,T) &= \psi(x), \quad -\infty < x < \infty, \end{aligned} \tag{3.7}$$

where

$$a(x,t) = \int_0^1 f'((1-s)u + su_h)\, ds.$$

Although the dual problem (3.7) is linear, this is a non-standard hyperbolic problem since the spatial velocity field a may be discontinuous. Thereby, in this case the standard classical theory of well–posedness of linear hyperbolic equations does not apply. However, assuming that $a \in L_\infty(\mathbb{R} \times (0,T))$ and satisfies a one–sided Lipschitz condition, Tadmor [53] has shown that (3.7) is a meaningful problem. For further details, and the verification that a satisfies the desired hypotheses when f is strictly convex, we refer to Süli [51]. For related work on the linearisation of conservation laws and the well–posedness of linear hyperbolic problems with discontinuous data, we refer to [9, 23, 55], and the references cited therein.

In general, however, it is not known whether the dual problem (3.3) possesses a unique solution when the hyperbolic conservation law (2.2) is nonlinear, and when the discontinuous Galerkin approximation (2.35) is employed, even for scalar problems, corresponding to $m = 1$. We can only *hypothesise* that the computation of a physically correct approximation to (2.2) will yield a uniquely solvable dual problem. In the absence of such analytical results, we must rely solely on numerical simulation to confirm the validity of this hypothesis. Indeed, in Chapter 4, we consider a series of test problems which provide numerical evidence to indicate that the dual problem is solvable for the particular problems and target functionals selected.

3.3 Error representation formula

As indicated in the last section it is not known whether the dual problem (3.3) possesses a unique solution when the hyperbolic conservation law (2.2) is nonlinear, and when the discontinuous Galerkin approximation (2.35) is employed.

For the proceeding error analysis, we must therefore *assume* that the dual problem (3.3) is well–posed. Under this assumption, we have the following general result.

Theorem 3.1 *Let u and u_h denote the solutions of (2.2) and (2.35), respectively, and suppose that the dual problem (3.3) is well–posed. Then, the following error representation formula holds:*

$$J(u) - J(u_h) = \mathcal{E}_\Omega(u, u_h, h, z - z_h) \equiv \sum_{K \in T_h} \eta_K, \tag{3.8}$$

where

$$\eta_K = \int_K R(u_h)\,(z - z_h)\,dx + \int_{\partial K} r(u_h)\,(z - z_h)^+\,ds - \int_K \varepsilon \nabla u_h \cdot \nabla (z - z_h)\,dx \tag{3.9}$$

for all z_h in V_h^p. Here, $R(u_h)$ and $r(u_h)$ denote the internal and boundary finite element residuals, respectively, defined on $K \in T_h$ by

$$R(u_h)|_K = -\nabla \cdot F(u_h) \quad \textit{and} \quad r(u_h)|_K = F(u_h^+) \cdot n - H(u_h^+, u_h^-, n), \tag{3.10}$$

respectively.

Proof: Choosing $w = u - u_h$ in (3.3), recalling the linearisation performed in (3.1), and exploiting the Galerkin orthogonality property (2.39), we get

$$\begin{aligned} J(u) - J(u_h) &= \bar{J}(u, u_h; u - u_h) \\ &= L(u, u_h; u - u_h, z) \\ &= L(u, u_h; u - u_h, z - z_h) \\ &= -a(u_h, z - z_h) \end{aligned}$$

for all z_h in the finite element space V_h^p. By employing the divergence theorem, we deduce that

$$\begin{aligned} J(u) - J(u_h) = \sum_{K \in T_h} \Big\{ &- \int_K \nabla \cdot F(u_h)(z - z_h)\,dx \\ &+ \int_{\partial K} \left(F(u_h^+) \cdot n - H(u_h^+, u_h^-, n)\right)\,(z - z_h)^+\,ds \\ &- \int_K \varepsilon \nabla u_h \cdot \nabla (z - z_h)\,dx \Big\}. \end{aligned} \tag{3.11}$$

Using the definition of the finite element residuals in (3.10), gives the desired result. □

Based on the general error representation formula derived in Theorem 3.1, *a posteriori* error estimates bounding the error in the computed functional $J(\cdot)$

may be deduced. Before we proceed, let us first recall the classification of Type I and Type II *a posteriori* error bounds introduced in [34]. Type I *a posteriori* error bounds, also referred to as *weighted a posteriori* error bounds, involve the multiplication of the residual terms $R(u_h)$ and $r(u_h)$ and the shock capturing term $\varepsilon\nabla u_h$, by the difference between the dual solution z and its projection, interpolant or quasi-interpolant z_h. On the other hand, Type II *a posteriori* bounds are in the spirit of the error estimates derived by C. Johnson *et al.* [20], and do not depend explicitly on the dual solution. These latter error estimates are derived from Type I *a posteriori* bounds by employing standard results from approximation theory, together with well–posedness results for the dual problem. The resulting Type II error bounds only involve certain norms of the residuals and shock–capturing term, the mesh function h, interpolation constants and the stability factor of the dual problem.

However, as we shall see in Chapter 4, the elimination of the 'weighting terms' involving the difference between z and z_h may adversely affect the performance of our adaptive finite element method. Indeed, mesh refinement strategies based on Type II *a posteriori* error bounds which do not require the computation of the dual solution may, under mesh refinement, lead to suboptimal convergence of the error $|J(u) - J(u_h)|$ in the computed functional, resulting in uneconomical mesh design and an inefficient adaptive algorithm (see [24, 32], for related work). For this reason, we only consider the derivation of the following Type I *a posteriori* error bound that follows trivially from (3.8) by application of the triangle inequality.

Corollary 3.2 *Under the assumptions of Theorem 3.1, the following Type I* a posteriori *error bound holds:*

$$|J(u) - J(u_h)| \leq \mathcal{E}_{|\Omega|}(u, u_h, h, z - z_h) \equiv \sum_{K \in T_h} |\eta_K| \,, \tag{3.12}$$

where η_K is defined as in (3.9).

We end this section by noting that for nonlinear hyperbolic conservation laws and/or nonlinear target functionals $J(\cdot)$, both the error representation formula (3.8) and the resulting Type I *a posteriori* error bound (3.12) depend on the unknown analytical solution to both the primal and dual problems, u and z, respectively. Thus, in order to render these error estimates computable, both u and z must be replaced by suitable approximations which do not adversely affect the quality of the resulting (approximate) error representation formula and (approximate) Type I *a posteriori* error bound, cf. Chapter 4 for numerical validation. This subject will be the topic of the following section.

3.4 Numerical approximation of the dual problem

To ensure that the error representation formula proved in Theorem 3.1 and the subsequent error bound derived in Corollary 3.2 are genuinely *a posteriori*,

all non-computable quantities appearing in the right–hand side of (3.8) and (3.12), respectively, must be replaced by suitable approximations; namely, the analytical solution to the primal problem u, and the analytical solution to the dual problem z.

Let us first deal with the approximation of u. To this end, we note that the dependence of the error representation formula $\mathcal{E}_\Omega$ and the *a posteriori* error bound $\mathcal{E}_{|\Omega|}$ on u stems from the mean value linearisation of $a(\cdot, v)$, $v \in V$ fixed, and $J(\cdot)$ carried out in (3.2) and (3.1), respectively. Hence, for linear hyperbolic conservation laws and linear target functionals $J(\cdot)$, the dependence of the error representation formula (3.8) and the *a posteriori* error bound (3.12) on u does not arise. In practice, for nonlinear problems and/or nonlinear target functionals, the linearisation leading to $L(u, u_h; \cdot, \cdot)$ and $\bar{J}(u, u_h; \cdot)$ is performed about the numerical solution u_h, rather than at some convex combination of u and u_h; i.e. $L(u, u_h; \cdot, \cdot)$ is approximated by $L(u_h, u_h; \cdot, \cdot)$ in (3.2) and $\bar{J}(u, u_h; \cdot)$ is approximated by $\bar{J}(u_h, u_h; \cdot)$ in (3.1), cf. [7], for example.

Secondly, the dual problem must be numerically approximated; here, there are essentially two further sources of error introduced into the computation of $\mathcal{E}_\Omega$ and $\mathcal{E}_{|\Omega|}$. Firstly, since the formal calculation of the Fréchet derivative of $u \to a(u, v)$, for $v \in V$ fixed, at some w in V may not in general exist, it must be replaced by a suitable approximation. Denoting this approximation by $\tilde{a}'[w](\cdot, v)$, we define

$$\tilde{L}(u, u_h; u - u_h, v) = \int_0^1 \tilde{a}'[su + (1-s)u_h](u - u_h, v)\, ds\,. \tag{3.13}$$

Calculating the linearisation performed in (3.13) at u_h, rather than at a convex combination of u and u_h (cf. above), we now define the following (approximate) dual problem: find $\tilde{z} \in V$ such that

$$\tilde{L}(u_h, u_h; w, \tilde{z}) = \bar{J}(u_h, u_h; w) \quad \forall w \in V. \tag{3.14}$$

We recall from the previous section that there were open questions concerned with the existence and uniqueness of the formal dual problem defined in (3.3). Indeed, for general nonlinear systems of hyperbolic conservation laws, it is not at all clear whether (3.3) is well posed for a given target functional $J(\cdot)$. In practice, we are concerned about the well–posedness of the approximate dual problem (3.14). Given that the dual problem (3.14) serves only as an approximation to the true dual problem (3.3), further approximations may be made in the definition of (3.14) in order to ensure that the resulting system of partial differential equations is well–posed. For example, the approximate dual problem (3.14) may have additional artificial viscosity included in order to guarantee the existence and uniqueness of $\tilde{z}$. For the purposes of the this work, this last approximation is not performed, since for the model problems considered here, the dual problem (3.14) was always found to be numerically solvable. Of course, for more complicated engineering problems, this may not be the case, and the additional regularisation provided by a 'tune–up' of an artificial viscosity term may be necessary.

Finally, we note that once a suitable approximate dual problem has been defined, its analytical solution $\tilde{z}$ must be numerically determined. Writing $\tilde{z}_{\bar{h}}$ to denote this approximation, we remark that $\tilde{z}_{\bar{h}}$ should not be calculated using the same finite element space V_h^p employed for the primal problem; otherwise the resulting error representation formula would be identically zero. In practice, there are essentially three approaches to computing a numerical approximation $\tilde{z}_{\bar{h}}$ of $\tilde{z}$. The first approach is to keep the degree p of the approximating polynomial used to compute u_h fixed, but compute $\tilde{z}_{\bar{h}}$ on a sequence of dual finite element meshes $\hat{T}_{\bar{h}}$ which, in general, differ from the "primal meshes" T_h. Alternatively, $\tilde{z}_{\bar{h}}$ may be computed using piecewise discontinuous polynomials of degree $\tilde{p}$, $\tilde{p} > p$, on the same finite element mesh T_h employed for the primal problem. A variant of this second approach is to compute the approximate dual problem using the same mesh T_h and polynomial degree p employed for the primal problem, i.e. $\tilde{z}_{\bar{h}} \in V_h^p$, and to take a patchwise higher order recovery $\tilde{z}_{\bar{h}} \in V_{2h}^{\tilde{p}}$, $\tilde{p} > p$. While this latter approach is the cheapest of the three methods, and is still capable of producing adaptively refined meshes specifically tailored to the selected target functional, the quality of the resulting approximate error representation formula may be poor, cf. Section 3.6. Therefore we prefer the second approach, i.e. $\tilde{z}_{\bar{h}} \equiv \tilde{z}_h \in V_h^{\tilde{p}}$, with $\tilde{p} > p$.

To summarise, in practice there are three sources of error in the computation of the error representation formula (3.8) and the Type I *a posteriori* error bound (3.12):

1. Linearisation error stemming from the replacement of u by u_h in (3.2) and (3.1);

2. Error created by the approximation of the Fréchet derivative of $a(\cdot, v)$, $v \in V$ fixed, in the direction $u - u_h$;

3. Error generated in the numerical approximation of the (approximate) dual problem.

Notwithstanding these approximations, we shall show through numerical experimentation in Chapter 4, that the reliability of the Type I *a posteriori* error bound (3.12) is not compromised, in the sense that

$$\tilde{\mathcal{E}}_{|\Omega|} \equiv \mathcal{E}_{|\Omega|}(u_h, u_h, h, \tilde{z}_h - z_h) = \sum_{K \in T_h} |\tilde{\eta}_K|, \tag{3.15}$$

where $\tilde{\eta}_K$ is defined as in (3.9) with z replaced by $\tilde{z}_h$, remains an upper bound on the true error in the target functional $J(\cdot)$. In particular, we shall show that the ratio of the approximate error representation formula

$$\tilde{\mathcal{E}}_{\Omega} \equiv \mathcal{E}_{\Omega}(u_h, u_h, h, \tilde{z}_h - z_h) = \sum_{K \in T_h} \tilde{\eta}_K \tag{3.16}$$

and the true value $J(u) - J(u_h)$ is very close to one; see [24, 31, 32, 40], for related work.

Remark 3.3 *In the error representation formula (3.8), the function z_h may be chosen to be any element from the finite element space V_h^p employed to approximate the primal problem (2.2). Here, we select z_h to be the projection of the numerical approximation $\tilde{z}_h$ to the dual problem (3.14) onto the finite element space V_h^p. In the theory of* a posteriori *error estimation of continuous Galerkin finite element methods this choice is extremely important in order to obtain local error indicators $|\tilde{\eta}_K|$ which exhibit the optimal rate of convergence as the mesh is refined. However, for the discontinuous Galerkin method, this choice is less important, since discontinuous finite element schemes satisfy a Galerkin orthogonality property, cf. (2.39), both on a global and* local *level. Thereby, employing the local Galerkin orthogonality property, the error indicators $\tilde{\eta}_K$ become*

$$\tilde{\eta}_K = \int_K R(u_h)\,\tilde{z}_h\,dx + \int_{\partial K} r(u_h)\,\tilde{z}_h^+\,ds - \int_K \varepsilon\nabla u_h\cdot\nabla\tilde{z}_h\,dx; \tag{3.17}$$

i.e. they no longer depend on the choice of z_h.

As a final note, we remark that even if the *a posteriori* error bound (3.12) should fail to remain an upper bound on the true error in the computed functional $J(\cdot)$, then the adaptive mesh refinement algorithm outlined in the following section still provides the necessary local information to ensure that economical meshes, specifically tailored to the approximation of the underlying functional $J(\cdot)$ of interest, are generated.

3.5 Adaptive mesh refinement

In this section we consider the design of an adaptive algorithm to ensure the efficient computation of the given target functional $J(\cdot)$ of practical interest. To this end, we employ the (approximate) *a posteriori* error bound $\tilde{\mathcal{E}}_{|\Omega|}$, cf. (3.15), to determine when the desired level of accuracy has been achieved. For example, suppose that the aim of the computation is to compute $J(\cdot)$ such that the error $|J(u)-J(u_h)|$ is less than some user–defined tolerance `TOL`, i.e.

$$|J(u)-J(u_h)| \leq \texttt{TOL},$$

then, in practice we may enforce the stopping criterion

$$\tilde{\mathcal{E}}_{|\Omega|} \leq \texttt{TOL}. \tag{3.18}$$

If the condition (3.18) is not satisfied on the current finite element mesh T_h, then the elementwise terms $|\tilde{\eta}_K|$ are employed as local error indicators to guide mesh refinement and coarsening. Hence, the cycle of the adaptive mesh refinement may be outlined as follows:

1. Construct an initial mesh T_h.
2. Compute the numerical approximation to the primal problem $u_h \in V_h^p$ on the current mesh T_h.

3. Compute the numerical approximation to the dual problem $\tilde{z}_h \in V_h^{\bar{p}}$ on the same mesh employed for the primal problem, with $\tilde{p} > p$.

4. Evaluate the approximate error representation $\tilde{\mathcal{E}}_\Omega = \sum_{K \in T_h} \tilde{\eta}_K$, cf. (3.15).

5. If $\tilde{\mathcal{E}}_\Omega \approx$ `TOL`, where `TOL` is a given tolerance, then STOP.

6. Otherwise, refine and coarsen a fixed fraction of the total number of elements according to the size of the local error indicators $|\tilde{\eta}_K|$ and generate a new mesh T_h; GOTO 2.

The goal of the mesh refinement and coarsening is to equidistribute the local error indicators $|\tilde{\eta}_K|$. During each refinement and coarsening step a fixed fraction of the total number of elements are refined and coarsened. In the following section, we set the refinement and coarsening fractions to be 20% and 10%, respectively.

3.6 Numerical approximation of the dual problem: Numerical Experiment

The solution $\tilde{z}$ to the approximate dual problem (3.14) is not known in general and must be determined numerically. Writing $\tilde{z}_h$ to denote this approximation, we recall from Section (3.4) that $\tilde{z}_h$ should not be calculated using the same finite element space V_h^p employed for the primal problem. In the following we employ the second approach of numerically approximating the dual solution introduced in Section 3.4, i.e. $\tilde{z}_h$ will be computed using piecewise discontinuous polynomials of degree $\tilde{p}$, $\tilde{p} > p$, on the same finite element mesh T_h employed for the primal problem. In comparison to that we employ a variant of the approach, i.e. we compute the approximate dual problem using the same mesh T_h and polynomial degree p employed for the primal problem, i.e. $\tilde{z}_h \in V_h^p$, and take a patchwise higher order recovery $\tilde{z}_h \in V_{2h}^{\bar{p}}$, $\tilde{p} > p$.

In the following we consider the numerical solution to the "Ringleb flow" problem, cf. [47], that represents a smooth transonic flow in a channel. The solution to this problem is known and may be obtained via hodograph transformation, see [13] or Appendix B.5. This example is one of the few non-trivial problems of the 2D Euler equations for which a (smooth) analytical solution is known. It offers the opportunity to properly test the sharpness of the approximate error representation for the 2D Euler equations.

In the following we approximate this problem by piecewise bilinear functions, i.e. $u_h \in V_h^p$ with $p = 1$, and a piecewise quadratic boundary approximation. We select the target functional to be the horizontal force component exerted on the channel walls. More precisely, we choose

$$J(u) = \int_{\text{wall}} (\psi \cdot n) p \, ds,$$

		$\tilde{z}_h^{(2)} \in V_h^2$		$\tilde{z}_h^{(3)} \in V_h^3$	$\tilde{z}_h^{(1)} \in V_{2h}^2$
# el.	$J(u) - J(u_h)$	$\tilde{\mathcal{E}}_\Omega^{(2)}$	$\theta^{(2)}$	$\theta^{(3)}$	$\theta^{(1)}$
32	-6.946e-04	-8.609e-04	1.239	1.001	0.65
56	9.940e-05	6.123e-05	0.616	0.969	-1.13
80	-5.332e-05	-7.316e-05	1.372	0.934	2.78
125	7.571e-05	7.165e-05	0.946	1.006	0.03
212	-6.735e-06	-9.230e-06	1.370	0.938	2.14
353	1.634e-05	1.621e-05	0.992	1.018	0.07
566	5.681e-06	5.541e-06	0.975	1.004	1.21
950	2.778e-06	2.763e-06	0.995	1.009	2.21
1562	1.344e-06	1.367e-06	1.017	1.020	1.05
2534	1.750e-07	1.807e-07	1.032	1.017	1.78

Table 3.1: Ringleb flow problem, $J(u)$ = horizontal force. Data of adaptive refinement by indicators $|\tilde{\eta}_K^{(2)}|$. Comparison of the efficiency indices $\theta^{(\tilde{p})} = \tilde{\mathcal{E}}_\Omega^{(\tilde{p})}/J(e)$ for $\tilde{p} = 2$, 3 and 1.

where $\psi = (1, 0)$, n denotes the unit outward normal vector to the boundary and p is the pressure; this is a slight variant of the target functional selected in [40]. In this case, the true value of the functional on the exact geometry is given by $J(u) = -0.5744441759095$. We note that since the pressure p is derived from the conserved variables $(\rho, \rho u, \rho v, \rho E)$ using the equation of state (2.5), the selected target functional $J(\cdot)$ is nonlinear.

First we compare the approximate error representations

$$\tilde{\mathcal{E}}_\Omega^{(\tilde{p})} = \sum_{K \in T_h} \tilde{\eta}_K^{(\tilde{p})}, \qquad \tilde{p} = 1, 2, 3,$$

with $\eta_K^{(\tilde{p})}$ as defined in (3.17) where the numerical dual solution $\tilde{z}_h$ is computed using polynomials of degree $\tilde{p}$, i.e. $\tilde{z}_h = \tilde{z}_h^{(1)} \in V_{2h}^{\tilde{p}}$ and $\tilde{z}_h = \tilde{z}_h^{(\tilde{p})} \in V_h^{\tilde{p}}$, $\tilde{p} = 2, 3$, respectively. Table 3.1 shows the data of the adaptive refinement using the weighted indicators $|\tilde{\eta}_K^{(2)}|$. It includes the number of elements, the true error in the functional, $J(u) - J(u_h)$, the computed approximate error representation $\tilde{\mathcal{E}}_\Omega^{(2)}$ and its effectivity index $\theta^{(2)} = \tilde{\mathcal{E}}_\Omega^{(2)}/J(e)$ and the effectivity indices $\theta^{(i)} = \tilde{\mathcal{E}}_\Omega^{(i)}/J(e)$, $i = 1, 3$.

We see that, although on the coarser meshes the effectivity index $\theta^{(2)}$ still shows some variation, the quality of the computed error representation formula is very good, i.e. $\theta^{(2)} \approx 1$. Indeed, already after five refinement steps it is very close to one.

For comparison, the next column of the table shows the respective effectivity indices $\theta^{(3)} = \tilde{\mathcal{E}}_\Omega^{(3)}/J(e)$ for the dual solutions computed with piecewise polynomials of degree three, i.e. $\tilde{z}_h^{(3)} \in V_h^3$. We see that here $\tilde{\mathcal{E}}_\Omega^{(3)}$ and $\tilde{\mathcal{E}}_\Omega^{(2)}$ result in comparably good error estimation; on some refinement levels, espe-

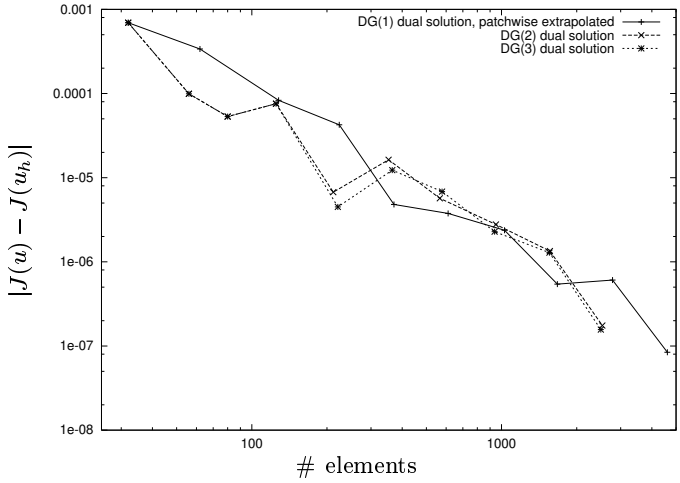


Figure 3.1: Ringleb flow problem, $J(u)$=horizontal force. Convergence of the error on meshes produced using weighted indicators $|\tilde{\eta}_K^{(\tilde{p})}|$, with $\tilde{p} = 1, 2$ and 3.

cially on coarse meshes, $\tilde{\mathcal{E}}_\Omega^{(3)}$ gives an even more accurate error estimation than $\tilde{\mathcal{E}}_\Omega^{(2)}$. However, for applications, this difference in quality of the error estimation is not significantly enough to justify the additional amount of work and time required for the computation of the dual solution $\tilde{z}_h^{(3)} \in V_h^3$ in comparison to the computation of $\tilde{z}_h^{(2)} \in V_h^2$.

Additionally these results are compared with the error estimation based on a very cheap variant of computing the dual solutions. For this variant the dual solution is computed on the same mesh T_h using polynomials of the same degree as for the primal solution, i.e. $\tilde{z}_h \in V_h^1$. Then a higher order recovery is taken resulting in a numerical dual solution $\tilde{z}_h \in V_{2h}^2$. In the last column of Table 3.1 we show the effectivity indices $\theta^{(1)}$ of the approximate error representation evaluated using $\tilde{z}_h \in V_{2h}^2$. We see that here the error estimation is far less accurate than in the case of $\tilde{z}_h \in V_h^2$.

Nevertheless, the meshes produced using the weighted indicators $|\tilde{\eta}_K^{(1)}|$ are as efficient for evaluating the target functional as the meshes produced using the weighted indicators $|\tilde{\eta}_K^{(2)}|$ and $|\tilde{\eta}_K^{(3)}|$. This can clearly be seen in Figure 3.1 that shows the convergence of the target functional evaluated for numerical solutions computed on adaptively refined meshes produced using the weighted indicators $|\tilde{\eta}_K^{(\tilde{p})}|$, for $\tilde{p} = 1, 2$ and 3.

If not stated differently the dual solution will always be $z_h \in V_h^2$ in the following. This provides a good compromise between accurate *a posteriori* error estimation and the computational expense of computing the dual solution.

Chapter 4

Numerical examples of adaptive mesh refinement and *a posteriori* error estimation

This chapter provides a collection of several numerical examples covering a wide range of different hyperbolic problems: simple scalar and linear problems as well as complex nonlinear problems pertained to the 2D compressible Euler equations. All examples illustrate aspects of adaptive mesh refinement and *a posteriori* error estimation for hyperbolic problems. The variety of problems allows a deep insight into the mechanism of information transport, sources and transport of errors and the adaptive strategy that is employed to reduce the error measured in terms of specific target functionals.

The numerical examples highlight the advantages of designing an adaptive finite element algorithm based on the weighted error indicators $|\tilde{\eta}_K|$ in comparison with traditional refinement strategies which do not require the solution of the dual problem (3.3). From (3.17), we recall the weighted indicator $|\tilde{\eta}_K|$ with

$$\tilde{\eta}_K = \int_K R(u_h)\,\tilde{z}_h\,dx + \int_{\partial K} r(u_h)\,\tilde{z}_h^+\,ds - \int_K \varepsilon \nabla u_h \cdot \nabla \tilde{z}_h\,dx. \tag{4.1}$$

It consists of the finite element residuals, see (3.10), multiplied by terms of the approximate dual solution $\tilde{z}_h$. Meshes produced using this indicator are specifically tailored to the efficient computation of the quantity of interest, $J(u)$.

The traditional refinement involves the use of *ad hoc* indicators that either include residuals of the solution or simple smoothness information like gradients or second derivatives of the solution, only. Here, we consider the *ad hoc* error indicator

$$\eta_K^{\text{adhoc}} = \|hR(u_h)\|_{L_2(K)} + \|h^{1/2} r(u_h)\|_{L_2(\partial K)}, \tag{4.2}$$

which stems from a Type II *a posteriori* error analysis, cf. [34].

In the following and if not stated differently, the primal problem will be approximated with discontinuous piecewise bilinear functions, i.e. $u_h \in V_h^1$ and the dual problem will be approximated by discontinuous piecewise biquadratic functions, i.e. $\tilde{z}_h \in V_h^2$. Furthermore, we note that all computations are performed with the fixed fraction mesh refinement algorithm with refinement and derefinement fractions set to 20% and 10%, respectively.

We begin this collection of numerical examples by considering several introductive test cases including scalar linear and nonlinear hyperbolic problems. For these examples it is easy to track paths of information and error transport and to understand the structure of the dual solution and the resulting adaptive mesh refinement. We then proceed with nonlinear hyperbolic systems including 2D compressible Euler flow computations around airfoils at the end of this chapter. In the following, we give a short overview of the examples.

4.1 Linear advection equation. Simple introductive example illustrating basics of information and error transport and the adaptive refinement tailored to the efficient computation of an weighted integral over the outflow boundary. Comparison of meshes produced using *ad hoc* and weighted indicator.

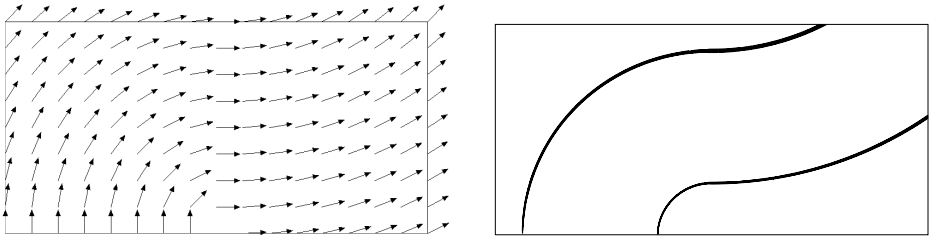
4.2 1D inviscid Burgers equation in (x, t) plane. Solution includes shocks. Target quantity is a point value. The approximate error representation is sharp whereas the Type I error bound largely overestimates the error. The meshes produced using weighted indicators are significantly more efficient than those produced using *ad hoc* indicators. Here, the comparison is based on number of elements as well as on cpu time.

4.3 Buckley-Leverett equation, an equation with non-convex flux. Solution to Riemann problem includes shock and rarefaction wave. Target functional is a point evaluation. Comparison of meshes produced by *ad hoc* and weighted indicator.

4.4 1D compressible Euler equation. Shock-tube problem. Dual solution to point evaluation consists of three spikes corresponding to the three characteristic directions of this hyperbolic system. Comparison of meshes produced by *ad hoc* and weighted indicator.

4.5 2D compressible Euler equation. Several examples including smooth solutions (Ringleb flow problem), solutions with shocks (supersonic flow past a wedge), and sub-, trans- and supersonic flows around airfoils. For a more detailed overview of these examples, see the beginning of Section 4.5.

We note that in some of the examples, mainly in the scalar problems, the convergence of the target quantities under adaptive refinement is displayed beyond relevant error sizes. These examples should be regarded as model problems only, explaining and illustrating the main ingredients of the approach. However, the airfoil computations in the last subsection are performed until a relevant size (1% - 5%) of the relative error in the target quantity is obtained.



(a) (b)

Figure 4.1: (a) Vector field β; (b) Isolines of primal solution.

4.1 Linear advection equation

In this first example we consider the linear hyperbolic equation

$$\begin{aligned} \beta \cdot \nabla u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma_-, \end{aligned}$$

with $\Omega = [0,2] \times [0,1] \subset \mathbb{R}^2$ and advection direction $\beta = \frac{\tilde{\beta}}{|\beta|}$, where

$$\tilde{\beta}(x,y) = \begin{cases} (y, 1-x), & \text{for } x < 1, \\ (2-y, x-1), & \text{otherwise;} \end{cases}$$

the vector field is displayed in Figure 4.1(a). Furthermore, we prescribe the boundary value function g on the inflow boundary to be

$$g(x,y) = \begin{cases} 1, & \text{for } (x,y) \in [\frac{1}{8}, \frac{3}{4}] \times \{0\}, \\ 0, & \text{otherwise.} \end{cases}$$

The solution, see the isoline plot in Figure 4.1(b), includes two discontinuities originating from the two jumps of the boundary function that are transported along the characteristic directions given by the vector field β.

Suppose that we are interested in the values of the solution on part $\frac{1}{4} < y < 1$ of the right outflow boundary. We set the target functional $J(u)$ to be

$$J(u) = \int_{\partial\Omega^+} u\psi \, ds,$$

where ψ is chosen to be a smooth weight function

$$\psi(x,y) = \begin{cases} \exp\left((\frac{3}{8})^{-2} - ((y-\frac{5}{8})^2 - \frac{3}{8})^{-2}\right), & \text{for } (x,y) \in \{2\} \times (\frac{1}{4}, 1), \\ 0, & \text{otherwise.} \end{cases}$$

In this case also the corresponding dual solution is smooth, see Figure 4.2. Thereby, the numerical dual solution $\tilde{z}_h$ is a very good approximation to the exact dual solution z. This results in an approximate error representation $\tilde{\mathcal{E}}_\Omega$

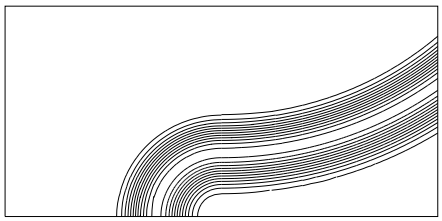


Figure 4.2: Isolines of the dual solution.

that even on coarse meshes gives very sharp *a posteriori* error estimation of the true error in the target functional.

Figure 4.3 shows the meshes and the respective numerical solutions generated using both the *ad hoc* error indicator η_K^{adhoc} and the weighted error indicator $|\tilde{\eta}_K|$. In Figure 4.3(c) we see that the refinement due to the *ad hoc* indicator takes place in the neighborhood of the two discontinuities of the solutions resulting in a very good resolution of both jumps, see Figure 4.3(a). In contrast to that, the mesh produced using the weighted indicator, see Figure 4.3(d), is refined in the neighborhood of one of the jumps only, but it is totally unrefined at the position of the second jump. Thereby, the first jump in the corresponding numerical solution, see Figure 4.3(a), is properly resolved but the solution is very inaccurate near the second jump, thereby leading to very large residuals. However, these residuals do not contribute to the error in the target functional as the residuals in the error representation (3.8) are multiplied by local weights involving the dual solution that in turn are zero at the position of the second jump. This corresponds to the fact that the error of the solution in the neighborhood of the second jump does not belong to the domain of influence of the target functional and thus does not affect the error in the target functional.

Comparing the two meshes shown in Figure 4.3 it is obvious that, in terms of number of elements, the mesh produced using weighted indicators is more efficient for accurately computing the value of the target functional than the mesh produced using the *ad hoc* indicator. Indeed, although the number of elements of the mesh in Figure 4.3(d) is less than half of the mesh in Figure 4.3(d), the numerical solutions on the two meshes have approximately the same error in the target functional. But, we need to emphasise, that for generating the mesh produced using the weighting indicators, an approximate dual problem, see (3.14), must be solved numerically for each adaptive refinement step. Given, that for this example, both, the primal and the dual problem, are linear problems, and the dual problem is solved with higher order than the primal one, in our case $\tilde{z}_h \in V_h^2$ when $u_h \in V_h^1$, the computational cost of solving the dual problem represents a significant proportion of the total time required.

In order to provide an objective comparison we additionally measure the time needed to produce the meshes and solutions shown in Figure 4.3. This includes the computation of the primal solutions as well as the solution of the dual

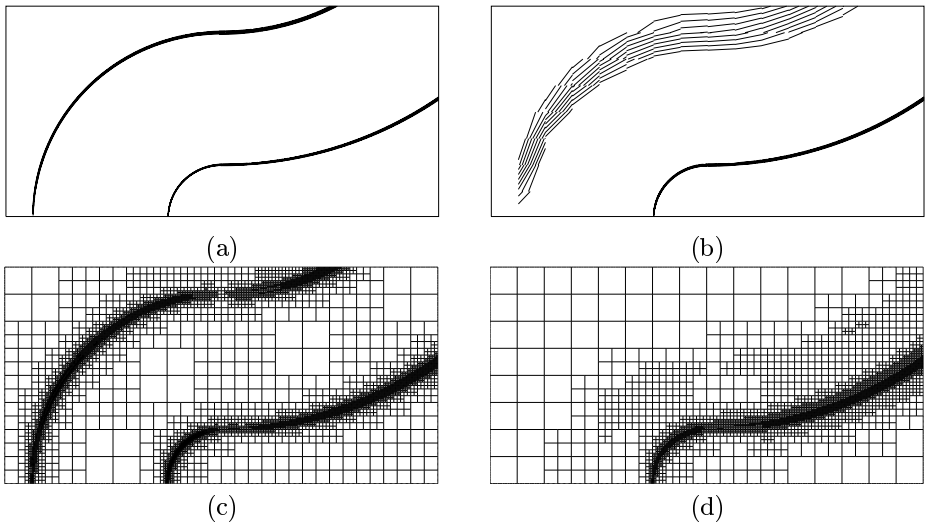


Figure 4.3: Weighted boundary integral for the linear advection equation: (a) Numerical solution on mesh (c) constructed using *ad hoc* error indicator with 11345 elements ($|J(e)| = 1.088 \times 10^{-7}$); (b) Numerical solution on mesh (d) constructed using weighted error indicator with 5087 elements ($|J(e)| = 1.096 \times 10^{-7}$).

problems needed for evaluating the weighted indicators. The present example gives a comparison of 58.6sec for the meshes produced using weighted indicators versus 63.4sec for the meshes produced using the *ad hoc* indicator.

4.2 1D inviscid Burgers equation

In this second example we consider the simplest scalar *nonlinear* hyperbolic problem, the unsteady 1D inviscid Burgers equation,

$$\partial_t u + u\partial_x u = 0.$$

Writing $F(u) = (\frac{1}{2}u^2, u)$ and $\nabla = (\partial_x, \partial_t)^T$ this problem may be represented as a stationary problem in the (x,t) plane by

$$\nabla \cdot F(u) = \partial_t u + \partial_x \left(\tfrac{1}{2}u^2\right) = 0,$$

cf. (2.2). We consider this problem on the (space-time) domain $(x,t) \in \Omega = [0,1]^2$ subject to the initial condition

$$u(x,0) = u_0(x) = \begin{cases} 1, & \text{for } x \leq 0.1, \\ -2.5x + 1.25, & \text{for } 0.1 < x \leq 0.3, \\ 0.5, & \text{for } 0.3 < x \leq 0.7, \\ 0, & \text{for } x > 0.7. \end{cases}$$

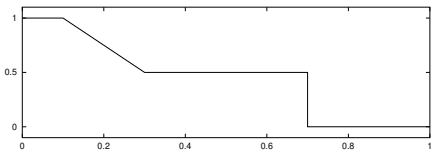
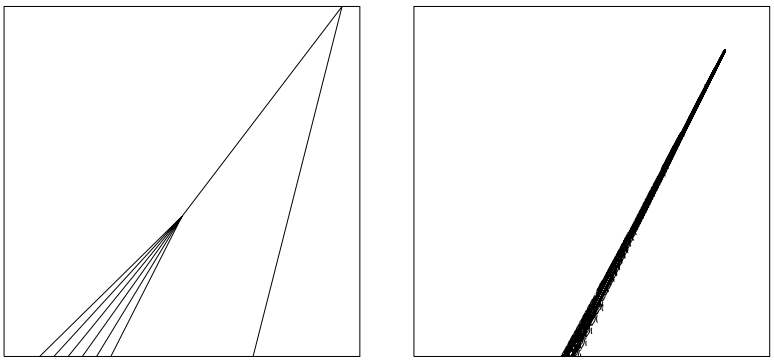


Figure 4.4: Burgers equation. Initial function $u_0(x)$.



(a) (b)

Figure 4.5: Point evaluation for the Burgers equation: (a) Isolines of the exact (primal) solution; (b) Isolines of the dual solution.

This function includes a linearly declined part and a discontinuity separated by constant states, see Figure 4.4.

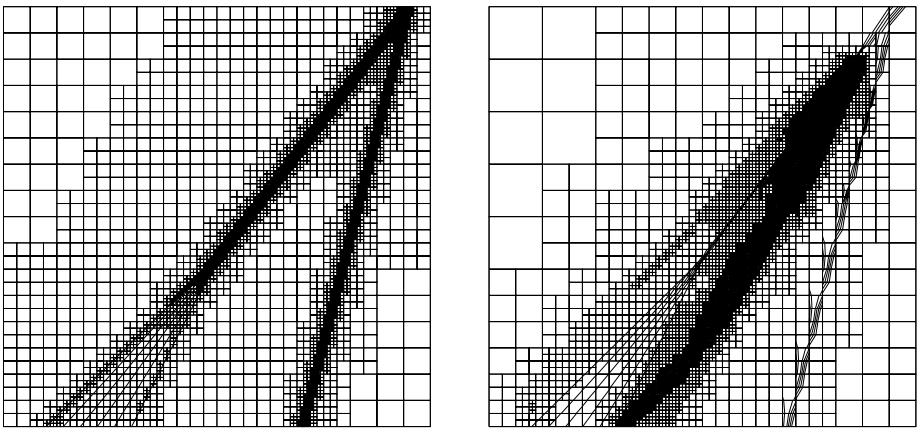
Figure 4.5(a) shows the structure of the exact solution: as time increases the slope of the linear declined part of the solution increases until a shock develops that moves to the right. This shock eventually merges with a second right moving shock that originates from the initial discontinuity.

Now we consider the point $x = 0.875$ at time $t = 0.875$ which is placed between the two shocks just a few moments before the two shocks merge. Assume that the task is to decide numerically whether one, two or none of the shocks have already crossed this point at this specific time and calculate the value of the solution at this point to high accuracy, i.e. we take the target functional to be the evaluation of the solution at the point of interest, i.e.

$$J(u) = u(x = 0.875, t = 0.875);$$

thereby the true value of the functional is given by $J(u) = 0.5$.

In Table 4.1, we demonstrate the performance of the adaptive algorithm. Here, we show the number of elements, the true error in the target functional, $J(e)$, the computed error representation $\tilde{\mathcal{E}}_\Omega = \sum_K \tilde{\eta}_K$, the computed Type I a



(a) (b)

Figure 4.6: Point evaluation for the Burgers equation problem. (a) Mesh constructed using *ad hoc* error indicator with 25603 elements ($|J(e)| = 8.605 \times 10^{-6}$); (b) Mesh constructed using weighted error indicator with 18001 elements ($|J(e)| = 9.316 \times 10^{-12}$).

posteriori error bound $\tilde{\mathcal{E}}_{|\Omega|} = \sum_K |\tilde{\eta}_K|$ and the effectivity indices $\theta_1 = \tilde{\mathcal{E}}_\Omega / J(e)$ and $\theta_2 = \tilde{\mathcal{E}}_{|\Omega|} / |J(e)|$. We see that initially on the coarsest mesh the quality of the computed error representation formula $\tilde{\mathcal{E}}_\Omega$ is poor, in the sense that θ_1 is not close to one; however, as the mesh is refined the effectivity index θ_1 approaches unity. On the other hand, we see that the Type I error bound $\tilde{\mathcal{E}}_{|\Omega|}$ is not sharp, in the sense that it largely overestimates the true error in the computed functional. We recall that the Type I error bound $\tilde{\mathcal{E}}_{|\Omega|}$ was derived from the error representation formula by simply employing the triangle inequality. In fact, the loss of sharpness of $\tilde{\mathcal{E}}_{|\Omega|}$ is attributed to the loss of inter–element cancellation of the local terms $\tilde{\eta}_K$, when the triangle inequality is employed. Thereby, it is clear that any further bounding performed *en route* to deriving a Type II *a posteriori* estimate will further adversely affect the quality of the computed error bound in the sense that the size of the resulting effectivity index will be even larger than θ_2.

In Figures 4.6(a) and (b) we show the solutions on meshes produced using the *ad hoc* and the weighted indicators, respectively. We see that in Figure 4.6(b) there is almost no refinement at the position of the two shocks. Both shocks and also the margin of the linear part of the solution are not well resolved. Most of the refinement takes place upstream of and in the neighborhood of the point of interest in an area that coincides with the support of the dual solution, see Figure 4.5(b). The shocks in the neighborhood of this point are resolved sufficiently enough so that the error at these shocks does not affect the solution at the point of interest.

# el.	$J(e)$	$\tilde{\mathcal{E}}_\Omega$	θ_1	$\tilde{\mathcal{E}}_{\lvert\Omega\rvert}$	θ_2
1024	9.625e-02	8.457e-03	0.09	3.061e-01	3.18
1558	6.151e-02	7.822e-02	1.27	2.725e-01	4.43
2551	-9.866e-03	-1.106e-02	1.12	6.092e-02	6.17
3985	-3.016e-04	-3.756e-04	1.25	2.814e-03	9.33
6328	-2.318e-07	-2.695e-07	1.16	3.917e-06	16.90
10558	-4.377e-10	-4.448e-10	1.02	1.599e-08	36.54
18001	-9.316e-12	-9.443e-12	1.01	8.448e-11	9.07

Table 4.1: Point evaluation for the Burgers equation: Adaptive algorithm.

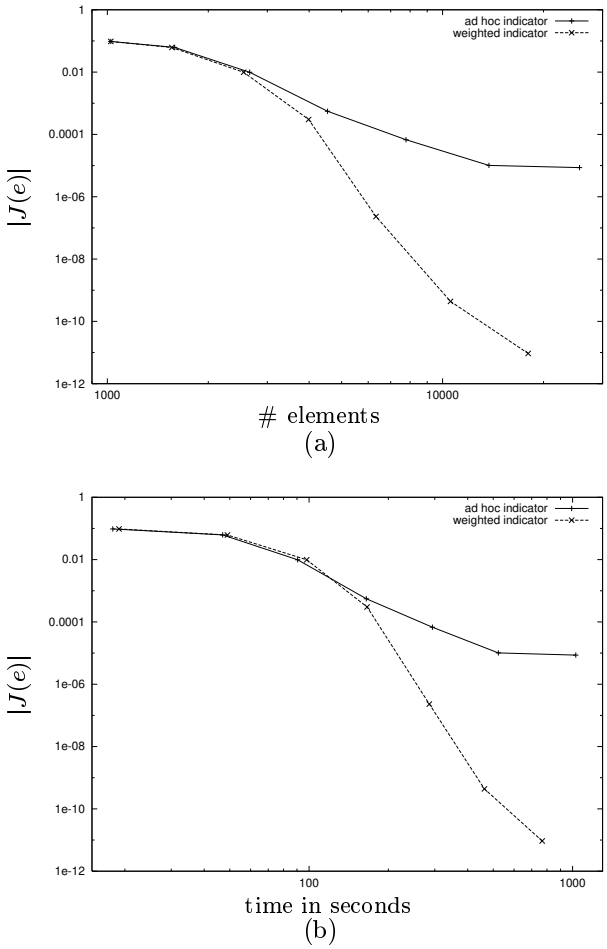


Figure 4.7: Point evaluation for the Burgers equation problem. (a) Error in $J(\cdot)$ vs. number of elements; (b) Error in $J(\cdot)$ vs. time in seconds.

Finally, in Figure 4.7, we compare the true error in the computed functional $J(\cdot)$ using the two mesh refinement strategies. Here, the true error $J(e)$ is plotted against the number of elements in Figure 4.7(a) and against the time needed for solving the problems in Figure 4.7(b). The time measurement includes the time required for solving the primal problems and for evaluating the error indicators as well as the time needed for the computation of the approximate dual solution $\tilde{z}_h$ and the evaluation of the approximation error representation when the meshes are produced using the weighted indicators.

Here, we clearly observe the superiority of the weighted *a posteriori* error indicators; on the final mesh the true error in the linear functional is six orders of magnitude smaller than $|J(e)|$ computed on the sequence of meshes produced using the *ad hoc* indicator.

4.3 Buckley-Leverett equation

The Buckley-Leverett equation [12] represents a simple model for *two phase flow* in a porous medium. An application is the simulation of an oil reservoir. The equation is given by

$$\partial_t u + \partial_x f(u) = 0,$$

with the *non-convex* flux function

$$f(u) = \frac{u^2}{u^2 + a(1-u)^2}.$$

The solution to Riemann problems for scalar equations with *convex* fluxes, like Burgers equation, includes either a shock or a rarefaction wave. Equations with non-convex fluxes, like the Buckley-Leverett equations, might involve both. A standard example is the Riemann problem

$$u(0,x) = u_0(x) = \begin{cases} 1 & \text{for } x \leq 0, \\ 0 & \text{for } x > 0, \end{cases}$$

modeling pure water ($u = 1$) for $x < 0$ and pure oil ($u = 0$) for $x > 0$ at initial time $t = 0$. The Buckley-Leverett equation simulates the replacement of the oil by water that is pumped from the left. This leads to a shock front at the interface between the oil and the water, followed by a rarefaction wave, that indicates for the time $t > 0$ a specific proportion of water and oil in the porous medium. The rarefaction wave in $x > 0$ exists for all time $t > 0$, hence $u < 1$ for all $(x,t) \in \mathbb{R}^+ \times \mathbb{R}^+$, simulating the effect that the total amount of oil may never be recovered by this 'secondary recovery' technique. The exact solution to this problem can be determined analytically, see Appendix B.1 for more details.

We choose the computational domain to be $(x,t) \in \Omega = [-1,1] \times [0,1]$, i.e. we compute the Riemann problem starting at the initial time $t = 0$. Here, the target functional is taken to be the value of the solution at point $x = 0.5$ at final time $t = 1$, i.e.

$$J(u) = u(0.5, 1).$$

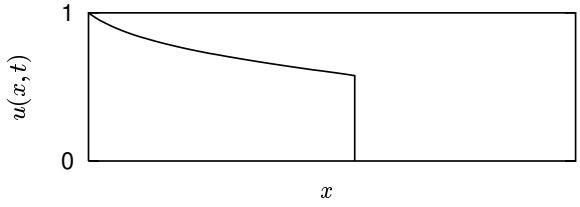


Figure 4.8: Solution to Riemann problem of the Buckley-Leverett equation.

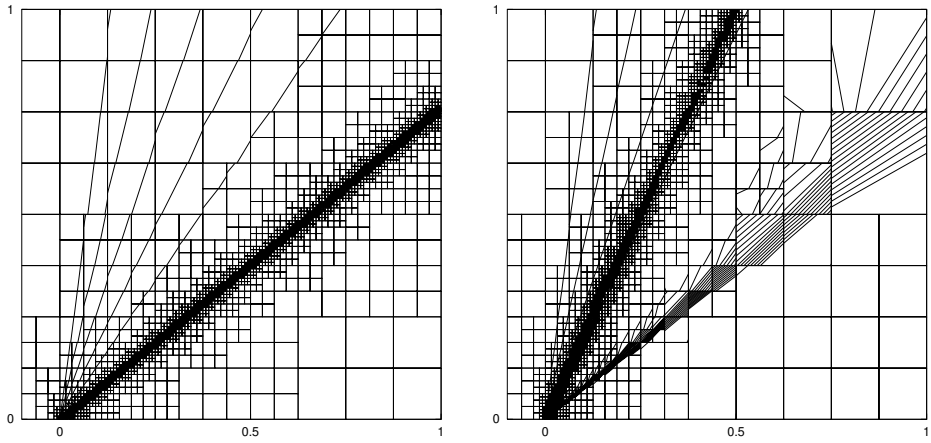


Figure 4.9: Point evaluation for the Riemann problem of the Buckley–Leverett equation. (a) Mesh constructed using *ad hoc* error indicator with 4532 elements ($|J(e)| = 4.064 \times 10^{-4}$); (b) Mesh constructed using weighted error indicator with 3845 elements ($|J(e)| = 1.953 \times 10^{-5}$).

The dual solution corresponding to this point value consists of a 'spike' originating from the point of interest. This spike is transported in the opposite direction to the characteristics within the support of the rarefaction wave until it is 'squeezed' into the initial discontinuity of the primal problem. No matter how far the mesh is refined in the neighborhood of the point $(0, 0)$ where the discontinuity in the primal problem enters Ω, the dual solution can never be numerically resolved there. This explains why for this example the approximate error representation does not provide as good error estimation as for the previous example. Nevertheless, the mesh, see Figure 4.9(b), produced using the weighted indicators turns out to be much more efficient than the mesh, see Figure 4.9(a), produced using the *ad hoc* indicators.

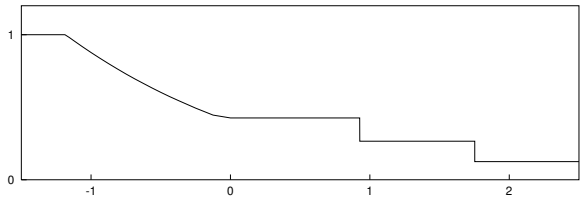


Figure 4.10: 1D Euler equation, shock-tube problem: Density $\varrho(x, 1)$.

4.4 1D compressible Euler equation

In this section we consider the one dimensional time-dependent Euler equations

$$\partial_t \begin{pmatrix} \varrho \\ \varrho v \\ e \end{pmatrix} + \partial_x \begin{pmatrix} \varrho v \\ \varrho v^2 + p \\ v(e+p) \end{pmatrix} = 0, \tag{4.3}$$

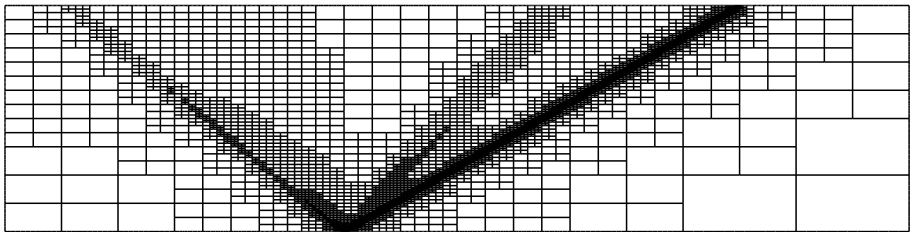
where ϱ, v and e represent the density, the velocity vector and the total energy per unit volume, respectively, see Appendix A.1 for more details. In particular, we look at the Riemann problem with the following initial condition

$$u(0, x) = u_0(x) = \begin{cases} u_L, & \text{if } x < 0, \\ u_R, & \text{if } x > 0, \end{cases} \tag{4.4}$$

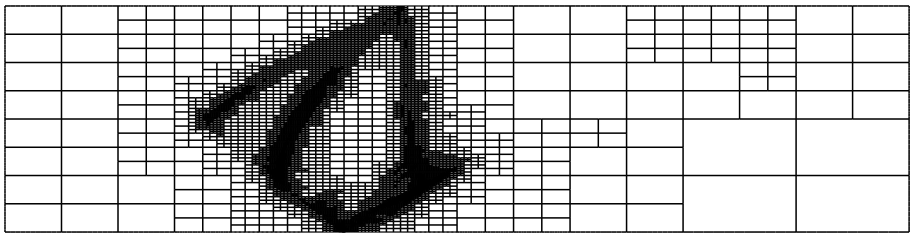
with $(\varrho_L, v_L, p_L) = (1, 0, 1)$ and $(\varrho_R, v_R, p_R) = (0.125, 0, 0.1)$. The exact solution to this Riemann problem, that is also referred to as *Sod's test problem* or *shock–tube problem*, is known and is given in analytical form. The solution includes a left rarefaction wave, a contact discontinuity and a right shock, see Figure 4.10 and cf. Appendix B.2 for a full description.

Assume that we are not interested in the whole solution but that we only are particularly interested in the height of the intermediate state between the contact discontinuity and the tail of the rarefaction wave. Hence we take the target functional to be $J(u) = \varrho(0.25, 1)$: the value of the solution at a point inside this state. The question is, to what extend the features of the solution like the shock and contact discontinuity must be resolved to obtain an exact value for this intermediate state.

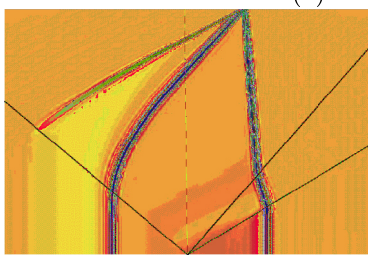
Figure 4.11(a) shows the mesh produced using the *ad hoc* indicators. As expected, most of the refinement takes place at the position of the shock. Furthermore, there is some refinement in the contact discontinuity and at the head of the rarefaction wave. In comparison to that in Figure 4.11(b) we show the mesh produced using the weighted indicators. This mesh shows some kind of combination of features of the primal and the dual solution. The dual solution, see Figure 4.11(c), corresponding to this point evaluation consists of 'spikes' that are transported in the opposite direction of the three characteristics, corresponding to the eigenvalues v and $v \pm c$, where $c = \sqrt{\gamma p/\varrho}$ denotes the speed of sound.



(a)



(b)



(c)

Figure 4.11: Point evaluation of the shock-tube problem for the 1D Euler equation: (a) Mesh constructed using *ad hoc* error indicator with 12578 elements ($|J(e)| = 3.156 \times 10^{-5}$); (b) Mesh constructed using weighted error indicator with 10310 elements ($|J(e)| = 6.630 \times 10^{-6}$); (c) Dual solution.

4.5 2D compressible Euler equations

Finally, we are concerned with the steady two–dimensional compressible Euler equations, see (2.2) and (2.4). In the following, we give a collection of several different problems governed by these equations, including smooth solutions and solutions with shocks like, for example, flows around airfoils. These problems will be regarded in combination with several different target functionals such as point evaluations and drag and lift coefficients. In the following, we give a short overview of the examples.

4.5.1 Ringleb flow problem. Computation of the horizontal force component exerted on the channel walls. Comparison of meshes produced using *ad hoc* and weighted indicator and with meshes specifically designed by hand. Comparison with mesh produced by weighted indicators corresponding to a different target quantity, namely a point evaluation.

4.5.2 Supersonic flow past a wedge including an inclined shock. Computation of density values in front and behind the shock. Consequences on the quality of the approximate error representation. Comparison of meshes produced using *ad hoc* and weighted indicator.

4.5.3 Sub-, trans- and supersonic flows around airfoils. Computation of drag and lift coefficients and single pressure point values on profile. For a more detailed overview of these examples see beginning of Section 4.5.3.

4.5.1 Ringleb flow problem

First, we consider the *Ringleb flow* problem, cf. [47], that we already considered in Section 3.6. The solution to this problem is smooth and represents a transonic flow in a channel, see the computational domain in Figure 4.12. The left and right boundaries may be considered as consisting of reflective walls, the bottom and the top boundary are inflow and outflow boundaries, respectively. All boundaries are aligned to the streamlines in the 'hodograph plane' variables, such that the solution to this smooth and irrotational problem is known and may be obtained via hodograph transformation, see [13] or Appendix B.5, for example.

Horizontal force component

At first we select the target functional to be the horizontal force component exerted on the channel walls. More precisely, we choose

$$J(u) = \int_{\text{wall}} (\psi \cdot n) p \, ds, \tag{4.5}$$

where $\psi = (1, 0)$, n denotes the unit outward normal vector to the boundary and p is the pressure; in this case, the true value of the functional on the exact geometry is given by $J(u) = -0.5744441759095$. We note that in Section 3.6

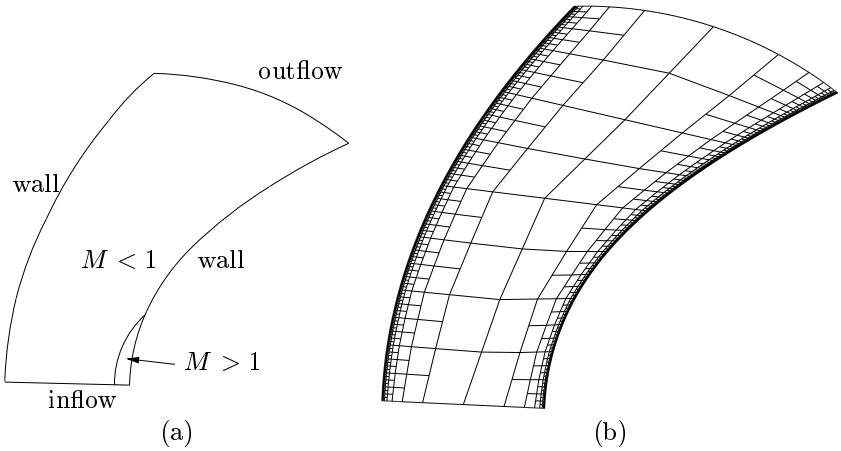
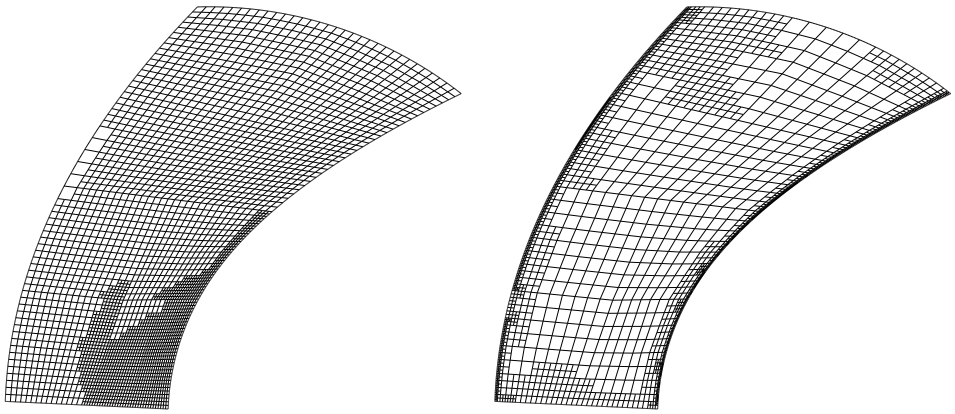


Figure 4.12: (a) Geometry of the Ringleb flow problem; (b) Mesh constructed by refining elements near the channel walls with 3056 elements; handmade mesh for computing the horizontal force of the flow exerted on the wall, see (4.5), with $|J(u) - J(u_h)| = 2.789 \times 10^{-5}$.

we already considered this problem subject to the same target functional. In Table 3.1 we already showed the performance of our adaptive algorithm and *a posteriori* error estimation applied to this problem.

Here, we now show the meshes produced using both the *ad hoc* error indicator η_K^{adhoc} and our weighted error indicator $|\tilde{\eta}_K|$, see Figures 4.13(a) and 4.13(b), respectively. We see that the mesh constructed using η_K^{adhoc} is largely concentrated in the region near the supersonic region at the right–hand wall, with further almost uniform refinement of the rest of the computational domain. In contrast to that the mesh constructed using the weighted error indicator $|\tilde{\eta}_K|$, see Figure 4.13(a), is concentrated near the walls of the channel. This in fact is due to the dual solution being large in the vicinity of the solid channel walls.

Finally, in Figure 4.14 we compare the true error in the computed functional $J(\cdot)$ using the two mesh refinement strategies. As before, we clearly see that our weighted *a posteriori* error indicator produces more economical meshes than when the traditional *ad hoc* error indicator is employed; on the final mesh the true error in the computed functional is almost two orders of magnitude smaller when the former error indicator is used. Additionally, in Figure 4.14 we also compare the performance of our weighted *a posteriori* error indicator $|\tilde{\eta}_K|$ with a mesh refinement algorithm which only refines elements adjacent to the channel walls, cf. Figure 4.12(b). Given the choice of the target functional, and the structure of the meshes produced using $|\tilde{\eta}_K|$, one may expect that this approach should work quite well without the need to solve the dual problem. However, we see that after an initial reduction in the error in the computed functional $J(\cdot)$, $|J(u) - J(u_h)|$ becomes $\mathcal{O}(1)$ under additional mesh refinement. This is attributed to the fact that while it is important to refine the mesh near the



(a) (b)

Figure 4.13: Ringleb flow problem. Horizontal force. (a) Mesh constructed using *ad hoc* error indicator with 3056 elements ($|J(u) - J(u_h)| = 2.1622 \times 10^{-5}$ for $J(u)$ as in (4.5)); (b) Mesh constructed using dual error indicator with 2534 elements ($|J(u) - J(u_h)| = 1.7504 \times 10^{-7}$).

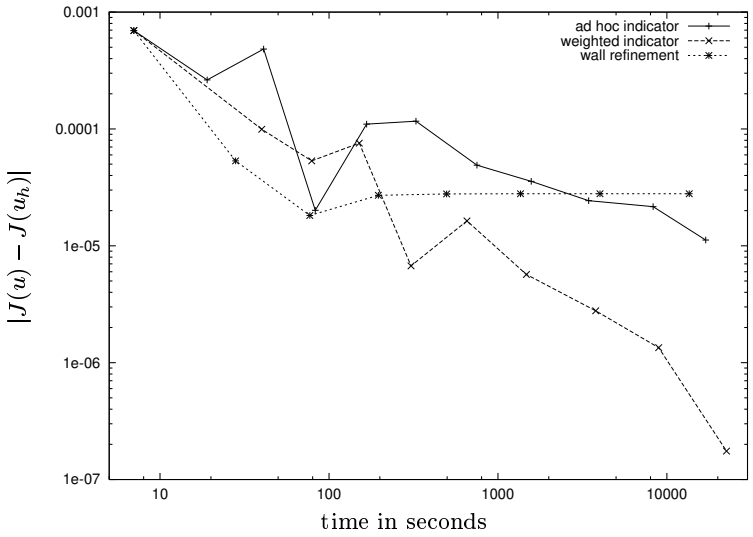


Figure 4.14: Ringleb flow problem. Horizontal force. Convergence of $|J(u) - J(u_h)|$ using the weighted and *ad hoc* error indicators, as well as simply refining elements near the channel walls.

# el.	$J(e)$	$\tilde{\mathcal{E}}_\Omega$	θ_1
32	-5.082e-04	-4.769e-04	0.94
53	-9.473e-05	-8.138e-05	0.86
98	-6.732e-05	-6.085e-05	0.90
164	-3.910e-05	-3.613e-05	0.92
257	-2.339e-05	-2.122e-05	0.91
425	-1.226e-05	-1.131e-05	0.92
710	-3.856e-06	-3.539e-06	0.92
1205	-5.693e-06	-5.540e-06	0.97
1961	-3.593e-06	-3.544e-06	0.99
3323	-1.261e-06	-1.240e-06	0.98

Table 4.2: Ringleb flow problem, $J(u) = \rho(-0.4, 2)$. Data of adaptive refinement by weighted indicators $|\tilde{\eta}_K|$.

channel walls in order to reduce the error in the target functional $J(\cdot)$, some mesh refinement is also required in the interior of the computational domain. Here, the weighted *a posteriori* error indicators $|\tilde{\eta}_K|$ give us information about not only where to refine the mesh, but also by how much; this leads to the consistent reduction in $|J(u) - J(u_h)|$ observed in Figure 4.14 as the finite element mesh is adaptively refined.

Point evaluation

As second and more challenging test case we choose the target functional to be

$$J(u) = \rho(-0.4, 2).$$

We note that this target functional is singular in the sense that it leads to a considerably rough dual solution that mainly consists of a single spike transported backwards with respect to the direction of the flow. Given that this dual solution is obviously hard to approximate numerically it is remarkable that the approximate error representation $\tilde{\mathcal{E}}_\Omega$ based on the numerical dual solutions gives an error estimation in terms of the considered point value that is very close to the exact error, cf. Table 4.2.

In Figure 4.15 we show the adaptive mesh produced using the weighted indicators. Comparing this mesh with the mesh in Figure 4.13(b) we observe that while the underlying (primal) problem has remained the same, the different choice of a target functional selected, point evaluation vs. horizontal force exerted on the walls, leads to a different dual solution and hence to a completely different mesh.

In contrast, the *ad hoc* error indicator has no information concerning the target functional of interest, and will produce the same mesh irrespective of the choice of $J(\cdot)$, see Figure 4.13(a).

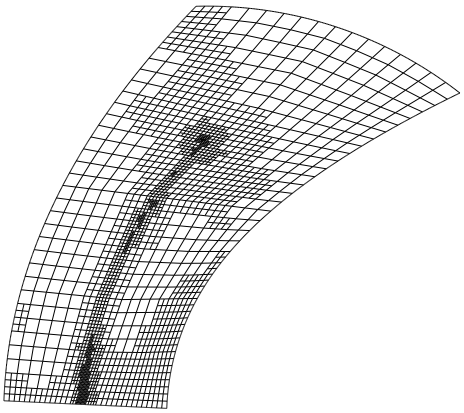


Figure 4.15: Ringleb flow problem, $J(u) = \rho(-0.4, 2)$. Mesh constructed using weighted indicator.

4.5.2 Supersonic flow past a wedge

In this example we study the formation of an oblique shock when supersonic flow is deflected by a sharp object or wedge. Here, we consider a Mach 3 flow with inflow density $\rho = 1$ and pressure $p = 1$, over a compression corner of angle α; this leads to the development of a shock wave at an angle β, cf. Figure 4.16. By employing the Rankine–Hugoniot jump conditions, the analytical solution to this problem for a given α may be determined , see Appendix B.4 for details. Here, we select the wedge angle $\alpha = 9.5^\circ$, cf. [11]; thereby, the angle of the shock is given by $\beta = 26.9308^\circ$. Furthermore, the true solution on the left– and right–hand side of the shock, in terms of conserved variables $(\rho, \rho v_1, \rho v_2, e)$, are given by

$$u_{\text{left}} = \begin{pmatrix} 1 \\ 3.5496 \\ 0 \\ 8.8 \end{pmatrix} \quad \text{and} \quad u_{\text{right}} = \begin{pmatrix} 1.6180 \\ 5.2933 \\ 0.8858 \\ 13.8692 \end{pmatrix},$$

respectively.

In Table 4.3, we show the performance of our adaptive algorithm when the target functional is chosen to be the value of the density just in front of the shock; more precisely, here

$$J(u) = \rho(5, 2.05).$$

As in the previous examples, we again see that the quality of the computed error representation formula (3.16) is quite poor on coarse meshes but θ_1 tends to unity, as the mesh is adaptively refined. Furthermore, the resulting Type I *a posteriori* bound $\tilde{\mathcal{E}}_{|\Omega|}$ overestimates the true error in the computed target functional by about one–two orders of magnitude.

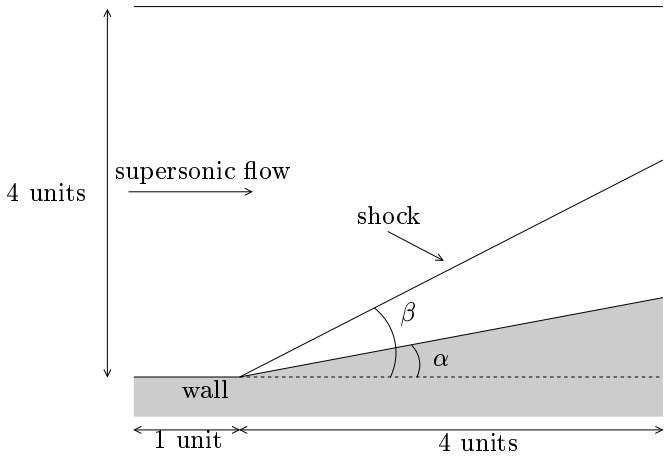


Figure 4.16: Geometry for the supersonic compression corner.

# el.	$J(u) - J(u_h)$	$\tilde{\mathcal{E}}_\Omega$	θ_1	$\tilde{\mathcal{E}}_{\lvert\Omega\rvert}$	θ_2
20	-3.428e-01	2.771e-02	-0.08	8.565e-02	0.25
35	-3.194e-01	-4.315e-02	0.14	8.134e-02	0.25
65	-3.320e-01	-6.235e-02	0.19	7.675e-02	0.23
128	-2.666e-01	-1.356e-01	0.51	1.819e-01	0.68
227	-2.164e-01	-1.665e-01	0.77	2.142e-01	0.99
386	-3.272e-03	-8.919e-02	27.26	2.094e-01	63.99
677	1.686e-02	1.077e-02	0.64	5.179e-02	3.07
1175	-5.475e-03	-5.887e-03	1.08	2.130e-02	3.89
1964	-3.383e-04	-3.635e-04	1.07	3.671e-03	10.85
3293	-4.208e-06	-3.837e-06	0.91	3.000e-04	71.30
5516	4.551e-07	5.056e-07	1.11	2.079e-05	45.67

Table 4.3: Supersonic compression corner, point evaluation of the density in front of the shock: Data of adaptive refinement by weighted indicators $|\tilde{\eta}_K|$.

# el.	$J(u) - J(u_h)$	$\sum_K \tilde{\eta}_K$	θ_1	$\sum_K \lvert\tilde{\eta}_K\rvert$	θ_2
20	2.548e-01	3.952e-02	0.16	9.546e-02	0.37
35	2.652e-01	-1.728e-02	-0.07	9.185e-02	0.35
68	2.226e-01	-1.148e-02	-0.05	4.927e-02	0.22
134	1.723e-01	6.407e-02	0.37	1.254e-01	0.73
209	1.259e-01	1.007e-01	0.80	1.369e-01	1.09
389	6.044e-02	9.786e-02	1.62	1.180e-01	1.95
650	-1.453e-02	-1.538e-02	1.06	5.050e-02	3.48
1124	5.425e-04	4.530e-03	8.35	3.220e-02	59.36
1997	4.470e-04	3.006e-03	6.73	2.733e-02	61.15
3395	2.888e-04	1.405e-03	4.86	2.401e-02	83.15

Table 4.4: Supersonic compression corner, point evaluation of the density behind the shock. Data of adaptive refinement by weighted indicators $|\tilde{\eta}_K|$.

We end this example by considering the more interesting case of estimating the value of the density just behind the shock; to this end, we select

$$J(u) = \rho(5, 2.01).$$

The performance of the adaptive algorithm in this case is presented in Table 4.4. Here, we see that while again, the Type I error bound $\tilde{\mathcal{E}}_{|\Omega|}$ overestimates the true error by about one–two orders of magnitude, now also the ratio θ_1 of the (computed) error representation formula $\tilde{\mathcal{E}}_\Omega$ and $J(u) - J(u_h)$ no longer tends to one as the finite element mesh is refined; indeed, here we see that (ignoring the first few refinement steps) θ_1 lies in the interval $(0.8, 8.35)$. This degradation in the quality of the computed error representation formula when the point of evaluation lies behind the shock is attributed to the effects of the linearisation error and the error in the approximation to the Fréchet derivative of the semi-linear form $a(\cdot,\cdot)$ (with respect to its first argument), becoming large in the vicinity of the shock. In this case, the weighting terms (cf. (4.1)) involving the dual solution are poorly approximated in the neighbourhood of the shock, where the residual terms $R(u_h)$ and $r(u_h)$, and the shock capturing term $\varepsilon\nabla u_h$ are large, thereby leading to a poor approximation of the true error in the target functional $J(\cdot)$. We note that this does *not* occur if the target functional is selected upstream of the shock, since then the dual solution no longer interacts with the shock; thereby, in the neighbourhood of the shock, the weighting terms are negligibly small, and the computed error representation formula is extremely good, cf. above. Thus, in order to guarantee the quality of the computed error representation formula, additional computational work is required to reduce the effects of the linearisation error, the error stemming from the approximation of $a'[u_h](\cdot, v)$, $v \in V$ fixed, and the error generated in the numerical approximation of the (approximate) dual problem; this is beyond the scope of this work, but will be considered in [25].

Notwithstanding this reduction in the quality of the computed error repre-

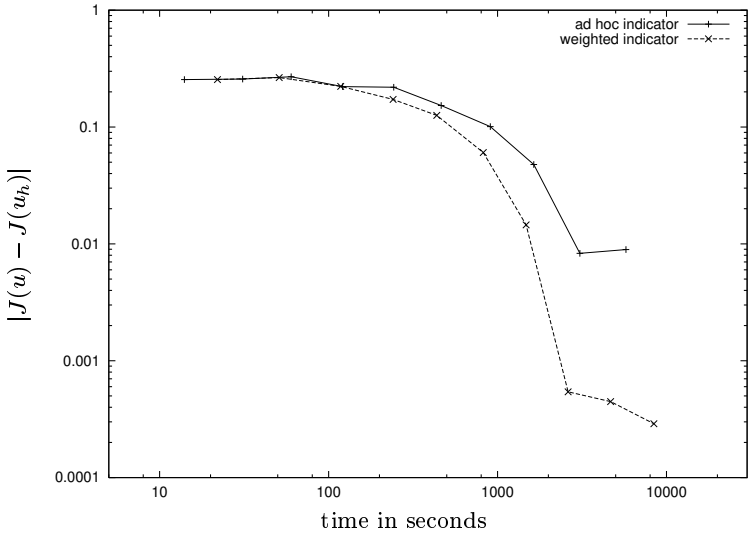


Figure 4.17: Supersonic compression corner, point evaluation of the density behind the shock. Convergence of $|J(u) - J(u_h)|$ using the weighted and *ad hoc* error indicators.

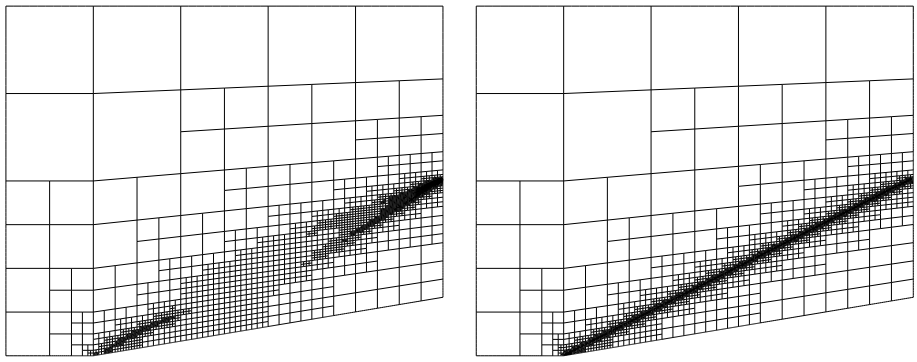
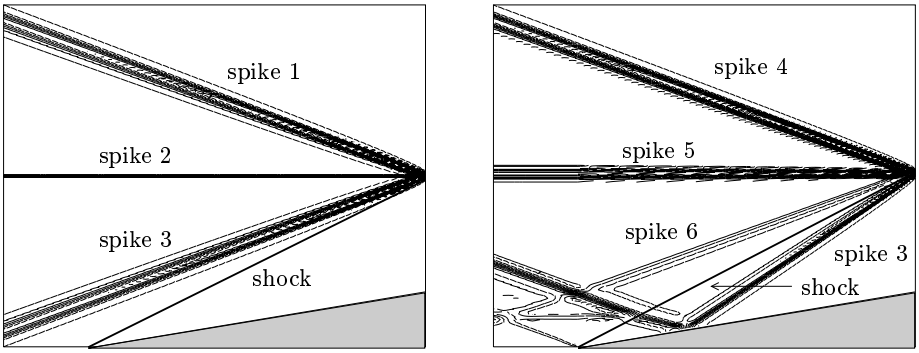


Figure 4.18: Supersonic compression corner, point evaluation of the density behind the shock (a) Mesh constructed using dual error indicator with 3395 elements ($|J(u) - J(u_h)| = 2.888 \times 10^{-4}$); (b) Mesh constructed using *ad hoc* error indicator with 3821 elements ($|J(u) - J(u_h)| = 8.938 \times 10^{-3}$).



(a) (b)

Figure 4.19: Supersonic compression corner: z_1 component of dual solution for the supersonic compression corner for point evaluation of the density (a) in front of shock (b) behind shock.

sentation formula when the point of interest lies behind the shock, the adaptively refined meshes generated by employing the weighted error indicator $|\tilde{\eta}_K|$ are significantly more efficient than those produced using the traditional *ad hoc* error indicator η_K^{adhoc}. Indeed, in Figure 4.17 we clearly observe the superiority of the former error indicator; on the final mesh the true error in the computed functional is over an order of magnitude smaller when the weighted error indicator is employed.

Finally, in Figure 4.18 we show the meshes produced using both error indicators. Here, we see that the mesh constructed using η_K^{adhoc} is concentrated in the neighbourhood of the shock. In contrast, the mesh produced using the weighted error indicator $|\tilde{\eta}_K|$ only refines the mesh in the vicinity of the top and bottom parts of the shock. To gain insight into the structure of this mesh, in Figures 4.19(a) and 4.19(b) we show the dual solutions corresponding to the point evaluation in front and behind the shock, respectively; for clarity, here we show the z_1 component of the dual solutions computed on globally refined meshes. The dual solution corresponding to the point evaluation of the density in front of the shock, see Figure 4.19(a), consists of three 'spikes', labelled 1, 2 and 3 in Figure 4.19(a), originating from the point of interest. These spikes are transported upstream along the characteristics corresponding to the three eigenvalues $\bar{v}$ and $\bar{v} \pm c$, with $\bar{v} = \sqrt{u^2 + v^2}$ denoting the velocity of the gas and $c = \sqrt{\gamma p/\rho}$ the speed of sound. We note that the support of this dual solution does not intersect the region of the computational domain where the shock in the primal solution is located. In contrast, the support of dual solution corresponding to the point evaluation of the density behind the shock, see Figure 4.19(b), now intersects the region containing the shock; here, we observe that z_1 has a rather complicated structure. The two upper spikes of the dual solution both cross the shock in the neighbourhood of the point of evaluation.

At their crossing points they again each split into a further three spikes. These six spikes correspond to the three pairs of spikes, labelled spikes 4, 5 and 6 in Figure 4.19(b); the two spikes in each pair cannot be distinguished on the resolution shown, as they are extremely close together. Spike 3, corresponding to the same spike in Figure 4.19(a), is reflected off the inclined wall and crosses the shock at its bottom part. A comparison of the dual solution in Figure 4.19(b) and the mesh in Figure 4.18(a) shows that the mesh has only been refined along the support of spikes 3 and 6 in the vicinity of the top part of the shock, and in the neighbourhood of the point where spike 3 crosses the bottom part of the shock. This demonstrates that it is not necessary to refine the entire shock in this example to gain an accurate evaluation of the target functional under consideration, but only those parts that influence the value of the target functional either by material transport (eigenvalue $\bar{v}$), or by information transported by the sound waves (eigenvalues $\bar{v} \pm c$). Indeed, as seen in Figure 4.17, the meshes produced using our weighted error indicator are much more economical for computing the value of the target functional than meshes produced using the *ad hoc* error indicator.

4.5.3 Flows around airfoils

This section is dedicated to the simulation of gas flows around airfoils. The focus of these computations will be the evaluation of some important and physically relevant quantities of the flows like the drag or lift coefficient of an airfoil immersed into the inviscid fluid as well as density or pressure point values at the leading edge of the airfoil. As in previous examples we assume that the two major aims of these simulations are, firstly, to compute the specific physical quantity of the flow to best accuracy and, secondly, to provide a sharp and reliable but not too expensive estimation of the error of the numerical solutions in terms of the quantity of interest.

We have tested many different problems including several different airfoils at several different flow velocities and angles of attack and subject to several different target functionals. However, we present only a small collection of test problems, in the following, including two different airfoils subject to subsonic, transonic and supersonic flow and three different target functionals such as the drag coefficient, lift coefficient and a pressure point value at the leading edge of an airfoil. A short overview of the three test problems are given in the following:

1. *Subsonic* smooth flow around the NACA0012 airfoil. The drag on meshes produced using weighted indicators converges monotone in contrast to a 'zig-zag' convergence for the *ad hoc* indicator. The approximate error representation is very sharp and can be employed for enhancing the accuracy of the computed quantity. The approximate error representation indicates that an accurate value of the lift is already attained on the coarsest mesh.

2. *Transonic* flow around the NACA0012 airfoil with small shock adjacent the airfoil. Computation of the drag coefficient. Despite the discontinuity of the solution the approximate error representation is still very sharp.

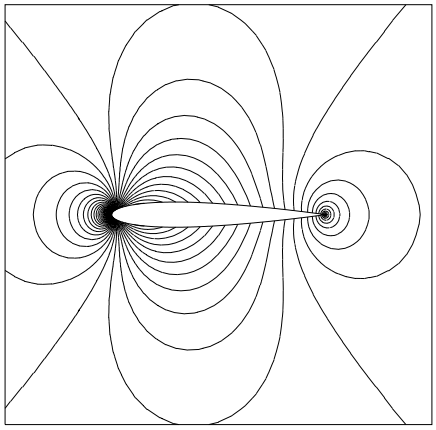


Figure 4.20: Subsonic NACA0012 flow: Mach number isolines $M = \frac{i}{100}$, $i = 1, 2, \ldots$.

3. *Supersonic* flow around the BAC3-11 airfoil. Computation of pressure value at leading edge. Meshes produced using weighted indicators are much more efficient than *ad hoc* refined meshes or handmade meshes that are specially designed for the computation of the specific pressure value. Due to the bow shock and the singular target functional the approximate error representation is not as good as in previous examples.

Subsonic flow around the NACA0012 airfoil

In this first example, we consider the subsonic flow around a NACA0012 airfoil, cf. Figure 2.22 in Section 2.10.1. The computational domain Ω is defined to be an annulus with inner boundary representing the geometry of the airfoil and the outer boundary consisting of a circle of radius 10 units, see Section 2.10. On the outer boundary, we prescribe a Mach 0.5 flow at a zero angle of attack, with farfield density $\rho = 1$ and pressure $p = 1$. The solution to this problem consists of a strictly subsonic flow, symmetric about the x–axis, cf. Figure 4.20.

In this example we are interested in computing the drag and lift coefficients defined by

$$J_{\text{drag}}(u) = \frac{2}{\bar{l}\bar{\rho}|\bar{v}|^2} \int_S (\psi_d \cdot n)\, p\, ds, \qquad J_{\text{lift}}(u) = \frac{2}{\bar{l}\bar{\rho}|\bar{v}|^2} \int_S (\psi_l \cdot n)\, p\, ds,$$

respectively. Here, S denotes the surface of the airfoil represented by $g^{\pm}$ defined in (2.54), $\bar{l}$ its chord length (equal to one), $\bar{v}$ and $\bar{\rho}$ denote the reference (or free-stream) velocity and density, respectively, and

$$\psi_d = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \psi_l = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

# el.	$J(u) - J(u_h)$	$\tilde{\mathcal{E}}_\Omega$	θ_1	$\tilde{\mathcal{E}}_{\lvert\Omega\rvert}$	θ_2
39	-3.902e-02	-2.729e-02	0.70	2.970e-02	0.76
66	-2.005e-02	-5.419e-03	0.27	1.035e-02	0.52
123	-1.339e-02	-1.540e-03	0.11	9.735e-03	0.73
210	-8.916e-03	-4.277e-03	0.48	9.144e-03	1.03
357	-4.212e-03	-3.237e-03	0.77	7.251e-03	1.72
582	-1.574e-03	-1.427e-03	0.91	3.542e-03	2.25
948	-5.494e-04	-5.264e-04	0.96	1.456e-03	2.65
1584	-2.270e-04	-2.238e-04	0.99	5.701e-04	2.51
2583	-9.428e-05	-9.288e-05	0.99	2.301e-04	2.44
4218	-4.758e-05	-4.573e-05	0.96	9.462e-05	1.99
6852	-2.303e-05	-2.082e-05	0.90	4.241e-05	1.84

Table 4.5: Subsonic NACA0012 flow. Adaptive algorithm for the evaluation of the drag coefficient.

where α denotes the angle of attack. We note that since the pressure p is derived from the conserved variables $(\rho, \rho u, \rho v, \rho E)$ using the equation of state (2.5), both target functionals $J_{\text{drag}}(\cdot)$ and $J_{\text{lift}}(\cdot)$ are nonlinear. Given that the angle of attack $\alpha = 0$, the true value of both target functionals $J_{\text{drag}}(\cdot)$ and $J_{\text{lift}}(\cdot)$ are known to be exactly zero.

Let us first consider the case of estimating the drag on the surface of the airfoil, i.e. when $J(\cdot) \equiv J_{\text{drag}}(\cdot)$. To this end, in Table 4.5 we show the performance of our adaptive algorithm; here we see that the quality of the computed error representation formula is very good, with $\theta_1 \approx 1$ even on relatively coarse meshes. Furthermore, the Type I *a posteriori* error bound $\tilde{\mathcal{E}}_{|\Omega|}$ is sharper for this smooth problem; here, $\tilde{\mathcal{E}}_{|\Omega|}$ overestimates the true error in the computed functional by about a factor 2–3, only.

The meshes produced using both our weighted error indicator $|\tilde{\eta}_K|$ and the *ad hoc* error indicator η_K^{adhoc} are shown in Figure 4.21. Here, we see that both meshes are fairly similar in character, in the sense that most of the refinement is concentrated in the neighbourhood of the leading and trailing edges of the airfoil. However, we see that the mesh designed by the weighted error indicator is also refined along the upper and lower surfaces of the airfoil. These extra regions of refinement are introduced as a result of the weighting terms present in the error indicator $|\tilde{\eta}_K|$, cf. (4.1). Indeed, in Figure 4.22, we see that while there is a singularity in the dual solution at the leading edge of the airfoil, which is transported upstream along the characteristic with eigenvalue $\bar{v}$, $\tilde{z}_1$ also has large gradients around the whole of the geometry.

In Figure 4.23 we compare the true error in the computed functional $J(\cdot)$ using the two mesh refinement strategies. First we note that here the convergence of $|J(u) - J(u_h)|$ on the sequence of meshes produced with the weighted error indicator is much 'smoother' than on the meshes designed with the *ad hoc* error indicator. Indeed, on finer meshes the error is even monotone allowing

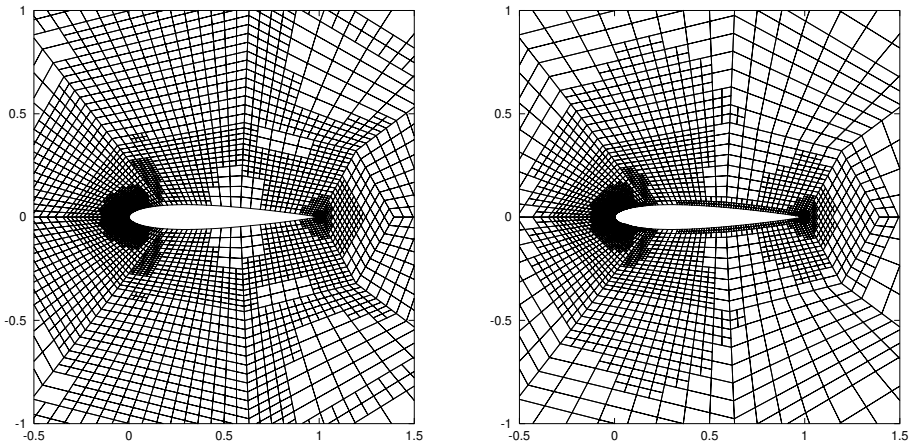


Figure 4.21: Subsonic NACA0012 flow. Computation of the drag coefficient. (a) Mesh constructed using *ad hoc* error indicator with 6567 elements ($|J(u) - J(u_h)| = 1.086 \times 10^{-4}$); (b) Mesh constructed using dual error indicator with 6852 elements ($|J(u) - J(u_h)| = 2.082 \times 10^{-5}$).

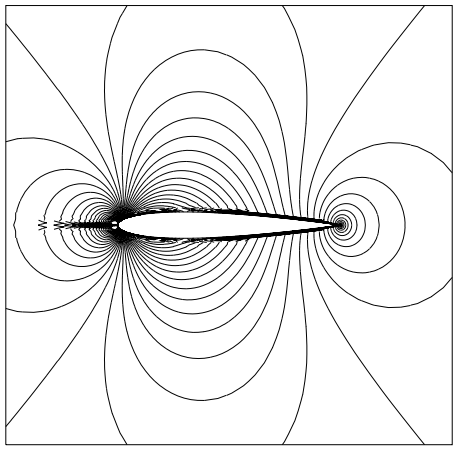


Figure 4.22: Subsonic NACA0012 airfoil. First component $\tilde{z}_1$ of the dual solution for the evaluation of the drag coefficient.

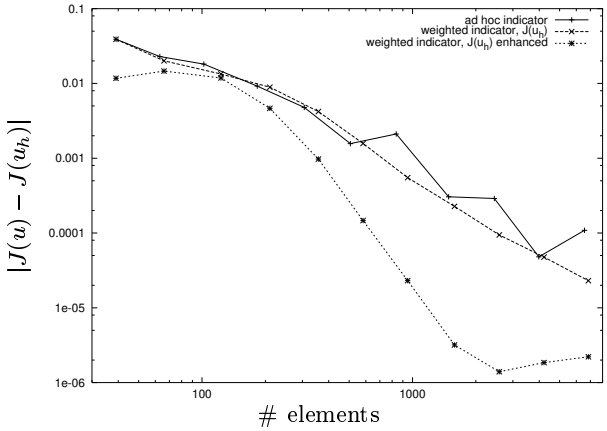


Figure 4.23: Subsonic NACA0012 flow. Computation of the drag coefficient. Convergence of $|J(u) - J(u_h)|$ using the weighted and *ad hoc* error indicators.

# el.	$J(u) - J(u_h)$	$\tilde{\mathcal{E}}_\Omega$	$\tilde{\mathcal{E}}_{\|\Omega\|}$
39	3.2e-16	-1.2e-14	1.7e-01

Table 4.6: Subsonic NACA0012 flow. Adaptive algorithm for the evaluation of the lift coefficient.

extrapolation of the value of the target functional in comparison to the 'zig–zag' convergence of $|J(u) - J(u_h)|$ on the meshes designed by η_K^{adhoc}. Again, as in the previous example, we see that our weighted *a posteriori* error indicators produce more economical meshes than when the traditional *ad hoc* error indicator is employed, though the difference is not as large as in previous examples. This is due to the fact that in this example the meshes are too similar in character. However, the accuracy of the computed quantity $J(u_h)$ can be enhanced by employing the approximate error representation $\tilde{\mathcal{E}}_\Omega$. Indeed, $\tilde{J}(u_h) = J(u_h) + \tilde{\mathcal{E}}_\Omega$ converges with higher order, cf. [34], than $J(u_h)$, see Figure 4.23.

Now we consider the case when $J(\cdot) \equiv J_{\text{lift}}(\cdot)$. The exact value of this target functional is known to be $J(u) = 0$ due to the flow being symmetric around the x–axis. Since the coarse computational mesh is symmetric, also the numerical solution on that mesh is symmetric yielding an error $J(e) \approx 0$ in the target functional. This we can see in Table 4.6 that shows the data on the coarsest mesh with 39 elements. Furthermore, we see that the approximate error representation is $\tilde{\mathcal{E}}_\Omega \approx 0$ indicating that the lift coefficient is correctly approximated already on the coarsest mesh and no further refinement is necessary. Note, that in contrast to that the Type I error bound $\tilde{\mathcal{E}}_{|\Omega|}$ is 14 orders of magnitude larger than $\tilde{\mathcal{E}}_\Omega \approx 0$.

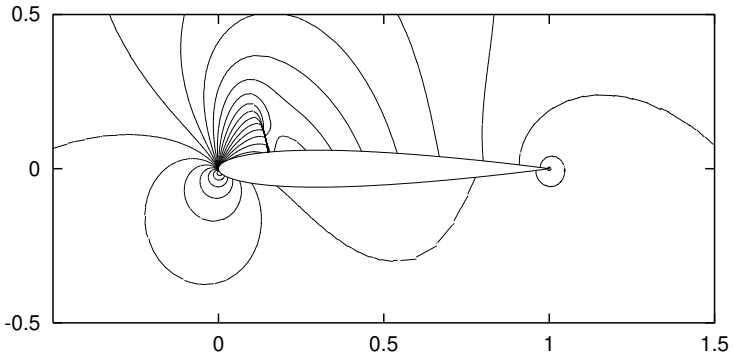


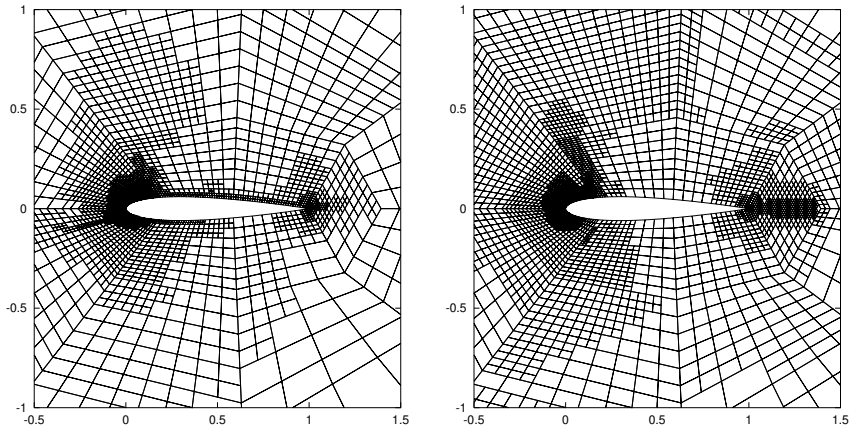
Figure 4.24: Transonic NACA0012 flow: Pressure isolines.

Transonic flow around the NACA0012 airfoil

In this example, we consider the transonic flow around a NACA0012 airfoil; here, we prescribe a Mach 0.6 flow at an angle of attack $\alpha = 5^\circ$, with farfield density $\rho = 1$ and pressure $p = 1$. The flow has a small supersonic region on the upper surface of the airfoil; as the flow slows to subsonic a small shock develops, cf. Figure 4.24. Here, we consider the evaluation of the drag coefficient on the surface of the airfoil, i.e. $J(\cdot) \equiv J_{\text{drag}}(\cdot)$. On the basis of a fine grid computation, the reference value of the functional is given by $J(u) = 6.88 \times 10^{-3}$.

Figure 4.25 shows the meshes generated using both error indicators $|\eta_K|$ and η_K^{adhoc}. Given that the structure of both meshes is very similar in the sense that they are both largely concentrated in the vicinity of the leading and trailing edges of the airfoil and in the neighborhood of the shock, it is remarkable that the error in the drag coefficient is approximately 40% smaller on the mesh produced using the weighted error indicator than on the mesh (of about 30% more elements) produced by η_K^{adhoc}. Looking at the meshes in more detail, we see that the former mesh is less refined in the neighborhood of the shock and the trailing edge than the latter; moreover it shows an additional line of refinement upstream of the airfoil and, more prominent, along the upper and lower surfaces of the airfoil. In Figure 4.26 we compare the error in the target functional using the two refinement strategies; at all refinement steps we observe the superiority of the weighted error indicators.

In Table 4.7 we collect the data of the adaptive algorithm when employing the weighted indicators. First we note that on all refinement steps the correct sign of the error is predicted by the computed error representation formula $\mathcal{E}_\Omega$. Furthermore, whereas on very coarse meshes the quality of $\mathcal{E}_\Omega$ is rather poor, in the sense that $\theta_1 = \mathcal{E}_\Omega/(J(u) - J(u_h))$ is noticeable smaller than one, we see that already after five refinement steps the computed error representation $\mathcal{E}_\Omega$ provides remarkably sharp error estimation resulting in effectivity indices $\theta_1 \in [0.91, 1]$.



(a) (b)

Figure 4.25: Transonic NACA0012 flow. (a) Mesh constructed using dual error indicator with 6087 elements ($|J(u) - J(u_h)| = 9.800 \times 10^{-5}$); (b) Mesh constructed using *ad hoc* error indicator with 7863 elements ($|J(u) - J(u_h)| = 1.656 \times 10^{-4}$).

# el.	$J(u) - J(u_h)$	$\tilde{\mathcal{E}}_\Omega$	θ_1
39	-7.065e-02	-2.624e-02	0.37
63	-5.586e-02	-1.889e-02	0.34
105	-3.938e-02	-1.772e-02	0.45
177	-1.964e-02	-1.351e-02	0.69
288	-7.871e-03	-6.190e-03	0.79
483	-3.131e-03	-2.859e-03	0.91
807	-1.234e-03	-1.183e-03	0.96
1347	-5.983e-04	-5.651e-04	0.94
2226	-3.193e-04	-2.916e-04	0.91
3639	-1.601e-04	-1.507e-04	0.94
6087	-9.800e-05	-9.802e-05	1.00

Table 4.7: Transonic NACA0012 flow. Adaptive algorithm for the evaluation of the drag coefficient.

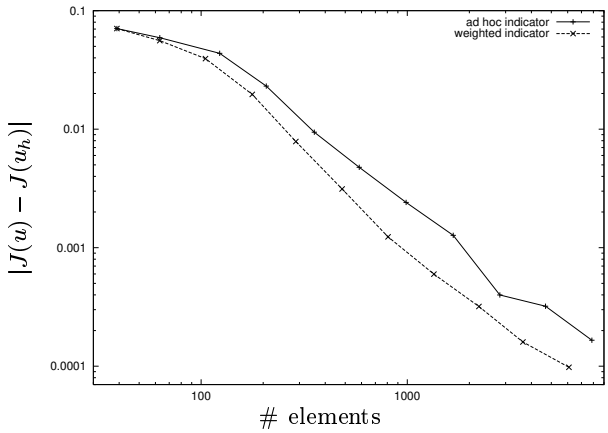


Figure 4.26: Transonic NACA0012 flow. Convergence of $|J(u) - J(u_h)|$ using the weighted and *ad hoc* error indicators.

Supersonic flow around the BAC3-11 airfoil

In this final example we study a supersonic flow around a BAC3-11 airfoil; this unsymmetric airfoil, see Figure 4.27, was originally specified in [2]. Here, we consider a Mach 1.2 flow at an angle of attack $\alpha = 5^\circ$, with inflow density $\rho = 1$ and pressure $p = 1$.

The solution to this problem includes two shocks: one located in front of the leading edge of the airfoil and one originating from the trailing edge; see Figure 4.28(a) and also Figure 4.29(b) which shows a mesh that is refined at the position of the two shocks. Here, Figure 4.28(a) shows the Mach 1 isolines of the solution; the Mach $M = 1$ isoline to the left of the airfoil indicates the position of the first shock. The $M = 1$ isolines that originate from the upper and lower surfaces of the airfoil represent the transonic lines of the flow. The flow left of the first shock is supersonic; it is simply the $M = 1.2$ flow prescribed on the inflow boundary of the computational domain. The flow in between the shock and the transonic lines is subsonic; we note that the leading edge of the airfoil is located within this subsonic part of the flow. Finally, the flow behind the transonic lines is supersonic again.

In this example we take the functional of interest to be the value of the pressure at the leading edge, i.e.

$$J(u) = p(0, 0),$$

cf. Figure 4.27. We note that, as in the case of estimating the drag and lift coefficients of a body immersed in an inviscid fluid, cf. Sections 4.5.3 & 4.5.3, this functional is nonlinear as it depends on the pressure. A computation on a fine mesh gives a reference value of $J(u) = 2.393$.

The structure of the dual solution $\tilde{z}$ corresponding to this point evaluation is

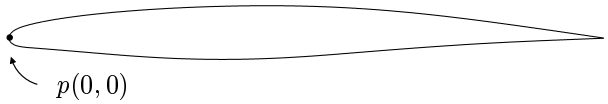


Figure 4.27: Profile of the BAC3-11 airfoil. Quantity of interest: pressure p at leading edge.

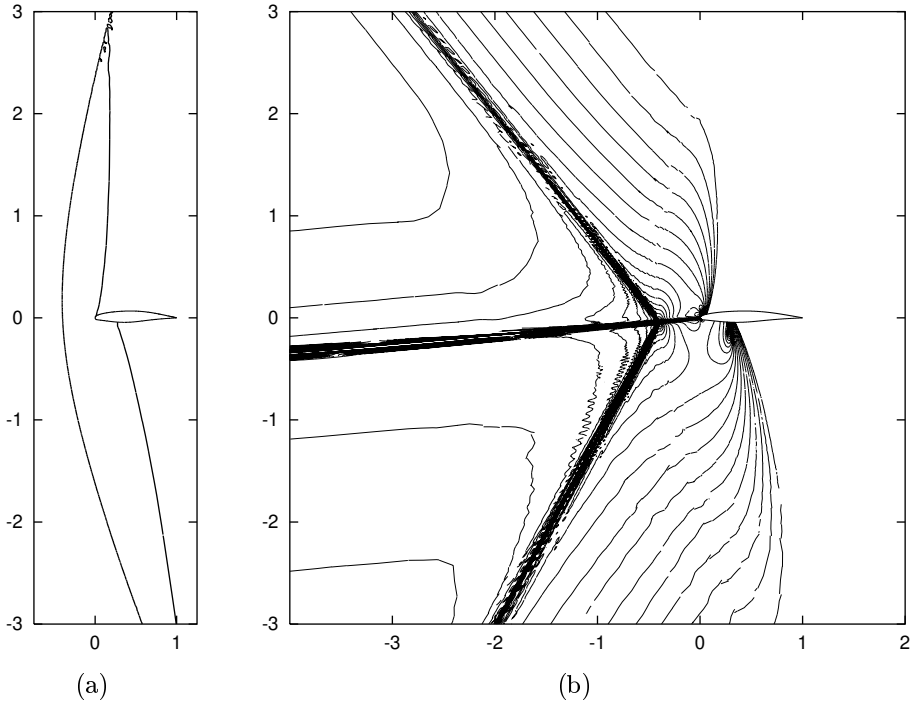


Figure 4.28: Supersonic BAC3-11 flow. (a) Mach 1 isolines of the primal solution; (b) $\tilde{z}_1$ isolines of dual solution.

# el.	$J(e)$	$\tilde{\mathcal{E}}_\Omega$	θ_1	$\tilde{\mathcal{E}}_{\lvert\Omega\rvert}$	θ_2
39	3.188e-01	2.773e-01	0.87	4.288e-01	1.35
63	2.313e-01	-1.501e-02	-0.06	2.003e-01	0.87
114	2.069e-01	7.275e-02	0.35	3.498e-01	1.69
192	7.398e-02	5.404e-02	0.73	2.680e-01	3.62
348	6.425e-02	2.695e-02	0.42	2.120e-01	3.30
609	2.876e-02	1.389e-02	0.48	1.839e-01	6.39
1065	5.066e-03	7.602e-03	1.50	1.171e-01	23.11
1803	3.042e-03	2.868e-03	0.94	1.028e-01	33.78
3045	1.561e-03	2.801e-03	1.79	1.067e-01	68.39
5643	5.790e-04	5.790e-04	1.00	5.551e-02	95.88

Table 4.8: Supersonic BAC3-11 flow. Adaptive algorithm for the point evaluation of the pressure at the leading edge.

displayed in Figure 4.28(b). This figure illustrates some principles of information transport in supersonic as well as in subsonic flow regions. To the right–hand side of the transonic lines the dual solution is zero as no information, neither by material transport nor even by information transport due to sound waves, can enter the subsonic region from the supersonic one. Within the whole subsonic region the dual solution is non-zero corresponding to the fact that sound waves can reach the point of evaluation from any point in the subsonic area and that all numerical errors which occur within this subsonic region can (even though possibly to a small portion) affect the value of the solution at the point of evaluation. However, the dual solution in the subsonic region is concentrated in a thin spike that is transported upstream from the point of evaluation in direction of the flow. This spike corresponds to the path of material transport and represents the main path of information transport. To the left of the airfoil, this spike crosses the shock and splits into three spikes while entering the supersonic region left of the shock. These spikes are transported upstream along the characteristics corresponding to the three eigenvalues $\bar{v}$ and $\bar{v} \pm c$. We recall that the characteristic corresponding to $\bar{v}$ represents the path of material transport, that in this example is given by the line inclined at 5 degrees, whereas the characteristics corresponding to $\bar{v} \pm c$ represent the paths of information transport due to sound waves.

In Table 4.8 we show the performance of our adaptive algorithm. Here, we see that on very coarse meshes the quality of the computed error representation formula $\tilde{\mathcal{E}}_\Omega$ is quite poor, in the sense that the effectivity index θ_1 is not close to one. However, ignoring the first few refinement steps, we see that θ_1 lies in the interval $(0.94, 1.79)$. Moreover, as in previous examples, we observe that the Type I error bound $\tilde{\mathcal{E}}_{\lvert\Omega\rvert}$ overestimates the true error by several orders of magnitude. Here, the effectivity index θ_2 is even increasing as the mesh is refined. This clearly shows that any error estimate (a Type II error bound, for example) derived by further bounding $\tilde{\mathcal{E}}_{\lvert\Omega\rvert}$ will dramatically overestimate the

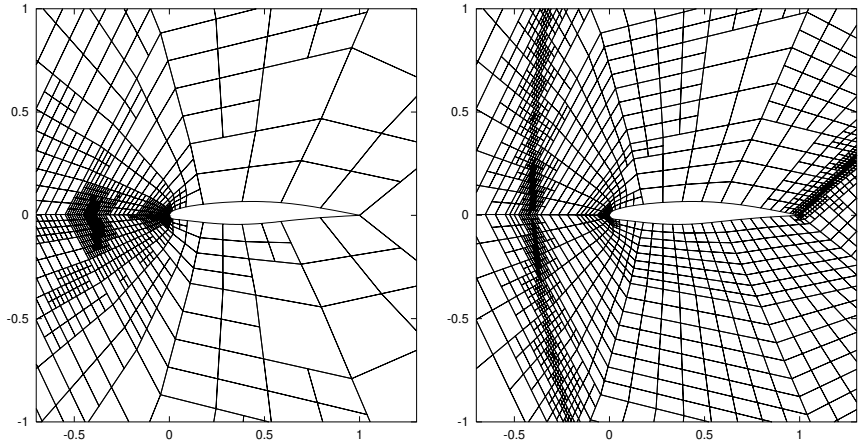


Figure 4.29: Supersonic BAC3-11 flow. (a) Mesh constructed using dual error indicator with 1803 elements ($|J(u) - J(u_h)| = 3.042 \times 10^{-3}$); (b) Mesh constructed using *ad hoc* error indicator with 13719 elements ($|J(u) - J(u_h)| = 3.542 \times 10^{-2}$).

true error rendering the error estimate useless in practice.

In Figure 4.29 we show the meshes produced using the weighted and the *ad hoc* error indicators. Here, we see that the mesh constructed using η_K^{adhoc} is concentrated in the neighbourhood of the two shocks. In contrast, the mesh produced using the weighted indicator $|\tilde{\eta}_K|$ only refines the mesh in the vicinity of the point of evaluation and the part of the shock where the spike of the dual solution, i.e. where the main part of information, crosses the shock. The other parts of the shock are not resolved, as the numerical error in these regions only has a small affect on the accuracy of the solution at the point of evaluation. Also there is no refinement in the vicinity of the shock emanating from the trailing edge of the airfoil; thereby, this shock is not well resolved at all. Nevertheless, the solution at the leading edge of the airfoil is not affected by this as no information is transported upstream from the trailing edge, located in a supersonic part of the flow, to the leading edge, located in the subsonic region. As in the previous examples, we see that the adaptively refined meshes generated by employing the weighted error indicator $|\tilde{\eta}_K|$ are much more economical than those produced using the traditional *ad hoc* error indicator η_K^{adhoc}. Indeed, in Figure 4.30 we clearly observe the superiority of the former error indicator; on the final mesh the true error in the computed functional is over two orders of magnitude smaller when the weighted error indicator is employed.

Motivated by the structure of the mesh generated by the weighted error indicator, here we also consider the performance of an alternative *ad hoc* error indicator based on a modification of η_K^{adhoc}, whereby only elements in a neighbourhood of a region upstream of the point of interest are marked for refinement.

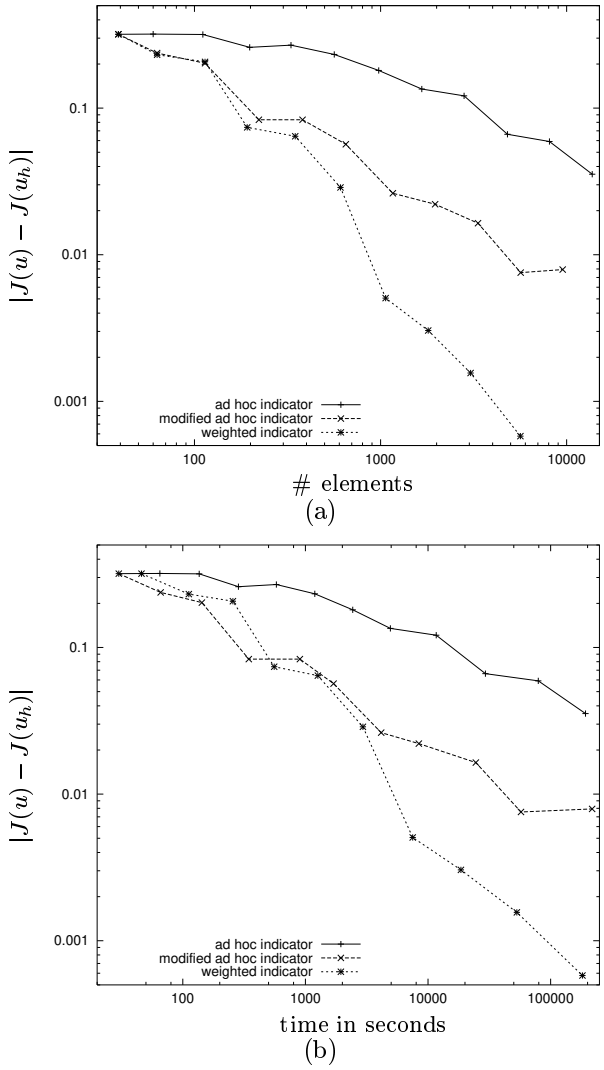


Figure 4.30: Supersonic BAC3-11 flow. Use of the weighted, the *ad hoc* and the modified *ad hoc* error indicators. Convergence of $|J(u) - J(u_h)|$ vs. (a) number of elements and (b) time in seconds.

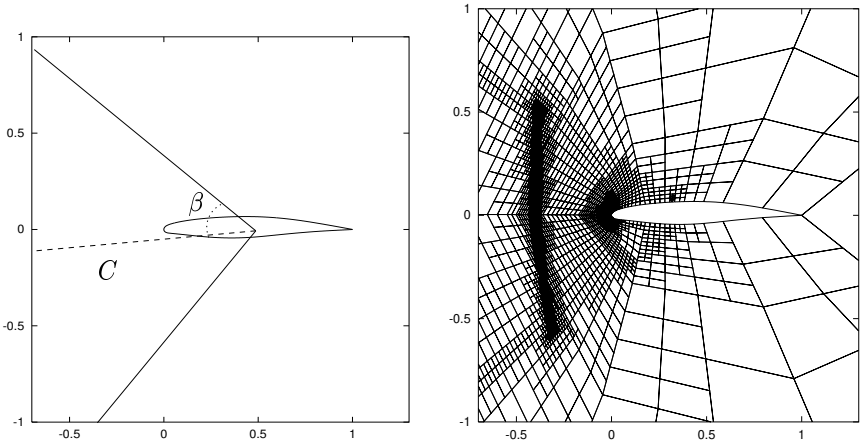


Figure 4.31: Supersonic BAC3-11 flow. (a) Cone C: domain where the modified *ad hoc* error indicator is active; (b) Mesh constructed using the modified *ad hoc* error indicator with 9516 elements ($|J(u) - J(u_h)| = 7.924 \times 10^{-3}$).

More precisely, we write C to denote the cone depicted in Figure 4.31(a) with apex half angle β, located in the center of the airfoil with symmetry axes inclined at $\alpha = 5°$ according to the direction of the inflow. We now define the modified *ad hoc* indicator $\eta_K^{\mathrm{adhoc,C}}$ as follows:

$$\eta_K^{\mathrm{adhoc,C}} = \begin{cases} \eta_K^{\mathrm{adhoc}}, & \text{if centroid}(K) \in C, \\ 0, & \text{otherwise.} \end{cases}$$

This modification takes into account that we are not interested in the flow field in the whole domain, but only in the point value of the pressure at the leading edge. Thereby, adaptive mesh refinement is inhibited in the region downstream of the airfoil including the neighbourhood of the shock emanating from the trailing edge. Furthermore, refinement of the shock in front of the leading edge of the airfoil is prevented in regions that are placed too far above or below the airfoil since a low resolution of this shock in these areas is believed to not significantly degrade the accuracy of the pressure value at the leading edge, cf. Figure 4.29(a). In Figure 4.31(b) we show the mesh produced by employing $\eta_K^{\mathrm{adhoc,C}}$ with $\beta = 45°$.

Finally, in Figure 4.30 we see that the modified *ad hoc* indicator produces meshes that are much more efficient for computing the value of the pressure at the leading edge of the airfoil in comparison to the (unmodified) *ad hoc* indicator η_K^{adhoc}. Nevertheless, the meshes produced using the weighted indicators are even more efficient than those designed by $\eta_K^{\mathrm{adhoc,C}}$; on the final mesh the true error in the computed functional is over an order of magnitude smaller when the weighted error indicator is employed. We note that the chosen shape and size of the subdomain C and the resulting modified indicator only represents an

'attempt' to find a reasonable modification of the *ad hoc* indicator η_K^{adhoc} that is capable of efficiently computing the pressure at the leading edge of the airfoil and to provide a 'fair' comparison with the goal–oriented weighted indicator $|\tilde{\eta}_K|$. Indeed, the value of the angle β may be chosen differently, though *a priori* it is unclear which parts of the shock in front of the leading edge of the airfoil will influence the target functional. The angle β should not be chosen too small as otherwise the lack of resolution of the shock in front of the leading edge of the airfoil will impact on the computed value of the pressure at the point of interest; on the other hand choosing β too large may lead to over–refinement. In contrast, the weighted error indicator provides all the necessary information in order to decide which regions of the shock should be refined, and by what extent.

Conclusion

In this work, we have developed the *a posteriori* error analysis of the adaptive discontinuous Galerkin finite element method for systems of nonlinear hyperbolic conservation laws.

Discretisation: We implemented a discontinuous Galerkin discretisation for systems of conservation equations including the 2D compressible Euler equations. This method satisfies both a global and a local conservation property. In order to avoid spurious oscillations in the vicinity of shocks, this method is enhanced by consistent shock-capturing terms. Furthermore, we employed higher order approximations of curved reflective boundaries to avoid unphysical solutions. For the discontinuous Galerkin method with piecewise polynomials of degree 1, the DG(1) method, full order of convergence was obtained when at least a piecewise quadratic approximation of curved reflective boundaries was employed. For the solution of the nonlinear problems we employed a Newton iteration method. This was found to be superior to pseudo-time iteration methods as they are computationally very expensive for stationary flows.

Error estimation: By employing a duality argument, we obtained a residual-based error representation formula and so–called weighted or Type I *a posteriori* error bounds with respect to physically relevant quantities of the solution; typical examples in aero-dynamics include the drag or lift of an airfoil, the pressure difference between the leading nose and trailing edge of the airfoil or single density or pressure values on the profile of the airfoil. The error representation includes the element–residuals multiplied by local weights involving the solution of a dual problem. To render the error representation formula computable we solved the dual problem numerically resulting in an approximative error representation. Numerical experiments covering a wide range of hyperbolic problems have been presented to illustrate the quality of this approximate error representation formula. Furthermore, it was shown that Type I *a posteriori* estimates will in general not be sharp; indeed, in some examples the Type I *a posteriori* bound derived from the error representation formula by simply employing the triangle inequality overestimates the error in the computed target functional by several orders of magnitude. Thereby, any further bounding performed *en route* to deriving Type II *a posteriori* estimates will further adversely affect the quality of the computed error bound.

Adaptivity: Based on the so–called weighted indicators occurring in the approximate error representation formula we designed and implemented an adap-

tive algorithm that produces meshes specifically tailored to the efficient computation of the quantity of interest. The performance of the proposed adaptive strategy was illustrated by a series of numerical experiments. In particular, we clearly demonstrated the superiority of this approach over standard mesh refinement algorithms which employ *ad hoc* error indicators as well as over hand-made meshes specifically tailored to the efficient computation of the target quantity.

Outlook: Several open questions still remain; first of all the issue of well–posedness of the dual problem which, although being linear, is a non-standard hyperbolic problem in the sense that the coefficients may be discontinuous. The only known analytical results, which hold in the one–dimensional case, are those by Tadmor [53]. In addition, further numerical tests are necessary concerning the quality of the (approximate) error representation formula in the presence of shocks, cf. [25]. Another subject of research activity will be the so–called *multi–target adaptivity*, see [28], that aims to design meshes for the efficient computation of several different quantities of interest *simultaneously*, as for example the drag and lift coefficient.

Finally, this work may act as basis for further research in several different major directions: Concerning adaptivity the approach proposed in this work can be extended to *hp*-methods, see [34] and [52]. Concerning the governing equations, extensions are possible to all hyperbolic and nearly hyperbolic problems like *viscous* compressible gas flows, for example. Further extensions of the *a posteriori* error estimation and adaptive mesh design are possible for the time-accurate solution of the *non-stationary* compressible Euler equations.

Appendix A

Governing equations

Here, we collect some useful properties of the 1D and 2D compressible Euler equations of gas dynamics.

A.1 1D Euler equations

The one-dimensional time-dependent Euler equations are given by

$$\partial_t u + \partial_x f(u) = 0, \tag{A.1}$$

with the state vector $u = (u_1, u_2, u_3) \equiv (\varrho, \varrho v, e)^T$ given in conservative variables and the flux $f(u) = (\varrho v, \varrho v^2 + p, v(e+p))^T$, where ϱ, v and e represent the density, the velocity vector and the total energy per unit volume, respectively. Here, e consists of two parts, $e = e_{kin} + e_{in}$, the kinetic energy $e_{kin} = \frac{1}{2}\varrho v^2$ and the internal energy $e_{in} = e_{in}(p)$ that, for a ideal gas, satifies the equation $e_{in} = e_{in}(p) = \frac{p}{(\gamma-1)}$, with $\gamma = \frac{c_p}{c_v}$ denoting the ratio of specific heats, for example, $\gamma = 1.4$ for dry air. Hence, the pressure is given by the following equation of state of an ideal gas

$$p = (\gamma - 1)(e - \frac{1}{2}\varrho v^2). \tag{A.2}$$

In quasi-linear form the 1D Euler equations (A.1) are represented by

$$u_t + A(u)u_x = 0,$$

with the Jacobi matrix $A(u) = \partial_u f(u)$,

$$A(u) = \begin{pmatrix} 0 & 1 & 0 \\ -\frac{1}{2}(3-\gamma)\left(\frac{u_2}{u_1}\right)^2 & (3-\gamma)\frac{u_2}{u_1} & (\gamma-1) \\ -\frac{\gamma u_2 u_3}{u_1^2} + (\gamma-1)\left(\frac{u_2}{u_1}\right)^3 & \gamma\frac{u_3}{u_1} - \frac{3}{2}(\gamma-1)\left(\frac{u_2}{u_1}\right)^2 & \gamma\frac{u_2}{u_1} \end{pmatrix}, \tag{A.3}$$

that represents the derivative of the flux $f(u)$ with respect to *conservative* variables u_i, $i = 1, 2, 3$.

Lemma A.1 (Eigenvalues and eigenvectors of $A(u)$) *The eigenvalues of the coefficient matrix $A(u)$ given in* (A.3) *are*

$$\lambda_1 = v - c, \qquad \lambda_2 = v, \qquad \lambda_3 = v + c,$$

with $c := \sqrt{\frac{\gamma p}{\varrho}}$ denoting the speed of sound. The corresponding right eigenvectors are

$$r^{(1)} = \begin{pmatrix} 1 \\ v - c \\ H - vc \end{pmatrix}, \quad r^{(2)} = \begin{pmatrix} 1 \\ v \\ \frac{1}{2}v^2 \end{pmatrix}, \quad r^{(3)} = \begin{pmatrix} 1 \\ v + c \\ H + vc \end{pmatrix},$$

with the total specific enthalpy H and the specific enthalpy h defined by

$$H = \frac{e + p}{\varrho} = \frac{1}{2}v^2 + h, \qquad h = \frac{e_{in} + p}{\varrho}, \tag{A.4}$$

respectively.

Proof: see [54] for example. □

A.2 2D Euler equations

The two-dimensional time-dependent version of the Euler equation is given by

$$u_t + f_1(u)_{x_1} + f_2(u)_{x_2} = 0,$$

with the state vector $u = (u_1, u_2, u_3, u_4)^T \equiv (\varrho, \varrho v_1, \varrho v_2, e)^T$ given in conservative variables and the fluxes

$$f_1(u) = \begin{pmatrix} \varrho v_1 \\ \varrho v_1^2 + p \\ \varrho v_1 v_2 \\ v_1(e + p) \end{pmatrix}, \qquad f_2(u) = \begin{pmatrix} \varrho v_2 \\ \varrho v_1 v_2 \\ \varrho v_2^2 + p \\ v_2(e + p) \end{pmatrix},$$

the velocity vector $v = (v_1, v_2)^T$ and the pressure $p = (\gamma - 1)(e - \frac{1}{2}\varrho v^2)$.

In quasi-linear form the stationary 2D Euler equations are given by

$$A_1(u)u_{x_1} + A_2(u)u_{x_2} = 0,$$

with the coefficient matrices $A_i(u) = \frac{\partial f_i}{\partial u}$, $i = 1, 2$, where

$$A_1(u) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -v_1^2 + \frac{1}{2}(\gamma - 1)V^2 & (3 - \gamma)v_1 & -(\gamma - 1)v_2 & \gamma - 1 \\ -v_1 v_2 & v_2 & v_1 & 0 \\ v_1\left(\frac{1}{2}(\gamma - 1)V^2 - H\right) & H - (\gamma - 1)v_1^2 & -(\gamma - 1)v_1 v_2 & \gamma v_1 \end{pmatrix}, \tag{A.5}$$

$$A_2(u) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -v_1 v_2 & v_2 & v_1 & 0 \\ -v_2^2 + \frac{1}{2}(\gamma-1)V^2 & -(\gamma-1)v_1 & (3-\gamma)v_2 & \gamma - 1 \\ v_2\left(\frac{1}{2}(\gamma-1)V^2 - H\right) & -(\gamma-1)v_1 v_2 & H - (\gamma-1)v_2^2 & \gamma v_2 \end{pmatrix}. \tag{A.6}$$

Definition A.2 *Given $u \in \mathbb{R}^m$, $n \in \mathbb{R}^d$ and some matrix functions $A_i : \mathbb{R}^m \to \mathbb{R}^{m\times m}$, $i = 1, \ldots, d$, we define $B(\cdot,\cdot) \in \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^{m\times m}$ by*

$$B(u,n) = \sum_{i=1}^{d} A_i(u) n_i.$$

This corresponds to the definition of the matrix B in (2.3).

Lemma A.3 (Eigenvalues of $B(u,n)$) *For the 2D Euler equations the eigenvalues of the matrix $B(u,n)$ defined in Definition A.2 are*

$$\lambda_1 = v \cdot n - c, \qquad \lambda_2 = \lambda_3 = v \cdot n, \qquad \lambda_4 = v \cdot n + c,$$

with $a := \sqrt{\frac{\gamma p}{\varrho}}$ denoting the speed of sound. The corresponding right eigenvectors are

$$r^{(1)} = \begin{pmatrix} 1 \\ v_1 - c \\ v_2 \\ H - v_1 c \end{pmatrix}, \qquad r^{(2)} = \begin{pmatrix} 1 \\ v_1 \\ v_2 \\ \frac{1}{2}v^2 \end{pmatrix},$$

$$r^{(3)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ v_2 \end{pmatrix}, \qquad r^{(4)} = \begin{pmatrix} 1 \\ v_1 + c \\ v_2 \\ H + v_1 c \end{pmatrix},$$

with the total specific enthalpy H and the specific enthalpy h defined as in (A.4).

Proof: see [54] for example. □

In Section 2.8.4 we even needed second derivatives of the fluxes $f_i(u)$, $i = 1,2$. In order to compute these derivatives, we replace v_1 and v_2 in (A.5) and (A.6) by conservative variables $u = (u_1, u_2, u_3, u_4) = (\varrho, \varrho v_1, \varrho v_2, e)$ and H by $H = \frac{\gamma u_4}{u_1} - \frac{1}{2}(\gamma-1)\left(\frac{u_2^2}{u_1^2} + \frac{u_3^2}{u_1^2}\right)$ resulting in following representation of the Jacobians

$$A_1(u) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2}(3-\gamma)\frac{u_2^2}{u_1^2} + \frac{1}{2}(\gamma-1)\frac{u_3^2}{u_1^2} & (3-\gamma)\frac{u_2}{u_1} & -(\gamma-1)\frac{u_3}{u_1} & \gamma - 1 \\ -\frac{u_2 u_3}{u_1^2} & \frac{u_3}{u_1} & \frac{u_2}{u_1} & 0 \\ \frac{u_2}{u_1}\left((\gamma-1)\left(\frac{u_2^2}{u_1^2} + \frac{u_3^2}{u_1^2}\right) - \frac{\gamma u_4}{u_1}\right) & \frac{\gamma u_4}{u_1} - \frac{1}{2}(\gamma-1)\left(3\frac{u_2^2}{u_1^2} + \frac{u_3^2}{u_1^2}\right) & -(\gamma-1)\frac{u_2 u_3}{u_1^2} & \gamma\frac{u_2}{u_1} \end{pmatrix},$$

$$A_2(u) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -\frac{u_2 u_3}{u_1^2} & \frac{u_3}{u_1} & \frac{u_2}{u_1} & 0 \\ -\frac{1}{2}(3-\gamma)\frac{u_2^2}{u_1^2} + \frac{1}{2}(\gamma-1)\frac{u_3^2}{u_1^2} & -(\gamma-1)\frac{u_2}{u_1} & (3-\gamma)\frac{u_3}{u_1} & \gamma-1 \\ \frac{u_3}{u_1}\left((\gamma-1)\left(\frac{u_2^2}{u_1^2}+\frac{u_3^2}{u_1^2}\right) - \frac{\gamma u_4}{u_1}\right) & -(\gamma-1)\frac{u_2 u_3}{u_1^2} & \frac{\gamma u_4}{u_1} - \frac{1}{2}(\gamma-1)\left(\frac{u_2^2}{u_1^2}+3\frac{u_3^2}{u_1^2}\right) & \gamma\frac{u_3}{u_1} \end{pmatrix}.$$

The derivatives of these matrices are given in the following

$$\partial_{u_1} A_1(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ (3-\gamma)\frac{u_2^2}{u_1^3} - (\gamma-1)\frac{u_3^2}{u_1^3} & -(3-\gamma)\frac{u_2}{u_1^2} & (\gamma-1)\frac{u_3}{u_1^2} & 0 \\ 2\frac{u_2 u_3}{u_1^3} & -\frac{u_3}{u_1^2} & -\frac{u_2}{u_1^2} & 0 \\ -3(\gamma-1)\left(\frac{u_2^3}{u_1^4}+\frac{u_2 u_3^2}{u_1^4}\right) + 2\frac{\gamma u_2 u_4}{u_1^3} & -\frac{\gamma u_4}{u_1^2} + (\gamma-1)\left(3\frac{u_2^2}{u_1^3}+\frac{u_3^2}{u_1^3}\right) & 2(\gamma-1)\frac{u_2 u_3}{u_1^3} & -\gamma\frac{u_2}{u_1^2} \end{pmatrix},$$

$$\partial_{u_2} A_1(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -(3-\gamma)\frac{u_2}{u_1^2} & (3-\gamma)\frac{1}{u_1} & 0 & 0 \\ -\frac{u_3}{u_1^2} & 0 & \frac{1}{u_1} & 0 \\ (\gamma-1)\left(3\frac{u_2^2}{u_1^3}+\frac{u_3^2}{u_1^3}\right) - \frac{\gamma u_4}{u_1^2} & -3(\gamma-1)\frac{u_2}{u_1^2} & -(\gamma-1)\frac{u_3}{u_1^2} & \frac{\gamma}{u_1} \end{pmatrix},$$

$$\partial_{u_3} A_1(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ (\gamma-1)\frac{u_3}{u_1^2} & 0 & -(\gamma-1)\frac{1}{u_1} & 0 \\ -\frac{u_2}{u_1^2} & \frac{1}{u_1} & 0 & 0 \\ 2(\gamma-1)\frac{u_2 u_3}{u_1^3} & -(\gamma-1)\frac{u_3}{u_1^2} & -(\gamma-1)\frac{u_2}{u_1^2} & 0 \end{pmatrix},$$

$$\partial_{u_4} A_1(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{\gamma u_2}{u_1^2} & \frac{\gamma}{u_1} & 0 & 0 \end{pmatrix}.$$

$$\partial_{u_1} A_2(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 2\frac{u_2 u_3}{u_1^3} & -\frac{u_3}{u_1^2} & -\frac{u_2}{u_1^2} & 0 \\ (3-\gamma)\frac{u_3^2}{u_1^3} - (\gamma-1)\frac{u_2^2}{u_1^3} & (\gamma-1)\frac{u_2}{u_1^2} & -(3-\gamma)\frac{u_3}{u_1^2} & 0 \\ -3(\gamma-1)\left(\frac{u_2^2 u_3}{u_1^4}+\frac{u_3^3}{u_1^4}\right) + 2\frac{\gamma u_3 u_4}{u_1^3} & 2(\gamma-1)\frac{u_2 u_3}{u_1^3} & -\frac{\gamma u_4}{u_1^2} + (\gamma-1)\left(\frac{u_2^2}{u_1^3}+3\frac{u_3^2}{u_1^3}\right) & -\gamma\frac{u_3}{u_1^2} \end{pmatrix},$$

$$\partial_{u_2} A_2(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\frac{u_3}{u_1^2} & 0 & \frac{1}{u_1} & 0 \\ (\gamma-1)\frac{u_2}{u_1^2} & -(\gamma-1)\frac{1}{u_1} & 0 & 0 \\ 2(\gamma-1)\frac{u_2 u_3}{u_1^3} & -(\gamma-1)\frac{u_3}{u_1^2} & -(\gamma-1)\frac{u_2}{u_1^2} & 0 \end{pmatrix},$$

$$\partial_{u_3} A_2(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\frac{u_2}{u_1^2} & \frac{1}{u_1} & 0 & 0 \\ -(3-\gamma)\frac{u_3}{u_1^2} & 0 & (3-\gamma)\frac{1}{u_1} & 0 \\ (\gamma-1)\left(\frac{u_2^2}{u_1^3}+3\frac{u_3^2}{u_1^3}\right) - \frac{\gamma u_4}{u_1^2} & -(\gamma-1)\frac{u_2}{u_1^2} & -3(\gamma-1)\frac{u_3}{u_1^2} & \frac{\gamma}{u_1} \end{pmatrix},$$

$$\partial_{u_4} A_2(u) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{\gamma u_3}{u_1^2} & 0 & \frac{\gamma}{u_1} & 0 \end{pmatrix}.$$

Appendix B

Analytical solutions to hyperbolic problems

B.1 Buckley-Leverett equation Riemann problem

The Buckley-Leverett equation represents a simple model for *two phase flow* in a porous medium. An application is the simulation of an oil reservoir. The equation is given by

$$u_t + f(u)_x = 0,$$

with the *non-convex* flux function

$$\begin{aligned} f(u) &= \frac{u^2}{u^2 + a(1-u)^2}, \\ f'(u) &= \frac{2au(1-u)}{(u^2 + a(1-u)^2)^2}. \end{aligned} \tag{B.1}$$

The solution to Riemann problems for scalar equations with *convex* fluxes, like Burgers' equation, includes either a shock or a rarefaction wave. Equations with non-convex fluxes, like the Buckley-Leverett equations, might involve both. A standard example is the Riemann problem

$$u(0,x) = u_0(x) = \begin{cases} 1 & \text{if} \quad x < 0, \\ 0 & \text{if} \quad x > 0, \end{cases} \tag{B.2}$$

modelling pure water ($u = 1$) for $x < 0$ and pure oil ($u = 0$) for $x > 0$ at initial time $t = 0$. The Buckley-Leverett equation simulates the replacement of the oil by water that is pumped from the left. This leads to a shock front at the interface between the oil and the water, followed by a rarefaction wave, that indicates for the time $t > 0$ a specific proportion of water and oil in the porous

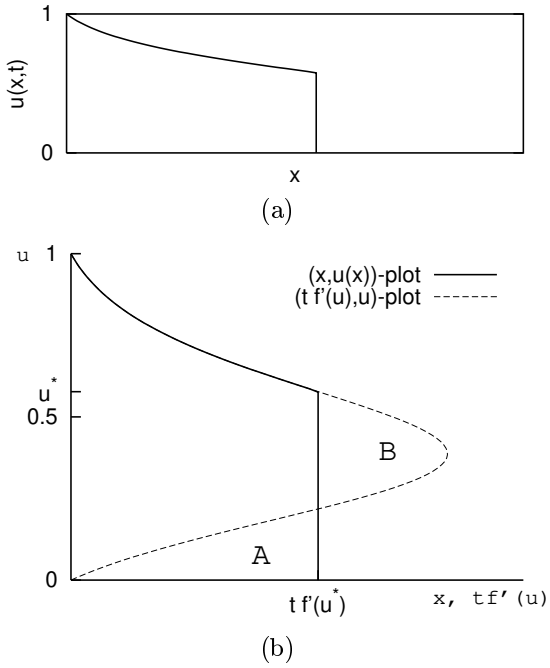


Figure B.1: Riemann solution for Buckley-Leverett equation.

medium, see Figure B.1(a). The rarefaction wave in $x > 0$ exists for all time $t > 0$, hence $u < 1$ for all $(x, t) \in \mathbb{R}^+ \times \mathbb{R}^+$, simulating the effect that the total amount of oil may never be recovered by this 'secondary recovery' technique.

Exact solution to the Riemann problem: Using the equal area rule (equalise areas A and B in Figure B.1(b)) in order to replace the triple-valued solution in the $(tf'(u),\ u(t))$ - plot by a correct shock leads to the following determinant equation at time t

$$\int_0^{u^\star(t)} tf'(u)\,du = u^\star(t)\,tf'(u^\star(t)),$$

with $u^\star(t)$ denoting the post-shock value. Evaluating the integral and division by $t > 0$ results in

$$f(u^\star(t)) - f(0) = u^\star(t)\,f'(u^\star(t)). \tag{B.3}$$

Using the explicit form (B.1) of the flux gives

$$(u^\star(t))^2 + a(1 - u^\star(t))^2 = 2a(1 - u^\star(t)) \tag{B.4}$$

leading to $u^\star(t) = u^\star$, with

$$u^\star = \left(\frac{a}{1+a}\right)^{\frac{1}{2}},$$

a constant value. The fact that this post-shock value is constant in time is known also due to the self-similarity of solutions to the Riemann problem. The shock velocity is $s = f'(u^\star)$, hence by using (B.3) and (B.4)

$$s = f'(u^\star) = \frac{f(u^\star)}{u^\star} = \frac{u^\star}{(u^\star)^2 + a\,(1-u^\star)^2} = \frac{u^\star}{2a\,(1-u^\star)}.$$

The solution to the Riemann problem (B.2) is given as follows

$$u(x,t) = \begin{cases} 1 & \text{for } x < 0, \\ u, \text{ with } x = tf'(u) & \text{for } 0 < x < st, \\ 0 & \text{for } x > st. \end{cases}$$

For general a the exact solution $u(x,t)$ in the part $0 < x < st$ is too complicated to be given here. In the following, we only give the solution $u(x,t)$ for a specific choice of a. For $a = \frac{1}{2}$ the solution u turns out to be

$$u(x,t) = \frac{1}{3} + \frac{1}{18}\sqrt{6}u_2(x,t) + \frac{1}{18\sqrt{x}}\left(-48(t+x) - 6u_1(x,t) - 24\frac{(t-2x)^2}{u_1(x,t)} + \frac{36t\sqrt{6}}{u_2(x,t)}\right)^{\frac{1}{2}},$$

with the functions $u_1(x,t)$ and $u_2(x,t)$ given by

$$u_1(x,t) = \left(8(t-2x)^3 + 243xt^2 + 9xt\left(\frac{3}{x}\left(16(t-2x)^3 + 243xt^2\right)\right)^{\frac{1}{2}}\right)^{\frac{1}{3}},$$

$$u_2(x,t) = \left(-4\left(1+\frac{t}{x}\right) + \frac{u_1(x,t)}{x} + \frac{4(t-2x)^2}{xu_1(x,t)}\right)^{\frac{1}{2}}.$$

B.2 1D Euler equations: Sod's test problem

This problem, also referred to as the *shock-tube problem*, is an initial value problem for the 1D Euler equations subject to the initial condition

$$u(0,x) = u_0(x) = \begin{cases} u_L & \text{if} \quad x < x_0, \\ u_R & \text{if} \quad x > x_0, \end{cases}$$

where $(\varrho_L, v_L, p_L) = (1, 0, 1)$ and $(\varrho_R, v_R, p_R) = (0.125, 0, 0.1)$. According to [54] the solution to this Riemann problem includes a left rarefaction wave and a right shock wave. The intermediate solution in the so called *star*-region consists of two states $u_L^\star$ and $u_R^\star$ separated by a contact discontinuity. The complete solution, also shown in Figure B.2, consists of five parts separated by the right shock with shock velocity

$$v_s = v_R + a_R\left(\frac{\gamma+1}{2\gamma}\frac{p^\star}{p_R} + \frac{\gamma-1}{2\gamma}\right)^{\frac{1}{2}},$$

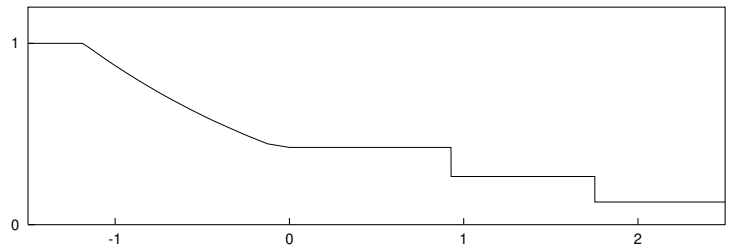


Figure B.2: Solution to Sod's problem: $\varrho(0.5, x)$

the contact discontinuity with velocity

$$v^\star = a_R \left(\frac{2}{\gamma(\gamma-1)}\right)^{\frac{1}{2}} \left(\frac{p^\star}{p_R} - 1\right) \left(1 + \frac{\gamma+1}{\gamma-1}\frac{p^\star}{p_R}\right)^{-\frac{1}{2}},$$

and the tail and the head of the rarefaction wave with velocities $v_{\text{tail}} = v^\star - a_L^\star$ and $v_{\text{head}} = v_L - a_L$, respectively. For $x < x_0 + v_{\text{head}}t$ and $x > x_0 + v_s t$ the solution equals the left and right initial values (ϱ_L, v_L, p_L) and (ϱ_R, v_R, p_R), respectively. The rarefaction wave in the region $x_0 + v_{\text{head}}t < x < x_0 + v_{\text{tail}}t$ is given by

$$\begin{aligned}
\varrho(x,t) &= \varrho_L \left(\frac{2}{\gamma+1} + \frac{\gamma-1}{(\gamma+1)a_L}\left(u_L - \frac{x}{t}\right)\right)^{\frac{2}{\gamma-1}}, \\
v(x,t) &= \frac{2}{\gamma+1}\left(a_L + \frac{\gamma-1}{2}u_L + \frac{x}{t}\right), \\
p(x,t) &= p_L \left(\frac{2}{\gamma+1} + \frac{\gamma-1}{(\gamma+1)a_L}\left(u_L - \frac{x}{t}\right)\right)^{\frac{2\gamma}{\gamma-1}};
\end{aligned}$$

the left star state $u_L^\star$ is given by $(\varrho_L^\star, v_L^\star, p_L^\star) = (\varrho_L^\star, v^\star, p^\star)$ and the right star state $u_R^\star$ by $(\varrho_R^\star, v_R^\star, p_R^\star) = (\varrho_R^\star, v^\star, p^\star)$, with

$$\varrho_L^\star = \varrho_L \left(\frac{p^\star}{p_L}\right)^{\frac{1}{g}}, \qquad \varrho_R^\star = \varrho_R \left(\frac{p^\star}{p_R} + \frac{\gamma-1}{\gamma+1}\right)\left(\frac{\gamma-1}{\gamma+1}\frac{p^\star}{p_R} + 1\right)^{-\frac{1}{2}}.$$

The last remaining unknown $p^\star$ may be computed by solving following nonlinear equation

$$\left(\frac{2}{\gamma(\gamma-1)}\right)\left(\frac{p^\star}{p_R} - 1\right)\left(1 + \frac{\gamma+1}{\gamma-1}\frac{p^\star}{p_R}\right)^{-\frac{1}{2}} = \frac{2}{\gamma-1}\frac{a_L}{a_R}\left(1 - \left(\frac{p^\star}{p_L}\right)^{\frac{\gamma-1}{2\gamma}}\right).$$

For the Sod's problem we get

$$p^\star = 0.30313.$$

B.3 1D Euler equations: Shock conditions

Here, we consider the shock conditions of the 1D compressible Euler equations that also represent normal shock conditions of the 2D Euler equations. The shock conditions are determined by the *Rankine-Hugoniot jump condition*

$$f(u^{(1)}) - f(u^{(0)}) = s(u^{(1)} - u^{(0)}),$$

for the states $u^{(0)}$ and $u^{(1)}$, upstream and downstream the shock, respectively. For the stationary (i.e. $s = 0$) 1D compressible Euler equations,

$$\partial_x \begin{pmatrix} \varrho v \\ \varrho v^2 + p \\ v(e+p) \end{pmatrix} = 0,$$

the Rankine-Hugoniot jump condition is given by the following equations (B.5)a,b and c:

$$\begin{pmatrix} \varrho_0 v_0 - \varrho_1 v_1 \\ p_0 + \varrho_0 v_0^1 - p_1 - \varrho_1 v_1^2 \\ (e_0 + p_0)v_0 - (e_1 + p_1)v_1 \end{pmatrix} = 0. \tag{B.5}$$

Assuming that all quantities with suffix 0 are known, we seek to find κ, with

$$\kappa = \frac{\varrho_0}{\varrho_1} = \frac{v_1}{v_0}.$$

To that end, we employ (B.5)a and $e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v$, see the equation of state (A.2), and replace (B.5)c by

$$\frac{\gamma}{\gamma - 1}\frac{p_0}{\varrho_0} + \frac{1}{2}v_0^2 = \frac{\gamma}{\gamma - 1}\frac{p_1}{\varrho_1} + \frac{1}{2}v_1^2. \tag{B.6}$$

Then we take $\frac{\gamma}{\gamma-1}\frac{1}{\varrho_1}\times$ (B.5)b, to obtain

$$\frac{\gamma}{\gamma - 1}\frac{p_0}{\varrho_0}\kappa + \frac{\gamma}{\gamma - 1}v_1 v_0 = \frac{\gamma}{\gamma - 1}\frac{p_1}{\varrho_1} + \frac{\gamma}{\gamma - 1}v_1^2,$$

and subtract this equation from (B.6) to get

$$\frac{\gamma}{\gamma - 1}\frac{p_0}{\varrho_0}(1 - \kappa) + \left(\frac{1}{2} - \frac{\gamma}{\gamma - 1}\kappa\right) v_0^2 = -\frac{\gamma + 1}{2(\gamma - 1)}v_0^2\kappa^2.$$

Now we use $\frac{\gamma p}{\rho} = a^2$ and divide by $\frac{v_0^2}{\gamma - 1}$ giving following quadratic equation

$$\frac{\gamma + 1}{2}\kappa^2 - \left(\gamma + \frac{1}{M_0^2}\right)\kappa + \frac{\gamma - 1}{2} + \frac{1}{M_0^2} = 0.$$

One solution is $\kappa = 1$ representing no change across the shock. Therefore,

$$(\kappa - 1)\left(\frac{\gamma + 1}{2}\kappa - \frac{\gamma - 1}{2} - \frac{1}{M_0^2}\right) = 0,$$

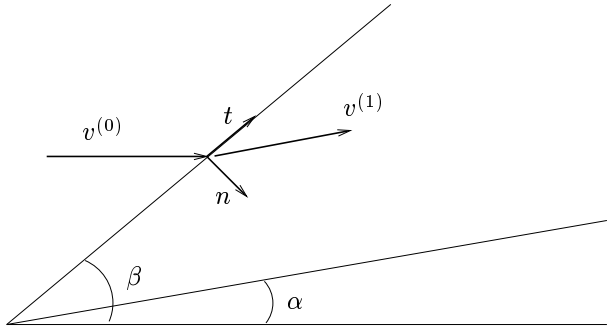


Figure B.3: Supersonic flow past a wedge problem.

resulting in

$$\kappa = \frac{\varrho_0}{\varrho_1} = \frac{v_1}{v_0} = \frac{\gamma - 1}{\gamma + 1} + \frac{2}{\gamma + 1}\frac{1}{M_0^2}. \tag{B.7}$$

To obtain p_1 we use (B.5)b

$$p_1 = p_0 + \varrho_0 v_0^2 - \varrho_1 v_1^2 = p_0 + \varrho_0 v_0^2(1 - \kappa)$$

and divide by p_0 to get

$$\begin{aligned} \frac{p_1}{p_0} &= 1 + \frac{\gamma v_0^2}{a_0^2}(1 - \kappa) \\ &= 1 + \frac{\gamma v_0^2}{a_0^2}(1 - \frac{\gamma - 1}{\gamma + 1} - \frac{2}{\gamma + 1}\frac{1}{M_0^2}) \\ &= 1 + \frac{2}{\gamma + 1}(M_0^2 - 1). \end{aligned} \tag{B.8}$$

B.4 2D Euler equations: Supersonic flow past a wedge

Now, we consider a supersonic flow past a wedge inclined at an angle α. Having a supersonic inflow of velocity $v^{(0)}$ the flow is deflected through an angle α to ensure that the velocity is parallel to the wedge surface. This is achieved by having a shock at some angle β to the inflow, see Figure B.3, with upstream and downstream states $u^{(0)}$ and $u^{(1)}$, respectively. Hence the normal and tangential velocity components denoted by subscripts n and t are given by

$$\begin{aligned} v_n^{(0)} &= |v^{(0)}|\sin(\beta), & v_t^{(0)} &= |v^{(0)}|\cos(\beta), \\ v_n^{(1)} &= |v^{(1)}|\sin(\beta - \alpha), & v_t^{(1)} &= |v^{(1)}|\cos(\beta - \alpha). \end{aligned}$$

Recalling from (B.7) and (B.8) that the normal shock conditions derived from the Rankine-Hugoniot jump condition applied to the 1D compressible Euler equations are given by

$$\begin{aligned}\frac{\varrho_0}{\varrho_1} &= \frac{v_n^{(1)}}{v_n^{(0)}} = \frac{\gamma-1}{\gamma+1} + \frac{2}{\gamma+1}\frac{1}{(M_n^{(0)})^2},\\ \frac{p_1}{p_0} &= 1 + \frac{2\gamma}{\gamma+1}((M_n^{(0)})^2 - 1),\end{aligned}$$

with $M_n^{(0)} = M^{(0)}\sin(\beta)$ denoting the normal Mach number of the inflow, and furthermore recalling that the tangential velocity is continuous across the shock,

$$v_t^{(1)} = v_t^{(0)},$$

we deduce that

$$\tan(\beta-\alpha) = \frac{v_n^{(1)}}{v_t^{(1)}} = \frac{\rho_0 v_n^{(0)}}{\rho_1}\frac{1}{v_t^{(1)}} = \frac{\rho_0}{\rho_1}\frac{v_n^{(0)}}{v_t^{(0)}} = \frac{\rho_0}{\rho_1}\tan(\beta),$$

and consequently

$$\tan(\beta-\alpha) = \left\{\frac{\gamma-1}{\gamma+1} + \frac{2}{\gamma+1}\frac{1}{(M^{(0)}\sin(\beta))^2}\right\}\tan(\beta). \tag{B.9}$$

As example, we consider a Mach 3 flow with inflow density $\rho_0 = 1$ and pressure $p_0 = 1$, over a wedge of angle $\alpha = 9.5°$, a test case previously regarded in [11]. Solving the nonlinear equation (B.9) for the angle β of the shock gives

$$\beta = 26.9308°.$$

All state variables in $u^{(1)} = (\rho_1, \rho_1 v_1^{(1)}, \rho_1 v_2^{(1)}, e_1)$ may then be determined, see Section 4.5.2 for the resulting state vector.

B.5 2D Euler equations: Ringleb flow problem

Here, we consider the Ringleb's flow problem, cf. [47], of the stationary 2D Euler equations, for which an analytical solution may be obtained using the hodograph method. Here, we briefly extract the solution to this problem from [13].

Under the restriction of an irrotational flow a stream function $\psi(x,y)$ can be introduced in terms of which the stationary 2D Euler equations reduces to the equation

$$\left(1 - \frac{v_1^2}{c^2}\right)\partial_x^2\psi - 2\frac{v_1 v_2}{c^2}\partial_x\partial_y\psi + \left(1 - \frac{v_1^2}{c^2}\right)\partial_y^2\psi = 0,$$

where c denotes the speed of sound. By applying the Molenbrock-Tschapligin transformation (for more information, see [13] and the references therein), this

problem is transformed in the (V, θ) plane, also called the 'hodograph plane', where V and θ denote the velocity and its deviation with respect to a reference direction. This leads to following equation

$$V^2 \partial_V^2 \psi + V \left(1 + \frac{V^2}{c^2}\right) \partial_V \psi + \left(1 - \frac{V^2}{c^2}\right) \partial_\theta^2 \psi = 0,$$

that may be solved by separation of variables. It is easy to verify that

$$\psi = \frac{\sin\theta}{V}, \tag{B.10}$$

is a solution to this problem. The streamlines in (B.10) are given in hodograph variables. Transformation into the physical plane results in

$$\begin{aligned} x &= \frac{1}{2}\frac{1}{\rho}\left(\frac{1}{V^2} - \frac{2}{k^2}\right) + \frac{J}{2}, \\ y &= \pm\frac{1}{k\rho V}\sqrt{1 - \left(\frac{V^2}{k^2}\right)}, \end{aligned}$$

where

$$k = \frac{1}{\psi}, \tag{B.11}$$

$$J = \frac{1}{a} + \frac{1}{3a^3} + \frac{1}{5a^5} - \frac{1}{2}\log\frac{1+a}{1-a}, \tag{B.12}$$

$$a = \sqrt{1 - \frac{\gamma - 1}{2}V^2}, \tag{B.13}$$

$$\rho = a^{2/(\gamma-1)}. \tag{B.14}$$

Furthermore, the constant velocity lines are circles given by

$$\left(x - \frac{J}{2}\right)^2 + y^2 = \frac{1}{4\rho^2 V^2}. \tag{B.15}$$

By choosing two streamlines and regarding them as reflective walls the flow represents a flow in a curved duct. For example, choosing $k_1 = 0.6$ and $k_2 = 0.98$ results in the left and right walls of the duct shown in Figure B.4. The curved top boundary is given by the circle for the constant velocity $V_1 = 0.43$. This is the same Ringleb geometry as considered [40]. This problem represents a transonic flow through a channel; it is mostly subsonic, with a small supersonic region near the right–hand side wall, cf. Figure B.4.

The exact solution to this problem is computed as follows: Given a point (x, y) in the domain, the speed of sound c is determined implicitly by the non-linear equation (B.15), where $J = J(c)$, $V = V(c)$ and $\rho = \rho(c)$ are given by (B.12), (B.13) and (B.14). Having computed c, all other variables can be evaluated.

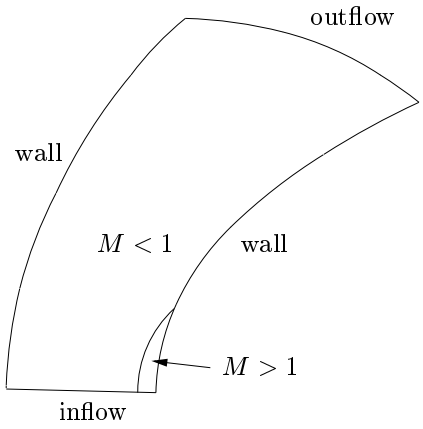


Figure B.4: Example 2. Geometry for Ringleb's flow; here M denotes the Mach number.

Appendix C

Curved elements

In many applications the domain Ω is not a polygonal domain but includes curved boundaries. For these cases the boundary cannot be represented exactly by the triangulation T_h. Approximating the boundary by a piecewise linear boundary interpolation, i.e. by a polygonal boundary, may in some applications not be sufficient, see Section 2.9 and [4]. In these cases a higher order boundary approximation, for example by piecewise quadratic or cubic boundary interpolation, must be employed. This necessitates the use of *curved elements* at the boundary of the domain. In finite element methods it is common practice to deal with curved elements by employing higher order polynomial mappings of the reference element $\hat{K}$ to the element K in real space.

In the following section we collect some formulae governing curved finite elements that hold for all smooth mapping functions. In the section thereafter we consider some implementational details for a specific class of mapping functions: namely polynomial mapping functions that are expressed in terms of Lagrangian basis functions and Lagrangian support points (nodes).

C.1 Elements with general mapping functions

This section provides a collection of some basic formulae governing geometric data of a curved element K, like the area of K and tangential or normal vectors to the boundary ∂K of the element, as well as integrals or derivatives of functions on curved elements given in terms of integrals or derivatives of functions on the reference element. These formulae hold for all smooth mappings and do not depend on the specific choice of the mapping function.

We begin by first introducing some notation. Let $K \in T_h$ be an element of the triangulation T_h with $K = \sigma_K(\hat{K})$, where σ_K is a smooth bijective mapping of the reference element $\hat{K}$ to the element K in real space, see Figure C.1.

For simplicity we suppress the letter K in the subscript and write σ instead of σ_K in the following. For a function $\varphi : K \to \mathbb{R}$ we define $\hat{\varphi} : \hat{K} \to \mathbb{R}$ to

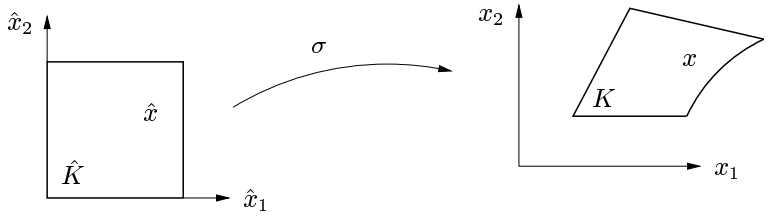


Figure C.1: Mapping σ of reference element $\hat{K}$ to the element K in real space.

satisfy

$$\hat{\varphi}(\hat{x}) = \varphi(\sigma(\hat{x})), \quad \hat{x} \in \hat{K}.$$

Differentiation with respect to variables x_j and $\hat{x}_j$ in real space and in the space of the reference element, respectively, is denoted by $\partial_j := \partial_{x_j}$ and $\hat{\partial}_j := \partial_{\hat{x}_j}$. Analogously, we define ∇ and $\hat{\nabla}$ to be the gradient vector with respect to the variables x and $\hat{x}$, respectively.

Integration of functions on elements $K = \sigma(\hat{K})$ in real space is performed by mapping the integral to the reference element $\hat{K}$ as follows

$$\int_K \varphi(x)\, dx = \int_{\hat{K}} \hat{\varphi}(\hat{x}) |\det J(\hat{x})|\, d\hat{x},$$

where J denotes the Jacobian matrix of the mapping σ, i.e. $J_{ij}(\hat{x}) := \hat{\partial}_j \sigma_i(\hat{x})$. This results in

$$|K| = \int_{\hat{K}} |\det J(\hat{x})|\, d\hat{x},$$

for the area $|K|$ of the element K. The derivatives $\hat{\partial}_j \hat{\varphi}$ and $\partial_j \varphi$ are related by

$$\hat{\partial}_j \hat{\varphi}(\hat{x}) = \hat{\partial}_j \varphi(\sigma(\hat{x})) = \partial_i \varphi(\sigma(\hat{x})) \hat{\partial}_j \sigma_i(\hat{x}) = \partial_i \varphi(\sigma(\hat{x})) J_{ij}(\hat{x})$$

and, with J^{-1} denoting the inverse of J,

$$\partial_l \varphi(x) = \partial_i \varphi(\sigma(\hat{x})) \delta_{il} = \partial_i \varphi(\sigma(\hat{x})) J_{ij}(\hat{x}) J^{-1}_{jl}(\hat{x}) = \hat{\partial}_j \hat{\varphi}(\hat{x}) J^{-1}_{jl}(\hat{x}).$$

Thereby, $\nabla \varphi(x)$ is given by the following covariant transformation

$$\nabla \varphi(x) = J^{-T}(\hat{x}) \hat{\nabla} \hat{\varphi}(\hat{x}),$$

where $J^{-T} := (J^{-1})^T = (J^T)^{-1}$.

Let $x : \mathbb{R} \to K$ be a parametric curve in real space and $\hat{x} : \mathbb{R} \to \hat{K}$ the corresponding curve transformed into the reference element, with $x(s) = \sigma(\hat{x}(s))$. The velocity vector $v(s)$ of curve $x(s)$ at s is given by

$$v_i(s) = \partial_s x_i(s) = \partial_s \sigma_i(\hat{x}(s)) = \hat{\partial}_j \sigma_i(\hat{x}(s)) \partial_s \hat{x}_j(s) = J_{ij}(\hat{x}(s)) \hat{v}_j(s);$$

hence $v(s)$ is given by following contravariant transformation

$$v(s) = J(\hat{x}(s))\hat{v}(s),$$

with $\hat{v}(s)$ denoting the velocity vector of curve $\hat{x}(s)$. A (unit) tangential vector τ is hence given by

$$\tau = \tfrac{1}{|v|}v.$$

The (unit) normal vectors are given by the cross product of the tangential vectors. Alternatively, they can be computed by a covariant transformation

$$\tilde{n} = J^{-T}\hat{n}, \qquad n = \tfrac{1}{|\tilde{n}|}\tilde{n},$$

of the (unit) normal vectors $\hat{n}$ of the curve $\hat{x}(s)$.

C.2 Polynomial mappings of higher degree

A mapping function σ that maps the reference element (unit square) $\hat{K}$ to an arbitrary quadrilateral element K, can be represented by a bilinear function, i.e. by a Q_1 mapping. For the case that K includes curved boundaries it might be necessary to employ polynomial mapping functions of higher degree.

Given a degree $p > 0$, an element $K \in T_h$, and $(p+1)^d$ mapping support points $p_i \in K$, $i = 1, \ldots, (p+1)^d$, we define a Q_p mapping $\sigma \in [Q_p]^d$ as follows

$$\sigma(\hat{x}) = \sum_{i=1}^{(p+1)^d} p_i \phi_i(\hat{x}). \tag{C.1}$$

Here, ϕ_i, $i = 1, \ldots, (p+1)^d$, denote the Lagrangian (interpolation) basis functions, that satisfy

$$\phi_i(\hat{p}_j) = \delta_{ij}, \quad i, j = 1, \ldots, (p+1)^d,$$

where $\hat{p}_i$, $i = 1, \ldots, (p+1)^d$, denote the Lagrangian support points (nodes) on the reference element $\hat{K}$. The definition of σ (C.1) ensures that each of the unit support points $\hat{p}_i$ is mapped onto the corresponding mapping support points p_i, i.e.

$$\sigma(\hat{p}_i) = p_i, \quad i = 1, \ldots, (p+1)^d. \tag{C.2}$$

Analogous to Lagrangian finite elements the unit Lagrangian support points $\hat{p}_i$ are equidistantly distributed on $\hat{K}$ based on a tensor product mesh. In the following we only consider the two–dimensional case, $d = 2$. For that case, Figure C.2 shows the distributions of the unit support points $\hat{p}_i$, $i = 1, \ldots, (p+1)^2$, for degrees $p = 1, \ldots, 4$. Let the ordering and numbering of the unit support points be as follows: first the corners, then the points on the edges and

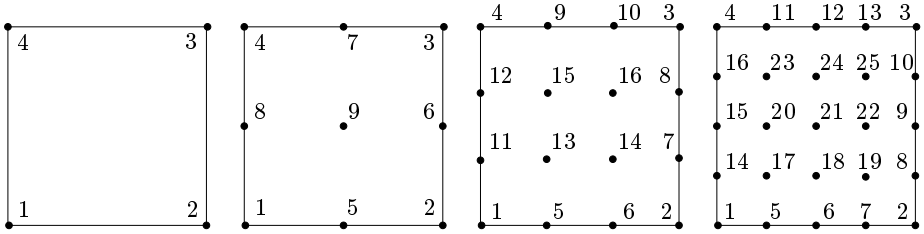


Figure C.2: Unit support points $\hat{p}_i$, $i = 1, \ldots, (p+1)^2$, for degrees $p = 1, \ldots, 4$.

finally the inner support points, see also Figure C.2. Thus the first $4p$ points are placed on the boundary $\partial\hat{K}$ of the reference element, i.e.

$$\hat{p}_k \in \partial\hat{K}, \quad k = 1, \ldots, 4p.$$

According to (C.2) these points are mapped to the mapping support points p_k, $k = 1, \ldots, 4p$ that are chosen to be placed in approximatively equal distances on the boundary of the element in real space, i.e.

$$p_k \in \partial K, \quad k = 1, \ldots, 4p.$$

While the support points p_k, $k = 1, \ldots, 4p$, on the boundary are given by the boundary description of the element K in real space, the *inner mapping support points*

$$p_i \in K \setminus \partial K, \quad i = 4p + 1, \ldots, (p + 1)^2,$$

are not uniquely determined. In the following we give two different approaches for computing appropriate positions of the inner mapping support points in real space. First we present an approach that is simple but not applicable as it produces degenerated mappings in some numerical tests. Then we introduce a more sophisticated one that employs solutions to Laplace equations on the reference element. This second approach gives a smooth transformation of the support points from the reference element to the element in real space.

C.2.1 Computation of inner support points: Simple approach

For $p = 2$ there exists only one inner mapping support point, p_9. Here we set it simply to the mean point of all points p_i on the boundary,

$$p_9 = \tfrac{1}{8} \sum_{i=1}^{8} p_i.$$

For the case of $p = 3$, four inner support points, p_i, $i = 13, \ldots, 16$ need to be computed. Let $\bar{p}$ be the mean point $\bar{p} = \frac{1}{8} \sum_{i=1}^{12} p_i$ of all points p_i on the

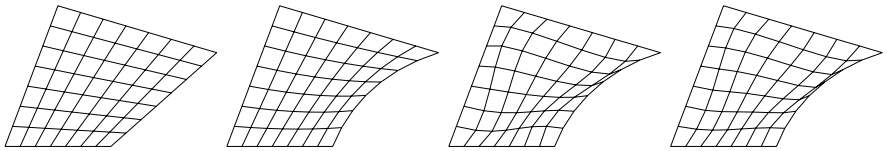


Figure C.3: $Q_1 - Q_4$ mapping of an element. Inner mapping support points are computed by a simple approach only. For the Q_3 and Q_4 mapping, right two cases, the mapping degenerates.

boundary. Then we set each of the inner support points to a weighted mean point of $\bar{p}$ and the corresponding neighbouring corner. For example we set

$$p_{13} = \tfrac{2}{3}\bar{p} + \tfrac{1}{3}p_1,$$

and points p_{14}, p_{15} and p_{16} accordingly, see third plot in Figure C.2.

Finally in the case when $p = 4$, we simply set p_{21} to the mean point

$$p_{21} = \bar{p} = \tfrac{1}{16}\sum_{i=1}^{16} p_i$$

of all points on the boundary, an inner corner point like p_{17} to

$$p_{17} = \frac{1}{2}\bar{p} + \frac{1}{2}p_1,$$

points p_{19}, p_{23} and p_{25} accordingly, see left plot in Figure C.2, and we set

$$p_{18} = \frac{1}{2}\bar{p} + \frac{1}{2}p_6,$$

and the points p_{20}, p_{22} and p_{24} accordingly.

For all cases presented here, the position of the inner points are compatible with the points in arbitrary quadrilaterals (with straight boundaries) that arise from a Q_1 mapping of the unit support points. Although this approach seems to be quite reasonable at first sight, it causes problems in some cases when the boundaries of the element K in real space are curved.

As an example, here we consider an element similar to the element K on the right of Figure 2.11, whose shape is typical for elements that arise in the neighbourhood of the leading edge of an airfoil wing. Figure C.3 shows the images of the Q_1-Q_4 mappings applied to the reference element $\hat{K}$ that, for demonstrating the behaviour of the mappings, is covered by a tensor product mesh of seven equal–sized parts in each coordinate direction. While the mesh in the image of the Q_2 mapping looks quite smooth, the images of the Q_3 and Q_4 mapping show distorted meshes with degenerated parts in the neighbourhood of the curved boundary. This behaviour is caused by the inner mapping support

points whose positions apparently are not properly chosen. We note, that if the element is slightly modified such that the curved boundary reaches even deeper into the element, then even an overlapping of the mesh has been encountered. In these degenerate regions the mapping σ is not bijective any more, and the Jacobian matrices J of the mappings degenerate and become singular.

C.2.2 Computation of inner support points: Smooth transformation

As seen in the last subsection it is not a trivial task to define the positions of the inner mapping support points such that the mapping does – in all cases – not degenerate. Therefore, here we employ an approach for the mapping of the support points, that is in the style of the smooth transformations that are used to transform structured triangulations to match complex boundary descriptions, see Section 2.10 or [21] for example. In the following, again for notational convenience, we consider only the two–dimensional case.

The smooth transformation mentioned above is based on solutions to the Laplace equation that is solved on the reference element $\hat{K}$. Discrete boundary conditions are imposed that are given by the coordinates of the mapping support points p_k, $k = 1, \ldots, 4p$, on the boundary of the element K in real space.

To be more explicit, we define a Laplace problem on $\hat{K}$

$$\begin{aligned} -\hat{\Delta}\sigma_l(\hat{x}) &= 0, & \hat{x} \in \hat{K}, \\ \sigma_l(\hat{x}) &= g_l|_{\partial\hat{K}}(\hat{x}), & \hat{x} \in \partial\hat{K}, \end{aligned} \tag{C.3}$$

for each component σ_l, $l = 1, 2$, of the Q_p mapping σ. Here, the discrete boundary function $g \in [Q_p]^2$ is given by

$$g_l(\hat{x}) = \sum_{i=1}^{4p} (p_i)_l \phi_i(\hat{x}), \quad l = 1, \ldots, d, \tag{C.4}$$

where $(p_i)_l$ denotes the lth component of the support point p_i, and ϕ_i the corresponding Lagrangian interpolation basis function. We recall that the numbering of the mapping support points involves $p_k \in \partial K$ for $k = 1, \ldots, 4p$. Substituting

$$\tilde{\sigma}_l := \sigma_l - g_l, \quad l = 1, 2, \tag{C.5}$$

into the Laplace problem (C.3) yields the zero boundary value problem,

$$\begin{aligned} -\hat{\Delta}\tilde{\sigma}_l(\hat{x}) &= \hat{\Delta} g_l(\hat{x}), & \hat{x} \in \hat{K}, \\ \tilde{\sigma}_l(\hat{x}) &= 0, & \hat{x} \in \partial\hat{K}, \end{aligned} \tag{C.6}$$

that is equivalent to the following variational formulation

$$\tilde{\sigma}_l \in H_0^1(\hat{K}): \qquad (\hat{\nabla}\tilde{\sigma}_l, \hat{\nabla}\phi)_{\hat{K}} = -(\hat{\nabla} g_l, \hat{\nabla}\phi)_{\hat{K}} \quad \forall \phi \in H_0^1(\hat{K}).$$

Discretisation of this problem is given by

$$\tilde{\sigma}_l \in Q_p(\hat{K}): \qquad (\hat{\nabla}\tilde{\sigma}_l, \hat{\nabla}\phi_{4p+i})_{\hat{K}} = -(\hat{\nabla} g_l, \hat{\nabla}\phi_{4p+i})_{\hat{K}} \quad \forall i = 1, \ldots, (p-1)^2.$$

Recalling definitions (C.1), (C.5) and (C.4) gives

$$\sum_{j=1}^{(p-1)^2} S_{ij}(p_{4p+j})_l = -\sum_{k=1}^{4p} T_{ik}(p_k)_l, \quad i = 1, \ldots, (p-1)^2, \tag{C.7}$$

with the matrices $S_{ij} \in \mathbb{R}^{(p-1)^2 \times (p-1)^2}$ and $T_{ik} \in \mathbb{R}^{(p-1)^2 \times 4p}$ given by

$$S_{ij} = (\hat{\nabla}\phi_{4p+i}, \hat{\nabla}\phi_{4p+j})_{\hat{K}}, \quad i, j = 1, \ldots, (p-1)^2,$$

and

$$T_{ik} = (\hat{\nabla}\phi_{4p+i}, \hat{\nabla}\phi_k)_{\hat{K}}, \quad i = 1, \ldots, (p-1)^2, \quad k = 1, \ldots, 4p.$$

The solutions to problem (C.7) for $l = 1, 2$ are

$$(p_{4p+j})_l = -\sum_{i=1}^{(p-1)^2} \sum_{k=1}^{4p} S_{ji}^{-1} T_{ik}(p_k)_l, \quad j = 1, \ldots, (p-1)^2,$$

that may be written in compact form:

$$p_{4p+j} = \sum_{k=1}^{4p} c_{jk} p_k, \quad j = 1, \ldots, (p-1)^2, \tag{C.8}$$

where c_{jk} represents the coefficient

$$c_{jk} = -\sum_{i=1}^{(p-1)^2} S_{ji}^{-1} T_{ik}$$

of the linear combination (C.8), that represents the dependency of the jth inner mapping support point p_{4p+j} on the support points p_k, $k = 1, \ldots, 4p$, that are placed on the boundary of the element K. For a fixed degree p, these coefficients c_{jk} are the same for the mapping of *all* cells K in real space because the c_{jk} depend only on the reference element $\hat{K}$. Therefore the coefficients c_{jk} can be pre-computed and result in following linear combinations.

For $p = 2$ the linear combination turns out to be

$$p_9 = \tfrac{1}{16} \sum_{k=1}^{4} p_k + \tfrac{3}{16} \sum_{k=5}^{8} p_k,$$

see also Figure C.4(a). For $p = 3$ the coefficients $c_{13,k}$ of the linear combination for the inner mapping support point p_{13} is shown in Figure C.4(b). The coefficients for the points p_{14}, p_{15} and p_{16} may be obtained by rotation of the coefficients.

In Figure C.5 we show the images of the Q_1-Q_4 mappings applied to the unit cell $\hat{K}$ that, like in the previous section, is covered by a tensor product mesh. While the corresponding images in Figure C.3 show degenerated meshes for Q_p mappings of degrees $p = 3$ and $p = 4$, here in Figure C.5, no such behaviour occurs. Indeed, the images of the Q_3 and the Q_4 mappings are as smooth as the one of the Q_2 mapping.

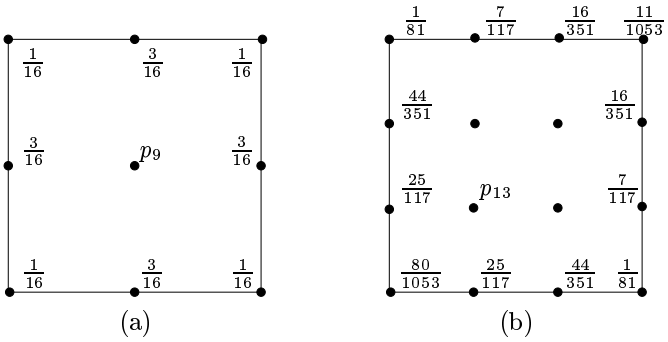


Figure C.4: (a) Coefficients $c_{9,k}$ for a Q_2 mapping. (b) Coefficients $c_{13,k}$ for a Q_3 mapping.

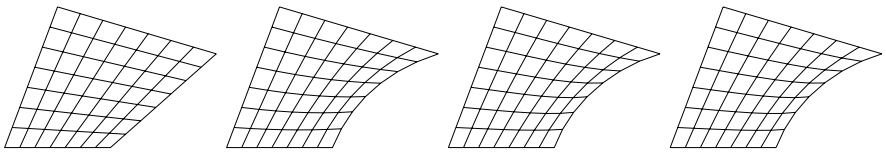


Figure C.5: $Q_1 - Q_4$ mapping of a cell. Inner mapping support points are computed by a smooth transformation.

Bibliography

[1] I. H. Abott and A. E. von Doenhoff. *Theory of Wing Sections*. Dover Publications, Inc., New York, 1959.

[2] A selection of experimental test cases for the validation of CFD codes. AGARD Report AR-303, 1994.

[3] W. Bangerth, R. Hartmann, and G. Kanschat. `deal.II` *Differential Equations Analysis Library, Technical Reference*. IWR, Universität Heidelberg, Apr. 2002. `http://www.dealii.org/`.

[4] F. Bassi and S. Rebay. High-order accurate discontinuous finite element solution of the 2d Euler equations. *J. Comput. Phys.*, 138:251–285, 1997.

[5] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: Basic analysis and examples. *East–West J. Numer. Math.*, 4:237–264, 1996.

[6] R. Becker and R. Rannacher. Weighted a posteriori error control in FE methods. In e. a. H. G. Bock, editor, *ENUMATH 95*, pages 621–637, Paris, September 1998. World Scientific Publ., Singapure. in [8].

[7] R. Becker and R. Rannacher. An optimal control approach to error estimation and mesh adaptation in finite element methods. *Acta Numerica*, 10:1–102, 2001.

[8] H. G. Bock, F. Brezzi, R. Glowinsky, G. Kanschat, Y. A. Kuznetsov, J. Priaux, and R. Rannacher, editors. *ENUMATH 97, Proceedings of the 2nd European Conference on Numerical Mathematics and Advanced Applications*, Singapore, 1998. World Scientific.

[9] F. Bouchut and F. James. One–dimensional transport equations with discontinuous coefficients. *Nonlinear Analysis TMA*, 14:255–283, 1998.

[10] F. Brezzi, P. Houston, D. Marini, and E. Süli. Modeling subgrid viscosity for advection–diffusion problems. *Comput. Meth. Appl. Mech. Engrg.*, 190(13-14):1601–1610, 2000.

[11] E. Brooksbanks. *A Numerical Investigation of Time Resolved Flows Around Turbine Blades*. PhD thesis, University of Leicester, 2001.

[12] J. M. Buckley and M. Leverett. Mechanism of fluid displacement in sands. *Trans. AIME*, 146:107–116, 1942.

[13] G. Chiocchia. Exact solutions to transonic and supersonic flows. Technical Report AR-211, AGARD, 1985.

[14] B. Cockburn, S. Hou, and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case. *Math. Comp.*, 54:545–581, 1990.

[15] B. Cockburn, G. Karniadakis, and C.-W. Shu. The development of discontinuous Galerkin methods. In B. Cockburn, G. Karniadakis, and C.-W. Shu, editors, *Discontinuous Galerkin Methods*, volume 11 of *Lecture Notes in Computational Science and Engineering*, pages 3–50. Springer, 1999.

[16] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, editors. *Discontinuous Galerkin Methods*, volume 11 of *LNCSE*. Springer, 2000.

[17] B. Cockburn and C. W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems. *J. Comput. Phys.*, 84:90–113, 1989.

[18] B. Cockburn and C.-W. Shu. The Runge–Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems. *J. Comput. Phys.*, 141:199–224, 1998.

[19] L. Elsner and V. Mehrmann. Convergence of block iterative methods for linear systems arising in the numerical solution of Euler equations. *Numer. Math.*, 59:541–559, 1991.

[20] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, pages 105–158, 1995.

[21] C. Fletcher. *Computational Techniques for Fluid Dynamics*, volume 1,2. Springer, 1997.

[22] C. Führer and R. Rannacher. An adaptive streamline–diffusion finite element method for hyperbolic conservation laws. *East–West J. Numer. Math.*, 5(3):145–162, 1997.

[23] E. Godlewski, M. Olazabal, and P.-A. Raviart. On the linearization of systems of conservation laws for fluids at a material contact discontinuity. *J. Math. Pures Appl.*, 17:1013–1042, 1999.

[24] R. Hartmann. Adaptive FE Methods for Conservation Equations. In H. Freistühler and G. Warnecke, editors, *Hyperbolic Problems: theory, numerics, applications: eighth international conference in Magdeburg, February, March 2000*, volume 2 of *International series of numerical mathematics; Vol. 141*, pages 495–503. Birkhäuser, Basel, 2001.

[25] R. Hartmann and P. Houston. A posteriori error estimation for nonlinear hyperbolic conservation laws involving discontinuous solutions. In preparation.

[26] R. Hartmann and P. Houston. Adaptive discontinuous Galerkin finite element methods for nonlinear hyperbolic conservation laws. Preprint 2001-20, (SFB 359), IWR Heidelberg, Mai 2001. To appear in SIAM J. Sci. Comp.

[27] R. Hartmann and P. Houston. Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. Preprint 2001-42, (SFB 359), IWR Heidelberg, Dez 2001. Submitted to J. Comp. Phys.

[28] R. Hartmann and P. Houston. Goal-oriented a posteriori error estimation for multiple target functionals. In *Hyperbolic problems: theory, numerics, applications: ninth international conference in Pasadena, California, 2002.* Springer, 2002. In preparation.

[29] F.-K. Hebeker and R. Rannacher. An adaptive finite element method for unsteady convection–dominated flows with stiff source terms. *SIAM J. Sci. Comp.*, 21:799–818, 1999.

[30] P. Houston and R. Hartmann. Goal–oriented a posteriori error estimation for compressible fluid flows. In *Proceedings of ENUMATH 2001*, 2001. Submitted.

[31] P. Houston, R. Hartmann, and A. Süli. Adaptive discontinuous Galerkin finite element methods for compressible fluid flows. In M. Baines, editor, *Numerical methods for Fluid Dynamics VII, ICFD*, pages 347–353, 2001.

[32] P. Houston, R. Rannacher, and E. Süli. A posteriori error analysis for stabilised finite element approximations of transport problems. *Comput. Meth. Appl. Mech. Engrg.*, 190(11-12):1483–1508, 2000.

[33] P. Houston, C. Schwab, and E. Süli. Stabilized hp-finite element methods for first–order hyperbolic problems. *SIAM J. Numer. Anal.*, 37(5):1618–1643, 2000.

[34] P. Houston and E. Süli. hp–adaptive discontinuous Galerkin finite element methods for first–order hyperbolic problems. *SIAM J. Sci. Comp.*, 23:1225–1251, 2001.

[35] J. Jaffre, C. Johnson, and A. Szepessy. Convergence of the discontinuous Galerkin finite element method for hyperbolic conservation laws. *Math. Models and Methods in Appl. Sciences*, 5:367–386, 1995.

[36] C. Johnson. *Finite Element Methods for Partial Differential Equations.* Cambridge University Press, 1993.

[37] C. Johnson, U. Nävert, and J. Pitkäranta. Finite element methods for linear hyperbolic problems. *Comp. Mech. in Appl. Mech. Engrg.*, 45:285–312, 1984.

[38] C. Johnson and J. Pitkäranta. An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation. *Math. Comp.*, 46(173):1–26, 1986.

[39] D. Kröner. *Numerical Schemes for Conservation Laws.* Wiley-Teubner, 1997.

[40] M. G. Larson and T. J. Barth. A posteriori error estimation for adaptive discontinuous Galerkin approximations of hyperbolic systems. In B. Cockburn, G. Karniadakis, and C.-W. Shu, editors, *Discontinuous Galerkin Methods*, volume 11 of *Lecture Notes in Computational Science and Engineering*, pages 363–368. Springer–Verlag, 2000.

[41] P. LeSaint and P. A. Raviart. On a finite element method for solving the neutron transport equation. In C. de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 89–145. Academic Press, 1974.

[42] D. W. Levy, T. Zickuhr, J. Vassberg, S. Agrawal, R. A. Wahls, S. Pirzadeh, and M. J. Hemsch. Summery of data from the frist AIAA CFD drag prediction workshop. *AIAA*, 2002-0841, 2002.

[43] T. E. Peterson. A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation. *SIAM J. Numer. Anal.*, 28(1):133–140, 1991.

[44] M. Rakowitz, M. Sutcliffe, B. Eisfeld, D. Schwamborn, H. Bleecke, and J. Fassbender. Structured and unstructured computations of the DLR-F4 wing-body configuration. *AIAA*, 2002-0837, 2002.

[45] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[46] G. Richter. An optimal–order error estimate for the discontinuous Galerkin method. *Math. Comp.*, 50:75–88, 1988.

[47] F. Ringleb. Exakte Lösungen der Differentialgleichungen einer adiabatischen Gasströmung. *ZAMM*, 20(4):185–198, 1940.

[48] C. Schwab. *p- and hp-Finite Element methods. Theory and Applications to Solid and Fluid Mechanics.* Oxford University Press, 1998.

[49] H. Schwarz. *Methode der finite Elemente.* B.G. Teubner Stuttgart, 1991.

[50] J. L. Steger and R. Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. *J. Comput. Phys.*, pages 263–293, 1981.

[51] E. Süli. A posteriori error analysis and adaptivity for finite element approximations of hyperbolic problems. In D. Kröner, M. Ohlberger, and C. Rhode, editors, *An introduction to recent developments in theory and numerics for conservation laws*, volume 5 of *Lecture Notes in Comput. Sciences and Engrg.*, pages 123–194. Springer, 1998.

[52] E. Süli and P. Houston. Adaptive finite element approximation of hyperbolic problems. Technical Report 2002/08, Leicester University, 2002. Submitted for publication 2002 to NASA Ames/VKI Lecture Series on Error Estimation and Solution Adaptive Discretization in CFD.

[53] E. Tadmor. Local error estimates for discontinuous solutions of nonlinear hyperbolic equations. *SIAM J. Numer. Anal.*, 28:891–906, 1991.

[54] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer, 1997.

[55] S. Ulbrich. *Optimal control of nonlinear hyperbolic conservation laws with source terms.* Habilitation thesis, Faculty for Mathematics, Munich Technical University, 2001.