Shape Derivatives and Shock Capturing for the Navier–Stokes Equations in Discontinuous Galerkin Methods

A thesis accepted by the Faculty of
Aerospace Engineering and Geodesy of the University of Stuttgart
in partial fulfillment of the requirements for the degree of
Doctor of Engineering Sciences (Dr.-Ing.)

by

Matthias Sonntag

born in Stuttgart

Main referee:	Prof. Dr. Claus-Dieter Munz
Co referee:	Prof. Dr. Nicolas R. Gauger
Date of defence:	August 10, 2017

Institute of Aerodynamics and Gas Dynamics
University of Stuttgart

2017

Für Patrizia

Preface

This thesis was developed during my work as academic employee at the Institute of Aerodynamics and Gas Dynamics (IAG) of the University of Stuttgart, the mathematics division of the center for Computational Engineering Science (MathCCES) of the RWTH Aachen University and the Chair for Scientific Computing (SciComp) of the University of Kaiserslautern.

I thank my doctoral supervisors Prof. Dr. Nicolas R. Gauger and Prof. Dr. Claus-Dieter Munz for their unconditional support during the last five years. Both granted me all scientific freedom and made it possible to work on different topics and finally even in a joint project. It was a great pleasure being associated to both of their research groups.

Many thanks also to all my colleagues at IAG, MathCCES and SciComp for the friendly working atmosphere, the scientific discussions and for sharing their knowledge with me. In particular, I would like to thank Max Sagebaum for all the conversations about computers, programming skills and non-scientific topics, but especially for the accommodation in his home on my trips to Aachen and Kaiserslautern. Furthermore I thank Dr. Stefan Fechter, Dr. Andrea Beck, Serena Keller and Timon Hitz for being kind roommates or reviewing this work.

This thesis was developed within projects financed by the German ministry for education and research (BMBF) and the Deutsche Forschungsgemeinschaft (DFG).

Last but not least I am extremely grateful for all the love and support of my wife Patrizia and my family.

Stuttgart, August 16, 2017

Matthias Sonntag

Contents

Symbols and abbreviations

Symbols

a^i	i-th contravariant basis vector
a_j	j-th covariant basis vector
$\alpha_{RK}(N)$	Scaling factor of the Runge-Kutta time integration for the advective time step
$\beta_{RK}(N)$	Scaling factor of the Runge-Kutta time integration for the viscous time step
c	Speed of sound
C_∞	Scaling factor of the cost function to reference length and freestream conditions
c_v	Heat capacity at constant volume
δ_{ij}	Kronecker delta
Δt	Time step
Δx	Minimum physical element size
div	Divergence, same as $\nabla\cdot$
div_Γ	Tangential divergence
$d\mathcal{J}(\Omega;V)$	Shape derivative of cost function
$d_V[\cdot]$	Material derivative
E	Reference element
e	Total energy
ϵ	Inner energy
ε	Step size of finite differences
η	Second component of coordinate in reference space
$\mathcal{F}$	Flux in reference space ($\mathcal{F} = \mathcal{F}^c + \mathcal{F}^v$)
$\mathcal{F}^c$	Convective flux in reference space
F	Flux in physical space ($F = F^c + F^v$)
F^c	Convective flux in physical space
F^v	Viscous flux in physical space
$\hat{\mathcal{F}}_{ijk}$	Nodal interpolation of the flux
$(F \cdot n)^*$	Numerical flux
f^*	Riemann solver

$F_{\mathbf{u}}$	Derivative of the flux with respect to the state
$\mathcal{F}^v$	Viscous flux in reference space
$\mathcal{F}^v_{\nabla\mathbf{u}}$	Derivative of the viscous flux with respect to the gradient of the state
G	Homogenity tensor
Γ	Boundary of computational domain
γ	Heat capacity ratio
Γ_{adia}	Adiabatic wall boundary
Γ_∞	Farfield boundary
Γ_{iso}	Isothermal wall boundary
Γ_W	Wall boundary
∇_Γ	Tangential gradient
H	Enthalpy
h	Grid size of structured block
I	Interpolation operator
$\mathcal{I}$	Indicator function
J	Jacobian determinant of the mapping from reference to physical space
$\mathcal{J}$	Target/cost function
K	additive curvature ($\mathrm{div}_\Gamma\, n$)
κ_{ij}	ij-th Finite Volume sub-cell of a DG element
ℓ_i	i-th Lagrange interpolation polynomial
Λ	Limiter function
λ^c	Maximum signal velocity of the inviscid Euler equations
λ^v	Maximum eigenvalue of the diffusion matrix
N	Polynomial degree
$\mathcal{N}$	Normal vector in reference space
n	Normal vector in physical space
Ω	Computational domain
$\partial\Omega$	Boundary of computational domain
P	Projection operator
p	Pressure
Φ	Test function
Pr	Prandtl number
ψ	Tensor product basis function
R	Ideal gas constant
ρ	Density
$\cdot'$	Shape derivative
$\hat{s}$	Surface element

Σ	Adjoint stress tensor
$\cdot^{-}$	Indicates the master side of an interface
$\cdot^{+}$	Indicates the slave side of an interface
T	Temperature
t	Time
τ	Viscous stress tensor
θ	Direction of the force of the cost function (drag or lift)
$T_t[V]$	Perturbation of identity into direction V
$\mathbf{u}$	State vector of conservative variables
$\hat{\hat{\mathbf{u}}}_{ijk}$	Nodal degrees of freedom of FV method
$\hat{\mathbf{u}}_{ijk}$	Nodal degrees of freedom of DG method
V	Vector field of perturbation of identity
v	Vector of velocity components
v_1, v_2, v_3	Velocity components
μ	Dynamic viscosity
ω_i	i-th weight of the Gauss integration
w	Size of an equidistant FV sub-cell in reference space
X	Mapping from reference to physical space
x	Coordinate in physical space
x^1, x^2, x^3	Components of coordinate in physical space
ξ	Coordinate in reference space
ξ^1, ξ^2, ξ^3	Components of coordinate in reference space
y	Second component of coordinate in physical space
$\mathbf{z}$	Adjoint solution

Abbreviations

BR1	Lifting scheme of Bassi and Rebay
BR2	Lifting scheme of Bassi and Rebay
CFD	Computational fluid dynamics
CFL	Courant, Friedrichs and Lewy number
DFL	Counterpart of the CFL number for the viscous time step restriction
DG	Discontinuous Galerkin
DGSEM	Discontinuous Galerkin Spectral Elements Method
DOF(s)	Degree(s) of freedom
FD	Finite Difference
FE	Finite Element
FV	Finite Volume

HLLC	Harten, Lax, van Leer - contact Riemann solver
HLLE	Harten, Lax, van Leer and Einfeldt Riemann solver
HPC	High performance computing
IAG	Institute of Aero- and Gasdynamics
JST	Jameson-Schmidt-Turkel scheme
MPI	Message passing interface
NRG	Numerics Research Group
NSE	Navier-Stokes equations
PDE	Partial differential equation
PID	Performance index
RK	Runke-Kutta time integration
TVD	Total Variation Diminishing
Vdm	Vandermonde matrix
WENO	Weighted essentially non-oscillatory

Kurzfassung

Diese Arbeit befasst sich mit zwei unterschiedlichen Themengebieten, mit den Formableitungen für die kompressiblen Navier-Stokes-Gleichungen einerseits und andererseits mit der Behandlung von Stößen oder anderer Strömungsunstetigkeiten in Discontinuous-Galerkin-Verfahren. In der Luftfahrtindustrie werden für die Formoptimierung, beispielsweise die Widerstandsreduktion oder Auftriebsmaximierung von Flugzeugen, äußerst effiziente Verfahren benötigt. Der Einsatz gradientenbasierter Optimierungsverfahren erfordert dabei Ableitungen der Zielfunktionen nach der Form eines Objekts. Mit den hier vorgestellten Formableitungen können diese Ableitungen unabhängig von der gewählten Parametrisierung der Form bestimmt werden und, da die Herleitung im Kontinuierlichen erfolgt, auf nahezu jede Diskretisierung angewendet werden. Dennoch kann die später verwendete numerische Methode nicht außer Acht gelassen werden. Für Verfahren, die auf einer variationellen Formulierung basieren, ergibt sich, im Vergleich zu einer punktweisen Betrachtung, ein Unterschied in der Formableitung, der nicht vernachlässigt werden kann. Ein Ziel dieser Arbeit ist es daher, die Formableitung des Widerstands- und Auftriebsbeiwertes für die Navier-Stokes-Gleichungen in variationeller Form herzuleiten und sowohl analytisch als auch numerisch mit einem punktweisen Ansatz zu vergleichen. Dabei ist eine Diskrepanz insbesondere für Strömungsphänomene mit starken Gradienten oder Unstetigkeiten, die nicht die starke Form der Erhaltungsgleichungen erfüllen, zu erwarten.

Diese Strömungsphänomene erfordern, bei Verwendung von Verfahren hoher Ordnung, prinzipiell eine spezielle numerische Behandlung. Im zweiten Teil dieser Arbeit wird daher ein Shock-Capturing für das Discontinuous-Galerkin-Verfahren (DG) entwickelt, das die Oszillationen, die durch die Approximation von Unstetigkeiten mit Polynomen hoher Ordnung entstehen, abfängt. Dazu wird in einem hybriden Ansatz das DG-Verfahren mit einem Finite-Volumen-Verfahren zweiter Ordnung gekoppelt. In Gitterzellen, die Stöße oder Unstetigkeiten enthalten, wird der DG-Operator durch die FV-Methode ersetzt, die aufgrund der Steigungslimitierung für ihre Stärken hinsichtlich von Stößen bekannt ist. Allerdings erfordert das Finite-Volumen-Verfahren in glatten Strömungsgebieten eine wesentlich höhere Auflösung als die DG-Methode. Um den Auflösungsverlust, der bei Verwendung desselben Gitters für das FV-Verfahren wie für die DG-Methode entstehen würde, zu kompensieren, werden daher die ursprünglichen Gitterzellen in logische

Subzellen unterteilt. Durch die Zuordnung von genau einer FV-Subzelle zu jedem Freiheitsgrad eines DG-Elements können die selben Datenstrukturen weiterverwendet werden. Dies ermöglicht eine effiziente Implementierung des skizzierten Shock-Capturings für den Einsatz auf Hochleistungsrechnern. Anhand zahlreicher Beispiele werden daher nicht nur die grundlegenden Eigenschaften dieser hybriden DG/FV-Subzellen-Methode analysiert, sondern auch Skalierungsstudien hinsichtlich der parallelen Effizienz durchgeführt.

Abstract

This work addresses two different topics, the shape derivatives for the compressible Navier–Stokes equations on the one hand and, on the other hand, the treatment of shocks or other flow discontinuities in Discontinuous Galerkin methods. There is a strong demand for very efficient methods for shape optimization in the aerospace industry, for example drag reduction or lift maximization of an aircraft. The use of gradient based optimization schemes requires derivatives of the cost function with respect to the shape of an object. With the shape derivatives presented in this work, these derivatives can be calculated independent of the parametrization of the object's shape, and, since the derivation takes place in the continuous space, they can be applied to almost any discretization. Nevertheless, one has to take the numerical scheme, which is later applied, into account. For methods based on the variational formulation a difference in the shape derivative, compared to the pointwise approach, arises, which cannot be neglected. Hence, one objective of this work is to derive the shape derivatives of the drag- and lift-coefficient for the Navier–Stokes equations in variational formulation and compare it with the pointwise approach both analytically and numerically. A discrepancy has to be expected, especially for flow phenomena with high gradients or discontinuities which do not fulfill the strong form of the governing equations.

These flow phenomena require a special treatment in numerical methods of high order. In the second part of this work, a shock capturing for the Discontinuous Galerkin method is developed which prevents the oscillations originating from the approximation of discontinuities with high order polynomials. Therefore a hybrid approach is presented, where the original DG scheme is coupled with a second order Finite Volume method. In all elements containing shocks or discontinuities the operator of the DG method is replaced by the Finite Volume scheme. This scheme is, due to the use of slope limiters, well known for its strengths in handling shocks. However, in regions where the flow is smooth the Finite Volume method requires a finer resolution for the same accuracy than the Discontinuous Galerkin scheme. Using the same mesh for the FV method as for the DG scheme would lead to a big reduction in resolution. Hence, to compensate this loss the original elements of the mesh are divided into logical sub-cells. By associating exactly one Finite Volume sub-cell to each degree of freedom of a DG element, the same data structures can be used. This enables an efficient implementation of the outlined shock capturing designated for high performance computations. Therefore, not only the basic properties of this hybrid DG/FV sub-cell approach are investigated with several examples, but also studies regarding the parallel efficiency are performed.

1. Introduction

In the field of computational fluid dynamics (CFD), the simulation of flows is generally well established not only in research, but also in industry. Aircraft manufacturers for example use CFD in the design stage to predict very precisely aerodynamic performances, like drag or lift of a wing or even a full aircraft configuration. With the growth of computing capacity over the past decades simulations became more and more detailed, and besides the pure simulation of a given design the optimization of specific cost functions with respect to the shape is nowadays possible. For these optimization two main categories of optimization methods are available.

Gradient based methods that use the direction of the steepest descent of the cost function on the one hand and on the other hand heuristic methods that are derivative-free. The latter type of optimization is widely used in practice, since it can be applied to more or less any kind of optimization problem. This independence is gained through not exploiting the structure of a given problem, but instead sampling the parameter space. At the same time this freedom is one of the drawbacks that makes an application to industry size cases very challenging and time consuming. One subclass of the heuristic methods are evolutionary algorithms which are inspired by mechanisms of the biological evolution, such as reproduction, mutation and survival of the fittest [4]. They require many populations to find an optimal solution. Additionally, the number of required iterations increases drastically with the number of design parameters, which become the key factor for the numerical costs. In the field of aerodynamic shape optimization it is therefore common, to restrict the modifications of the shape to a small set of smooth design parameters, like B-splines or the popular Hicks-Henne functions [36]. This restriction to only a few parameters on the one hand speeds up the optimization, but on the other hand it limits the optimum of the achievable shape. The same problem occurs for gradient based methods, if the derivatives are evaluated by finite differences. To evaluate the sensitivities of the cost function with respect to all parameters, a flow simulation for each of the individually disturbed design parameters is required.

Alternative concepts where the computational effort of an optimization does not depend on the number of design parameters are required. For this purpose the introduction of adjoint calculus by Pironneau in 1973 [65] initiated a major advance in the field of fluid dynamics. Jameson [46] as well as Giles and Pierce [31, 32] further extended the application of adjoint equations to the aeronautical computational fluid dynamics. More recently, this has been applied to very large scale aerodynamic design optimization [63, 96, 74].

Another active field of research to prevent expensive computations for a large number of design variables is the mathematical framework of shape calculus, where derivatives with respect to the shape are derived for the continuous equations. Hence, the resulting shape derivatives are independent from the discretization and thereby from the number of design parameters. A detailed introduction into the general theoretical framework of shape calculus is given by Sokolowski and Zolésio [83] as well as Delfour and Zolésio [22]. Following this approach the shape derivatives have already been applied to viscous incompressible and compressible flow [13, 75], but since they are all based on a pointwise formulation of the governing equations they assume the existence of a strong form solution. However, the existence of such a solution is not clear in general and especially not in the presence of shock waves or flow discontinuities. In this case, one is restricted to weak solutions of the governing equations in variational form. Actually, this is not a major drawback for numerical schemes which are based on this variational formulation and compute only weak solutions. But unfortunately, these weak solutions do not necessarily fulfill the strong equations in a pointwise manner and, hence, there is a small consistency gap when the weak numerical solution is used to evaluate the shape derivatives stemming from a pointwise derivation. To overcome this, the whole calculus of the shape derivatives must be based on the variational form of the partial differential equations, which is actually only rarely considered in literature [43, 42, 82]. Therefore, one of the main topics in this work is the investigation of the differences between shape derivatives based on the pointwise Navier–Stokes equations and shape derivatives in the variational setting. After introducing some fundamentals in chapter 2, the shape derivatives of the drag and lift coefficient for the Navier–Stokes equations are derived analytically in chapter 3. Additionally, both approaches are investigated numerically to demonstrate the mismatch of the pointwise shape derivatives for a numerical scheme based on the variational formulation.

Regarding the numerical scheme for this research, one is not bound to a specific method and can freely choose an appropriate scheme since the entire concept of shape calculus takes place in the continuous setting. For the same reason the evaluation of the derivatives of the cost function with respect to the shape is independent of the number of design parameters, but nevertheless the optimization still requires the computation of multiple flow solution. Hence, for large scale cases the efficiency of the numerical method is a major issue. Due to favorable properties, the Discontinuous Galerkin (DG) scheme has become popular for the aeronautical computational fluid dynamics in the recent years. Originally introduced by Reed and Hill [68] in 1973, the Discontinuous Galerkin method is a combination of a local Finite Element (FE) and a Finite Volume (FV) scheme. The solution is approximated with high order polynomial ansatz functions, but in contrast to the Finite Element method these are totally element local. Therefore, the elements are in principle not coupled, leading to only piecewise continuous data where the states at the interfaces are double valued. Nevertheless the elements have to be connected, where the Finite Volume method comes into play. The discontinuities at the element interfaces are interpreted as Riemann problems and the numerical fluxes computed with Riemann solvers become the link between the elements. This idea of coupling different domains weakly was introduced by Nitsche [61] even earlier in 1971. After this first work it took quite a long time, until in the late 1980's Cockburn and Shu started their series of papers [15, 16, 18, 19, 20] where they extended the Discontinuous Galerkin method to systems of non-linear conservation laws. Furthermore, Bassi and Rebay [7, 8] applied the DG scheme to the compressible Navier–Stokes equations by transforming the parabolic second order terms into a system of first order equations.

One of the key features of the Discontinuous Galerkin method is the unrestricted polynomial degree of the ansatz functions, which theoretically generates schemes of arbitrary high order. In practice there are restrictions due the floating point arithmetic of computers for very large polynomial degrees. Nevertheless, the method is very efficient for smooth regions in terms of the points required to resolve a wave of given length [28]. However, this is not the case for shock waves or other non-smooth flow discontinuities where the high order polynomials suffer from the Gibbs phenomenon [30] and will produce spurious oscillations which may violate the physics by generating negative pressure for example. This will eventually let the computation fail. To cure the simulation from this misbehavior, several different techniques, known as shock capturing, are available. Already in 1950, von Neumann and Richtmyer [90] proposed the idea of adding artificial viscosity to the original equation with the intention to smear the discontinuity until the solution could be properly resolved by their numerical Finite Differencing scheme. This approach was adapted by Persson and Peraire [64] to high order Discontinuous Galerkin methods, where an elementwise constant artificial viscosity is used to capture shocks without widening them over several cells. In [6], a smooth artificial viscosity PDE model is used to further improve this technique and reduce the effects that occur due to non constant viscosity at element interfaces. Another approach for shock capturing in high order methods is to directly filter the degrees of freedom (DOFs) of the solution. Yee et al. [94]

introduced nonlinear filters for high order finite differencing schemes to prevent the generation of oscillations at shock fronts. This adaptive control mechanism of the numerical dissipation can quite generally be applied to high order methods and it was adapted by Panourgias and Ekaterinaris [62] for the Discontinuous Galerkin scheme. A third class of shock capturing techniques are hybrid methods, where the DG method is combined with a method classically designed for shock involving computations. Either the original scheme is fully replaced in troubled regions or the additional method is utilized as a limiter. The latter approach is for example proposed by Qiu and Shu [67] by using a Hermite weighted non-oscillatory (HWENO) scheme for the limiting of Runge-Kutta DG methods in shock regions. A full replacement of the scheme in all problematic elements is presented by Dumbser et al. [24]. They apply the MOOD paradigm [59] which a posteriori detects nonphysical candidate solutions and recomputes the troubled cells with a lower order method on a sub-cell mesh. This is a quite expensive but robust variant of the classical *h*-/*p*-refinement where the polynomial degree of elements containing a discontinuity is reduced, since they are less susceptible to the Gibbs phenomenon. At the same time the mesh is refined to compensate the loss in overall resolution. Huerta et al. [41] adapted this idea of spatial refinement with low order polynomials for the Discontinuous Galerkin scheme by extending the space of ansatz functions with piecewise constant functions in sub-cells.

In chapter 5 several of the concepts presented above are revisited to develop a shock capturing for the Discontinuous Galerkin Spectral Elements Method, which is summarized in chapter 4. This shock capturing is a hybrid approach, where the Finite Volume method takes care of the troubled cells. Due to the low order (second order with reconstruction) and its total variation diminishing property, provided by slope limiters, this scheme is spared from the generation of oscillations at discontinuities. Except for the requirement of finer grids than for the DG method, the Finite Volume scheme is ideally suited to resolve shocks and other flow discontinuities. This demand on the resolution is handled by introducing a logical sub-cell refinement of the original elements. Since special attention is paid to an efficient implementation of the shock capturing for high performance computations, to each DOF of a DG element exactly one Finite Volume sub-cell is associated. The properties of this shock capturing for the Discontinuous Galerkin method using Finite Volume sub-cells are examined in chapter 6 with several test cases numerically. In addition, the parallel efficiency is investigated to proof the capability of the hybrid scheme for high performance computations.

In summary, the two main objectives of this work are as follows. In the field of shape optimization the shape derivatives of the lift and drag coefficient for the compressible Navier–Stokes equations will be derived, where special attention is paid to the form of the governing equations. A comparison of the variational approach with the former approach, based on the pointwise Navier–Stokes equations, will demonstrate that numerical schemes based on the variational form of the governing equations require a derivation of the shape derivatives which is suitable to the form of the Navier–Stokes equations. The second objective addresses the treatment of shocks or other flow discontinuities in Discontinuous Galerkin methods. Therefore a shock capturing based on the Finite Volume method will be presented. This scheme prevents the oscillations originating from the approximation of discontinuities with high order polynomials by reducing the polynomial degree of the ansatz functions. This leads to a loss in resolution which is compensated by dividing the original elements into sub-cells. One main focus of this hybrid scheme will be on the computational efficiency to obtain an algorithm that is qualified for high performance computing of large scale problems.

2. Fundamentals

In this chapter the basic ingredients for the text at hand are summarized. The fundamental conservation laws in this work are the Navier–Stokes equations. They are introduced in the pointwise form and are additionally given in the variational formulation, which is necessary for the derivation of shape derivatives as well as for the Discontinuous Galerkin and Finite Volume scheme. Furthermore, they are mapped from physical space to a reference element on which the numerical methods operate.

2.1. Navier–Stokes equations

The compressible Navier–Stokes equations [60, 87] describe the motion of viscous fluids and gases and are an extension of the Euler equations [25] by second order terms. They are conservation laws for mass, momentum and energy, which means the total amounts of these physical quantities do not change over time within the considered volume if the boundary fluxes are zero. Considering an infinitesimal fluid particle leads to the pointwise or strong form of the Navier–Stokes equations, where they are fulfilled in every point of the domain Ω. This pointwise form, is given by

$$\mathbf{u}_t + \nabla \cdot (F^c(\mathbf{u}) - F^v(\mathbf{u}, \nabla \mathbf{u})) = 0 \quad \text{in } \Omega, \tag{2.1}$$

where $\mathbf{u}$ and $\mathbf{u}_t$ denote the vector of conservative variables and its temporal derivative, $F^c(\mathbf{u}) = (f_1^c, f_2^c, f_3^c)$ the convective fluxes and $F^v(\mathbf{u}, \nabla \mathbf{u}) = (f_1^v, f_2^v, f_3^v)$ the viscous fluxes. They are defined in three space dimensions by

$$\mathbf{u} = \begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ \rho e \end{pmatrix}, \quad f_i^c = \begin{pmatrix} \rho v_i \\ \rho v_i v_1 + \delta_{1i} p \\ \rho v_i v_2 + \delta_{2i} p \\ \rho v_i v_3 + \delta_{3i} p \\ \rho H v_i \end{pmatrix}, \quad f_i^v = \begin{pmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \sum_j \tau_{ij} v_j + \kappa \frac{\partial T}{\partial x_i} \end{pmatrix} \qquad i = 1, 2, 3$$

where δ_{ij} is the Kronecker delta and the viscous stress tensor τ is given by

$$\tau = \mu \left(\nabla v + (\nabla v)^\top - \frac{2}{3} (\nabla \cdot v) I \right) \qquad I = \text{identity matrix.}$$

In table 2.1 the remaining physical quantities and some relations between them are summarized. To close the Navier–Stokes equations, a relation between the thermodynamic quantities ρ, p and T is necessary. For an ideal gas, this equation of state reads

$$p = \rho RT = \rho R \frac{e - \frac{1}{2}v^2}{c_v} = \frac{R}{c_v}\rho\left(e - \frac{1}{2}v^2\right).$$

ρ	density	
v_1, v_2, v_3	velocity	
p	pressure	
ϵ	inner energy	
e	total energy	$e = \epsilon + \frac{1}{2}v^2$
H	enthalpy	$H = e + \frac{p}{\rho}$
μ	viscosity	
τ	viscous stress tensor	
T	temperature	$T\kappa = \frac{\mu\gamma}{Pr}\left(e - \frac{1}{2}v^2\right)$
κ	specific heat transfer coefficient	$\kappa = \frac{\mu\gamma c_v}{Pr}$
R	ideal gas constant	
γ	heat capacity ratio	
c_v	heat capacity at constant volume	
Pr	Prandtl number	

Table 2.1.: Physical quantities.

Remark 2.1. *Since the Navier–Stokes equations are an extension of the Euler equations, the Euler equations can be recovered by omitting the viscous flux F^v in equation* (2.1). *Both conservation laws can be reduced to the two dimensional space by deleting the equation of the momentum in z-direction and the respective convective and viscous flux.*

2.1.1. Stationary Navier–Stokes equations

A flow field is called steady if the solution does not change over time. This is equivalent to a vanishing time derivative of the state $\mathbf{u}_t$. In this case the stationary Navier–Stokes equations read

$$\nabla \cdot (F^c(\mathbf{u}) - F^v(\mathbf{u}, \nabla\mathbf{u})) = 0 \quad \text{in } \Omega. \tag{2.2}$$

2.2. Weak formulation or variational form

The shape derivative for the compressible Navier–Stokes equations in chapter 3, the Discontinuous Galerkin method in chapter 4 and the shock capturing with Finite Volume sub-cell in chapter 5 are all based on the weak formulation of the Navier–Stokes equations. To obtain the weak formulation the pointwise Navier–Stokes equations from equation (2.1) are multiplied with an arbitrary test function $\Phi \in \mathcal{H} := H^1 \times H^2 \times \ldots \times H^5$, where H^i are suitable Hilbert-spaces. Integration over the domain Ω yields

$$\int_\Omega \mathbf{u}_t \Phi \, \mathrm{d}x + \int_\Omega \nabla \cdot (F^c - F^v) \, \Phi \, \mathrm{d}x = 0.$$

After partial integration of the second integral one obtains

$$\int_\Omega \mathbf{u}_t \Phi \, \mathrm{d}x - \int_\Omega (F^c - F^v) \cdot \nabla \Phi \, \mathrm{d}x + \int_{\partial\Omega} ((F^c - F^v) \cdot n) \, \Phi \, \mathrm{d}S = 0,$$

where n is the outwards pointing normal vector.

Respectively, the weak formulation of the stationary Navier–Stokes equations becomes

$$- \int_\Omega (F^c - F^v) \cdot \nabla \Phi \, \mathrm{d}x + \int_{\partial\Omega} ((F^c - F^v) \cdot n) \, \Phi \, \mathrm{d}S = 0. \tag{2.3}$$

The weak formulation of the Navier–Stokes equations is also called variational form of the Navier–Stokes equations.

2.3. Mapping to reference space

The above conservation laws are stated in physical space in an arbitrary domain. Solving these equations numerically requires the discretization of the domain into elements. The technique to map all elements of a mesh to a reference space to obtain a general formulation for all elements is very common. In general, different types of elements, e.g. tetrahedra, pyramids, prisms, hexahedra and so on, are used to subdivide the domain, leading to different reference elements. This work restricts the elements to hexahedral meshes only, which requires only a single reference element $E = [-1, 1]^3$. Of course this limitation has some drawbacks, but it also enables very efficient implementations as will be discussed in chapter 4. One of the limitations of the hexahedral meshes is the mesh generation process for complex geometries, which is still an ongoing topic of research [37, 12, 89].

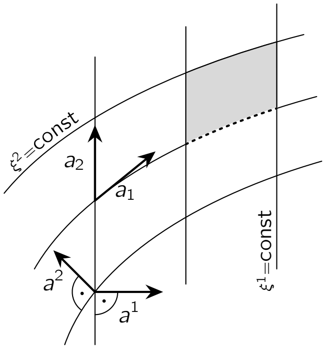


Figure 2.1.: Covariant a_1, a_2 and contravariant a^1, a^2 basis vectors in two space dimensions. The line integral along the dotted line gives the physical size of the gray element.

The physical domain Ω is divided into hexahedral elements which can be curved. The mapping between reference space $\xi = \left(\xi^1, \xi^2, \xi^3\right)^\top$ and an arbitrary element in physical space is given by

$$X : \xi \mapsto \left(x^1, x^2, x^3\right)^\top$$

Given this mapping the covariant basis vectors are given in [51, 50] by

$$a_j = \frac{\partial X}{\partial \xi^j}. \tag{2.4}$$

They are tangential to faces of the grid as shown in figure 2.1. In contrast to that the contravariant basis vectors are perpendicular to the grid faces and can be computed from the covariant basis with

$$Ja^i = a_j \times a_k \quad (i, j, k) \text{ cyclic},$$

where $J = J(\xi)$ is the Jacobian determinant of the mapping.

Definition 2.2 (Line integral)**.** *Let $r : [a, b] \to C$ be a bijective parametrization of the piecewise smooth curve C in $\mathbb{R}^d$. For a scalar field $f : \mathbb{R}^d \to \mathbb{R}$ the line integral along the curve C is defined by*

$$\int_C f \, ds = \int_a^b f(r(t)) \left\| \frac{\partial r(t)}{\partial t} \right\|_2 \, dt.$$

The special case where $f = 1$ gives the length of the curve C

$$\textit{length of } C = \int_a^b \left\| \frac{\partial r(t)}{\partial t} \right\|_2 dt.$$

Using this definition one can calculate the length of a grid line in physical space by fixing two coordinate directions of the mapping X. For example, fixing $\xi^2, \xi^3 =$ constant gives a curve $C(\xi^1)$ following the grid lines of the ξ^1 direction. This is done in the reference element $E = [-1,1]^3$ by building the line integral for ξ^1 from -1 to 1, which then results in the length of the grid line. A two dimensional example is shown in figure 2.1. Here, the south side of the gray element is marked by a dotted line. If this side corresponds to the lower side of the reference element, where $\xi^2 = -1$ holds, the length of this side can be calculated as the line integral of the curve $C\left(\xi^1\right) := X\left(\xi^1, -1\right)$, which leads to

$$\int_{C\xi^1} \mathrm{d}s = \int_{-1}^{1} \left\| \frac{\partial X(\xi^1, -1)}{\partial \xi^1} \right\|_2 \mathrm{d}\xi^1 = \int_{-1}^{1} \left\| a_1\left(\xi^1, -1\right) \right\|_2 \mathrm{d}\xi^1.$$

2.3.1. Mapping of the equations

As already mentioned, the physical domain Ω is divided into hexahedra, which are all mapped to the reference element E. Mapping the Navier–Stokes equations from a physical grid cell to the reference element they read

$$J\mathbf{u}_t + \nabla_\xi \cdot (\mathcal{F}^c - \mathcal{F}^v) = 0 \quad \text{in } E, \tag{2.5}$$

where $\mathcal{F}^c$ and $\mathcal{F}^v$ are the transformed fluxes which are defined by the transformation of the divergence operator from physical space to reference space

$$\nabla \cdot F = \frac{1}{J} \sum_{i=1}^{3} \frac{\partial J a^i \cdot F}{\partial \xi^i} =: \frac{1}{J} \nabla_\xi \cdot \mathcal{F}. \tag{2.6}$$

Here, the divergence operator is written with a ξ-subscript to indicate the derivative with respect to the reference coordinates. In the following this subscript will be omitted, as the divergence always corresponds to the space a flux or quantity lives in. More details on the transformation, especially of other derivatives, can be found in [51]. A subset of only the necessary formulas for the Discontinuous Galerkin method in chapter 4 is introduced briefly in [39]. In the following the calligraphic $\mathcal{F}$ denotes the flux in reference space and the typographic F the flux in physical space.

Using the same steps as for the derivation of the weak formulation in physical space, the weak formulation of the mapped Navier–Stokes equations on the reference element E is then given by

$$\int_E J\mathbf{u}_t \Phi \, \mathrm{d}\xi - \int_E \left(\mathcal{F}^c - \mathcal{F}^v\right) \cdot \nabla\Phi \, \mathrm{d}\xi + \int_{\partial E} \left(\left(\mathcal{F}^c - \mathcal{F}^v\right) \cdot \mathcal{N}\right) \Phi \, \mathrm{d}S_\xi = 0, \quad (2.7)$$

where $\mathcal{N}$ denotes the normal vector in the reference space.

3. Shape derivative for the compressible Navier–Stokes equations

In the field of aerodynamics, the simulation of flows around an object in general, e.g. a wing or even a full aircraft, is well established. The capability of predicting the aerodynamic performance, like the drag or lift coefficient, for a single design very accurately leads to the wish of not only investigating this design, but to optimize it with respect to specific cost functions. Derivative free, heuristic optimization methods are used in practice, but due to the fact that they do not exploit the structure of a problem heavily, the application to industry size large scale problems is very challenging and time consuming. Hence, the key factor that drives the numerical costs, i.e. the number of design parameters, must be kept low. Therefore, it is very common to limit the modifications of a shape to a finite set of design parameters, like B-splines or the popular Hicks-Henne functions. This restriction to a reduced set of parameters can conflict with the aim of achieving not only a good, but the best design. Alternative concepts, where the numerical effort does not depend on the number of design parameters, are required. The introduction of adjoint calculus is one of the major advances in the field of aerodynamic design optimization [65, 31, 29]. In this context the evaluation of the governing equation by a numerical flow simulation is called the forward problem, which depends on input quantities that are parameters of an optimization. The numerical effort required for this optimization directly depends on the number of these parameters. The basic idea of the adjoint approach is to solve an adjoint problem, associated to the forward problem, which allows to remove the dependence of the optimization on the number of design parameters.

A further drawback, which also affects the general adjoint approach, is the use of proprietary software in the design chain. This is often the case when computer aided design (CAD) is involved in the mesh generation process. It then becomes very challenging to apply the adjoint approach to these proprietary parts of overall optimization and one is forced to use finite differences once again.

One approach to overcome the problem of expensive computations for a large set of design parameters or non-adjoinable mesh generators is the use of shape calculus. Shape calculus is the mathematical framework to build derivatives with respect to the shape in a continuous setting, where the equations are not discretized yet. A detailed introduction into the general theoretical framework can be found in [22, 83] and is one of the bases of this work. Due to the fact that shape calculus is performed on the continuous equations, it is always independent of the number of design parameters. All surface nodes can be chosen as unknowns leading to the so called "free node parameterization".

This approach has already been applied to very large scale aerodynamic design optimization [13, 73, 63, 96], but since they are all based on the strong, pointwise form of the governing equations they all assume the existence of a strong form solution. However, the existence of such solutions in the presence of shock waves or flow discontinuities is not clear and one is restricted to the variational form. This is not a big issue when the numerical scheme is also based on the variational form of the governing equations and therefore, anyway only computes a solution to this form. In contrast to the strong form, only a few work on shape derivatives based on the variational forms are found in literature, like [43, 82] and especially [42], where the incompressible Navier–Stokes equations are considered. In this chapter, which summarizes [85], the focus lies on the shape derivatives for the compressible Navier–Stokes equations and mainly on the differences that arise due to the use of the variational form instead of the strong form as origin of the whole derivation.

This chapter is structured as follows. After describing the aerodynamic setting, the required concepts of shape calculus are introduced. They are then applied to the strong and the variational form of the Navier–Stokes equations. With adjoint calculus, the shape derivative of the aerodynamic cost functions are built, which in the last part are investigated numerically. A comparison to finite differences shows the advantages of the variational approach over the strong approach.

3.1. Aerodynamic objective functions and boundary conditions

In the field of aircraft design the lift and drag coefficient are the main quantities characterizing the performance of an airfoil. For an optimization, derivatives of those coefficients with respect to the shape are of interest since they allow a reduction of fuel consumption, for example. Having a wing or aircraft flying through air in mind, the general flow setup is given as follows. As before, Ω denotes the domain of the fluid with Γ the boundaries of the domain. These boundaries consist of the farfield Γ_∞ and the wall boundary Γ_W, which describe the outline of the airfoil, wing or aircraft. The wall can be split up in the parts $\Gamma_{iso} \cup \Gamma_{adia} = \Gamma_W$, where either the isothermal boundary condition $T = T_W$ or the adiabatic boundary condition $\nabla T \cdot n = 0$ holds. Furthermore, the no-slip boundary condition of a zero velocity $v = 0$ is imposed along the whole wall Γ_W.

Definition 3.1 (Cost Function)**.** *Integrating pressure p and viscous forces τ along the geometry gives the drag and lift coefficients of a body immersed in the flow. These are considered as cost (or target) functions, given by*

$$\mathcal{J}(\mathbf{u}) := \frac{1}{C_\infty} \int_{\Gamma_W} (pn - \tau n) \cdot \theta \, ds, \tag{3.1}$$

where C_∞ is a constant and θ is either $\theta_l = (-\sin(\alpha), \cos(\alpha))^\top$ for the lift or $\theta_d = (\cos(\alpha), \sin(\alpha))^\top$ for the drag coefficient and α is the angle of attack.

3.2. Shape calculus

3.2.1. Definition of the shape derivative and the Hadamard theorem

In this section, the basic concepts of shape derivatives are introduced, especially the fundamental Hadamard theorem, as stated in [22, 83].

Definition 3.2 (Perturbation of identity)**.** *Let D be an open set in $\mathbb{R}^d$ and let $\Omega \subset D$ be a measurable subset of D. For vector fields $V \in C_0^k(D;\mathbb{R}^d)$,*

$$T_t[V] : D \times [0, \delta) \to \mathbb{R}^d, \quad (x, t) \mapsto x + tV(x)$$

is called the **perturbation of identity***.*

In this definition, the open set D is the so-called hold-all set which contains the domain Ω. The perturbation of identity is a very common approach to describe deformations $\Omega_t = T_t[V](\Omega)$ of the original domain Ω and can directly be used to define the shape derivative of a domain functional in the direction of a vector field.

Definition 3.3 (Shape derivative, shape differentiable, shape gradient)**.** *Let $V \in C_0^k(D;\mathbb{R}^d)$ and $\mathcal{J}(\Omega)$ be a domain functional at Ω. The Eulerian derivative*

$$d\mathcal{J}(\Omega;V) := \lim_{t \searrow 0} \frac{\mathcal{J}(\Omega_t) - \mathcal{J}(\Omega)}{t}$$

is called the **shape derivative** *of $\mathcal{J}(\Omega)$ in the direction V. If this shape derivative $d\mathcal{J}(\Omega;V)$ exists for all directions V, and the mapping*

$$G(\Omega) : C_0^k(D;\mathbb{R}^d) \to \mathbb{R}, \; V \mapsto d\mathcal{J}(\Omega;V)$$

is linear and continuous, the functional $\mathcal{J}$ is called **shape differentiable***. The mapping $G(\Omega) \in \left(C_0^k(D;\mathbb{R}^d)\right)^*$ is the* **shape gradient***, which fulfills*

$$d\mathcal{J}(\Omega;V) = \langle G(\Omega), V \rangle_{\left(C_0^k(D;\mathbb{R}^d)\right)^* \times C_0^k(D;\mathbb{R}^d)} \quad \forall \, V \in C_0^k(D;\mathbb{R}^d). \tag{3.2}$$

Therewith, the shape derivative can be computed by the dual pair, which is a generalized scalar product of the shape gradient and the direction V. If this vector field fulfills $V \cdot n = 0$ at the boundary of the domain, meaning it is tangential to this boundary $\Gamma = \partial\Omega \in C^k$, the shape derivative in this direction becomes zero. A proof of this can be found in [83], but it is intuitively clear. Since deformations in the tangential directions do not change the shape of the domain, there will be no change in the flow solution and, therewith, no change in the cost functional. One can think of such deformations being a reparameterization of the geometry.

Additionally it becomes obvious that the shape derivative only depends on the normal component of the vector field at the boundary of the domain. There exists a continuous linear mapping $d\mathcal{J}(\Gamma;\cdot) : C^k(\Gamma) \to \mathbb{R}$ such that for all vector fields $V \in C^k(\overline{D};\mathbb{R}^d)$ the relation

$$d\mathcal{J}(\Omega;V) = d\mathcal{J}(\Gamma;V \cdot n)$$

holds. In the following theorem, the idea of this relation is expanded to the definition of the shape gradient. If the shape derivative only depends on the normal component of the vector field at the boundary, there must be a scalar distribution $g(\Gamma)$ on the boundary which takes the role of the shape gradient $G(\Omega)$ in equation (3.2).

Theorem 3.4 (Hadamard Theorem, Hadamard formula)**.** *For every domain $\Omega \subset D$ of class C^k, let $\mathcal{J}(\Omega)$ be a shape differentiable function. Furthermore, let the boundary Γ be of class C^{k-1}. There exists the following scalar distribution $g(\Gamma) \in C_0^k(\Gamma)^*$, such that the shape gradient $G(\Omega) \in C_0^k(\Omega, \mathbb{R}^d)^*$ of $\mathcal{J}(\Omega)$ is given by*

$$G(\Omega) = \gamma_\Gamma^*(g \cdot n),$$

where $\gamma_\Gamma \in L\left(C_0^k(\Omega, \mathbb{R}^d), C_0^k(\Gamma, \mathbb{R}^d)\right)$ and γ_Γ^* denote the trace operator and its adjoint operator. In this situation, one can show that [83]

$$d\mathcal{J}(\Omega;V) = d\mathcal{J}(\Gamma;V \cdot n) = \langle g, V \cdot n\rangle_{\left(C_0^k(\Gamma)\right)^* \times C_0^k(\Gamma)}.$$

If $g(\Gamma)$ is integrable over Γ, the **Hadamard Formula**

$$d\mathcal{J}(\Omega;V) = \int_\Gamma (V \cdot n) g \, ds$$

is fulfilled. Terms being of the structure "$(V \cdot n) \ldots$" are called to be in **Hadamard form***.*

Proof: Further details and a proof can be found in [22] or in [83]. □

The goal is now to find this scalar distribution $g(\Gamma)$ for a cost function, like drag or lift coefficient, in the context of the Navier–Stokes equations. In a numerical context, where the shape is represented by a finite amount of points, one would then be able to evaluate the shape derivative in each point by simply evaluating a boundary integral with a vector field that is associated to a deformation of the shape in this point. For the derivation of such a distribution further general definitions in the field of shape calculus are required.

Definition 3.5 (Material derivative / Local shape derivative)**.** *The total derivative*

$$d_V[f](x) := \left.\frac{d}{dt}\right|_{t=0} f(t, T_t[V](x))$$

of f is called the **material derivative***. Furthermore, the partial derivative*

$$f'(x) := f'[V](x) := \frac{\partial}{\partial t} f(t, x)$$

is called the **local shape derivative** *of f.*

For a vector field V being orthogonal to the boundary Γ, the material derivative of the normal vector fulfills

$$d_V[n] = -\nabla_\Gamma (V \cdot n),$$

which can be found in [72].

Remark 3.6. *This material derivative and the local shape derivative are linked to each other by the chain rule, if both exist*

$$d_V[f](x) = f'[V](x) + \nabla f(0, x) \cdot V(x) = f' + \nabla f \cdot V,$$

where $\left.\frac{d}{dt}\right|_{t=0} T_t[V](x) = \left.\frac{d}{dt}\right|_{t=0} (x + tV(x)) = V(x)$ is used for the perturbation of identity. Since this relation only holds if the local shape derivative exists, one usually accepts the above formula as a definition of the local shape derivative instead. This is for example the case at sharp convex corners of the domain, like trailing edges or other feature edges.

3.2.2. Tangential calculus

Before coming to the shape derivative of volume and boundary integrals, a brief summary of tangential calculus, which will later be used to derive a preliminary shape derivative of the lift and drag coefficients, is needed. For further discussions on tangential calculus the reader is referred to [22], Chapter 8, Section 5.

Definition 3.7 (Tangential gradient, tangential divergence)**.** *Let $f \in C^1(\Gamma)$ be a function with a C^1-extension F into a tubular neighborhood of the boundary Γ and let n be the normal vector of this boundary. The* **tangential gradient** *is given by*

$$\nabla_\Gamma f := \nabla F|_\Gamma - \frac{\partial F}{\partial n} n,$$

which is the ordinary gradient minus the normal component.

Analogously, for a smooth vector field $W \in (C^1(\Gamma))^d \cap (C^1(\Omega))^d$, the **tangential divergence** *is defined by*

$$\operatorname{div}_\Gamma W := \operatorname{div} W - DWn \cdot n.$$

Theorem 3.8 (Tangential Green's formula)**.** *Let f and W be as defined above. Then, the* **tangential Green's formula** *is given by*

$$\int_\Gamma W \cdot \nabla_\Gamma f \, ds = \int_\Gamma f K(W \cdot n) - f \operatorname{div}_\Gamma W \, ds,$$

where $K := \operatorname{div}_\Gamma n$ denotes the sum of the principal curvatures, the so called additive curvature, or $(d-1)$ times the mean curvature.

Proof: A proof can be found in [22], Chapter 8. □

3.2.3. Shape derivative for volume and boundary integrals

The cost functions of interest are integrals over the boundary of the domain. For numerical investigations these cost functions are evaluated with values from a flow solution, which is based on the variational form of the governing equations, here the Navier–Stokes equations. This variational form is a volume integral. Therefore, not only the shape derivative for a general boundary integral, but also for volume integrals is needed, as they can be found in [22] for example and will be recapitulated here. Based on this, a preliminary shape derivative of the drag and lift coefficients, not yet in Hadamard form, can be stated.

For a general volume integral $\mathcal{J}(\Omega) = \int_\Omega f(x) \, \mathrm{d}x$ the shape derivative is given by

$$d\mathcal{J}(\Omega;V) = \int_\Omega f' \, \mathrm{d}x + \int_\Gamma (V \cdot n) f \, \mathrm{d}s,$$

while for a general boundary cost function $\mathcal{J}(\Omega) = \int_\Gamma f \, \mathrm{d}s$ the shape derivative fulfills

$$d\mathcal{J}(\Omega;V) = \int_\Gamma f' + (V \cdot n)\left(\frac{\partial f}{\partial n} + Kf\right) \mathrm{d}s.$$

Theorem 3.9 (Preliminary shape derivative of the cost functional)**.** *Let the vector field of the perturbation of identity be zero in the neighborhood of the farfield boundary Γ_∞. Then, the shape derivative of the lift and drag coefficients from equation* (3.1) *fulfills*

$$d\mathcal{J}(\Omega;V) = \frac{1}{C_\infty} \int_{\Gamma_W} (p'n - \tau' n) \cdot \theta + (V \cdot n) \operatorname{div}(p\theta - \tau\theta)\, ds. \tag{3.3}$$

Proof: A proof is given in [85], which follows the argumentation in [72]. □

This theorem already states a possible form of the shape derivative of the target functional. Nevertheless, it is still preliminary since it is not in Hadamard form, due to the local shape derivatives p' and τ' it still contains. A computation of these, with finite differences, would require one forward flow solution for each design parameter of the parameterization of the shape, which is prohibitively costly. To turn this preliminary formula into Hadamard form, one needs adjoint calculus to remove these local shape derivatives of the pressure and the viscous stress tensor. But before this, the shape derivative of the forward problem, the Navier–Stokes equations, is required.

3.3. Shape derivative in strong and variational form

The application of adjoint calculus to the preliminary shape derivative of the drag and lift coefficients requires a corresponding forward problem. The compressible Navier–Stokes equations are therefore linearized with respect to a variation of the flow domain Ω. As already mentioned above, the distinction between the Navier–Stokes equations in pointwise and in variational form is of major importance, which will in the end lead to different Hadamard forms for the shape derivative of the cost function. The proofs of the following theorems for the shape derivative of the Navier–Stokes equations in both forms can be found in [85].

Theorem 3.10 (Shape derivative of the pointwise Navier–Stokes equations)**.** *The local shape derivative $\mathbf{u}'$ of the solution of the stationary Navier–Stokes equations in strong form* (2.2) *is given as the solution of the equation*

$$0 = \nabla \cdot \left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\nabla\mathbf{u}' \right) \text{ in } \Omega, \tag{3.4}$$

where $\mathcal{F}^c_{\mathbf{u}} := \frac{\partial \mathcal{F}^c}{\partial \mathbf{u}}$, $\mathcal{F}^v_{\mathbf{u}} := \frac{\partial \mathcal{F}^v}{\partial \mathbf{u}}$ and $\mathcal{F}^v_{\nabla\mathbf{u}} := \frac{\partial \mathcal{F}^v}{\partial \nabla\mathbf{u}}$ are the derivatives of the convective and viscous fluxes with respect to the state and the gradient of the state.

Theorem 3.11 (Shape derivative of the variational Navier–Stokes equations)**.** *The shape derivative of the variational form of the stationary Navier–Stokes equations* (2.3) *is given by the problem: Find* $\mathbf{u}' \in \mathcal{H}$*, such that*

$$
\begin{aligned}
0 = &- \left(\mathbf{u}', \left[\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}) - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right]^\top \nabla\Phi\right)_\Omega - \left(\langle V, n\rangle \left[\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right], \nabla\Phi\right)_{\Gamma_W} \\
&- \left(\mathbf{u}', \nabla\cdot\left[\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\Phi\right]\right)_\Omega + \left(\mathbf{u}', n\cdot\left[\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\Phi\right]\right)_\Gamma \\
&+ \left(\mathbf{u}', \left[n\cdot\left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}) - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)\right]^\top \Phi\right)_{\Gamma\setminus\Gamma_W} \\
&- \left(\nabla\mathbf{u}', \left(n\cdot\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \Phi\right)_{\Gamma\setminus\Gamma_W} + \left(n\cdot\left(\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right)', \Phi\right)_{\Gamma_W} \\
&+ \int_{\Gamma_W} \langle V, n\rangle \nabla\cdot\left(\left[\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right]\cdot\Phi\right) ds \qquad \forall\Phi\in\mathcal{H}.
\end{aligned}
\tag{3.5}
$$

Comparing equation (3.4) and equation (3.5), the main difference is that the shape derivative of the variational Navier–Stokes equations already contains the multiplication with the test function and the integration over the domain, since these two operations are performed before the shape differentiation. For the pointwise form these two steps will be performed in section 3.4.2 not before, but after taking the shape derivative. The differences in the final Hadamard form for the shape derivative of the cost functions will be based on this varied order of shape differentiation and building of the weak form.

3.4. Adjoint calculus

Using adjoint calculus is a common approach to reformulate shape optimization problems as documented in [31], [47] or [44]. The basic concept will be stated in the following for a cost function $\mathcal{J} = \mathcal{J}(\mathbf{u}, S)$, which depends on a function S describing the shape and the flow solution $\mathbf{u}$ of the governing equation. For a general governing equation

$$Q(\mathbf{u}, S) = 0,$$

the solution $\mathbf{u}$ of this equation depends also on the shape function S. Therewith, a variation of the shape δS leads to a variation of the cost function given by

$$\delta\mathcal{J} = \frac{\partial\mathcal{J}}{\partial\mathbf{u}}\delta\mathbf{u} + \frac{\partial\mathcal{J}}{\partial S}\delta S. \tag{3.6}$$

It is clear to see that a numerical computation of this variation $\delta\mathcal{J}$ requires the sensitivity of the flow solution $\delta\mathbf{u}$ with respect to each degree of freedom within the shape deformation. However, the calculation of this variation would require the computation of a complete flow solution for each parameter defining the shape, which results in a prohibitive large numerical effort. To avoid this, one has to eliminate the variation $\delta\mathbf{u}$ from equation (3.6) by using the variation of the governing equation

$$\delta Q = \frac{\partial Q}{\partial \mathbf{u}}\delta\mathbf{u} + \frac{\partial Q}{\partial S}\delta S = 0,$$

which provides another equation determining the variation $\delta\mathbf{u}$. Multiplying this equation by a Lagrange multiplier $\mathbf{z}$ and subtracting it from the variation of the cost function leads to

$$\delta\mathcal{J} = \delta\mathcal{J} - \mathbf{z}^\top \delta Q.$$

Putting the last three equations together yields

$$\begin{aligned}\delta\mathcal{J} &= \frac{\partial \mathcal{J}}{\partial \mathbf{u}}\delta\mathbf{u} + \frac{\partial \mathcal{J}}{\partial S}\delta S - \mathbf{z}^\top\left(\frac{\partial Q}{\partial \mathbf{u}}\delta\mathbf{u} + \frac{\partial Q}{\partial S}\delta S\right)\\ &= \left(\frac{\partial \mathcal{J}}{\partial \mathbf{u}} - \mathbf{z}^\top\frac{\partial Q}{\partial \mathbf{u}}\right)\delta\mathbf{u} + \left(\frac{\partial \mathcal{J}}{\partial S} - \mathbf{z}^\top\frac{\partial Q}{\partial S}\right)\delta S,\end{aligned}$$

where the first term, containing $\delta\mathbf{u}$, can be eliminated, if $\mathbf{z}$ is the solution of the adjoint equation

$$\frac{\partial \mathcal{J}}{\partial \mathbf{u}} - \mathbf{z}^\top\frac{\partial Q}{\partial \mathbf{u}} = 0.$$

Altogether, the variation of the cost function becomes

$$\delta\mathcal{J} = \left(\frac{\partial \mathcal{J}}{\partial S} - \mathbf{z}^\top\frac{\partial Q}{\partial S}\right)\delta S, \tag{3.7}$$

which only requires a single solve of the adjoint equation instead of multiple primal solves.

In the following, this approach is used to remove the remaining local shape derivatives p' and τ' from the preliminary shape derivative of the drag and lift coefficients (3.3) and thereby transform it into Hadamard form. Therefore, several intermediate steps are required. Looking at the right hand side of equation (3.7), one needs, besides the already given preliminary shape derivative, the solution of the adjoint equation $\mathbf{z}$ and the shape derivative of the Navier–Stokes equations $\frac{\partial Q}{\partial S}\delta S$. In figure 3.1 the essential steps of the following way to the Hadamard form are plotted. After inserting the adjoint Navier–Stokes equations into the shape derivative of the Navier–Stokes equations one ends up with a combined equation (3.12) for both approaches, pointwise and variational, where one term is specially marked and must be omitted for the pointwise approach. This equation is further transformed into an equation (3.17), including also local shape derivatives

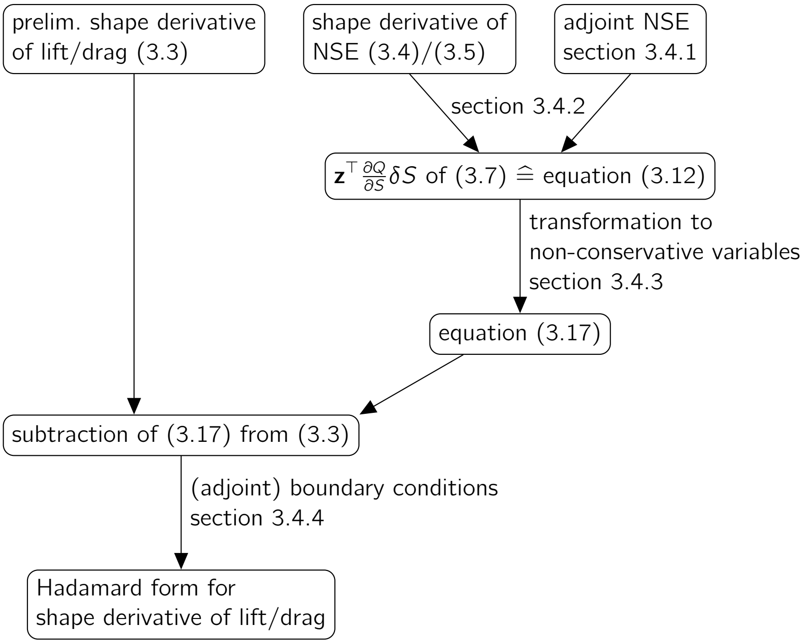

prelim. shape derivative of lift/drag (3.3)
shape derivative of NSE (3.4)/(3.5)
adjoint NSE section 3.4.1
section 3.4.2
$\mathbf{z}^\top \frac{\partial Q}{\partial S}\delta S$ of (3.7) $\widehat{=}$ equation (3.12)
transformation to non-conservative variables section 3.4.3
equation (3.17)
subtraction of (3.17) from (3.3)
(adjoint) boundary conditions section 3.4.4
Hadamard form for shape derivative of lift/drag

Figure 3.1.: Flow chart of the basic steps to bring the preliminary shape derivative of the lift and drag coefficients, using adjoint calculus, into Hadamard form.

of non-conservative variables. As explained above for the general adjoint approach, this equation is subtracted from the preliminary shape derivatives of the target function. In the last step the boundary conditions and additionally the adjoint boundary conditions are inserted to finally end up with the Hadamard form.

3.4.1. Variational formulation of the continuous adjoint problem

The above general adjoint methodology requires the solution of the adjoint equation. In a numerical framework, where the solver of the forward problem is built on the variational formulation, for example Finite Volume or Discontinuous Galerkin methods, one also needs the variational formulation of the continuous adjoint problem. A lot of details on this variational approach can be found in [42], and the respective integral transformations are covered in more depth in [35].

Theorem 3.12 (Variational form of the adjoint Navier–Stokes equations)**.** *The variational formulation of the adjoint Navier–Stokes equations is given by finding* $\mathbf{z} \in \mathcal{H}$ *such that*

$$\begin{aligned} &-\left(\mathbf{w}, (\mathcal{F}^c_{\mathbf{u}} - \mathcal{F}^v_{\mathbf{u}})^\top \nabla \mathbf{z}\right)_\Omega - \left(\mathbf{w}, \nabla \cdot \left(\left(\mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \nabla \mathbf{z}\right)\right)_\Omega + \left(\mathbf{w}, n \cdot \left(\left(\mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \nabla \mathbf{z}\right)\right)_\Gamma \\ &+ \left(\mathbf{w}, \left(n \cdot (\mathcal{F}^c_{\mathbf{u}} - \mathcal{F}^v_{\mathbf{u}})\right)^\top \mathbf{z}\right)_\Gamma - \left(\nabla \mathbf{w}, \left(n \cdot \mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \mathbf{z}\right)_\Gamma = \mathcal{J}'[\mathbf{u}](\mathbf{w}) \quad \forall \mathbf{w} \in \mathcal{H}, \end{aligned} \tag{3.8}$$

where the linearization of the cost function in case of drag or lift coefficient is given by

$$\mathcal{J}'[\mathbf{u}](\mathbf{w}) = \left(\tfrac{1}{C_\infty}\left(p_{\mathbf{u}} n - \tau_{\mathbf{u}} n\right) \cdot \theta, \mathbf{w}\right)_{\Gamma_W} - \left(\tfrac{1}{C_\infty}\left(\tau_{\nabla\mathbf{u}} n\right) \cdot \theta, \nabla \mathbf{w}\right)_{\Gamma_W}.$$

Proof: A proof following the outline of [42] is given in [85] □

Corollary 3.13. *Choosing* $\mathbf{w}$ *in equation* (3.8) *with appropriate compact support in either* Ω *or on* Γ_W *and* $\Gamma \setminus \Gamma_W$*, one can see that*

$$-\left(\mathbf{w}, (\mathcal{F}^c_{\mathbf{u}} - \mathcal{F}^v_{\mathbf{u}})^\top \nabla \mathbf{z}\right)_\Omega - \left(\mathbf{w}, \nabla \cdot \left(\left(\mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \nabla \mathbf{z}\right)\right)_\Omega = 0 \quad \forall \mathbf{w} \in \mathcal{H}_0(\Omega) \tag{3.9}$$

for the volume. For a test function with compact support on Γ_W *it holds*

$$\begin{aligned} &\left(\mathbf{w}, n \cdot \left(\left(\mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \nabla \mathbf{z}\right)\right)_{\Gamma_W} + \left(\mathbf{w}, \left(n \cdot (\mathcal{F}^c_{\mathbf{u}} - \mathcal{F}^v_{\mathbf{u}})\right)^\top \mathbf{z}\right)_{\Gamma_W} - \left(\nabla \mathbf{w}, \left(n \cdot \mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \mathbf{z}\right)_{\Gamma_W} \\ &\quad = \left(\tfrac{1}{C_\infty}\left(p_{\mathbf{u}} n - \tau_{\mathbf{u}} n\right) \cdot \theta, \mathbf{w}\right)_{\Gamma_W} - \left(\tfrac{1}{C_\infty}\left(\tau_{\nabla\mathbf{u}} n\right) \cdot \theta, \nabla \mathbf{w}\right)_{\Gamma_W} \\ &\qquad \forall \mathbf{w} \in \mathcal{H} \cap \mathcal{H}_0(\Gamma_W), \end{aligned}$$

and finally using the same argumentation on all remaining boundaries $\Gamma \setminus \Gamma_W$

$$\begin{aligned} &\left(\mathbf{w}, n \cdot \left(\left(\mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \nabla \mathbf{z}\right)\right)_{\Gamma\setminus\Gamma_W} + \left(\mathbf{w}, \left(n \cdot (\mathcal{F}^c_{\mathbf{u}} - \mathcal{F}^v_{\mathbf{u}})\right)^\top \mathbf{z}\right)_{\Gamma\setminus\Gamma_W} \\ &\qquad - \left(\nabla \mathbf{w}, \left(n \cdot \mathcal{F}^v_{\nabla\mathbf{u}}\right)^\top \mathbf{z}\right)_{\Gamma\setminus\Gamma_W} = 0 \\ &\qquad\qquad \forall \mathbf{w} \in \mathcal{H} \cap \mathcal{H}_0(\Gamma \setminus \Gamma_W). \end{aligned} \tag{3.10}$$

3.4.2. Application of adjoint equation to the shape derivative of the Navier–Stokes equations

The adjoint equations from the last subsection are now inserted into the shape derivative of the Navier–Stokes equations to derive two intermediate relationships between the adjoint equation and the pointwise linearization of the Navier–Stokes equations on the one hand and the linearization of the weak form on the other hand.

For the pointwise problem, the shape derivative of the Navier–Stokes equations from theorem 3.10 is multiplied with a test function Φ and integrated over the domain Ω

$$0 = \left(\nabla \cdot \left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\nabla\mathbf{u}'\right), \Phi\right)_\Omega \quad \forall\Phi \in \mathcal{H}.$$

Integration by parts results in

$$\begin{aligned} 0 = &- \left(\left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\nabla\mathbf{u}'\right), \nabla\Phi\right)_\Omega \\ &+ \left(n \cdot \left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\nabla\mathbf{u}'\right), \Phi\right)_{\Gamma\setminus\Gamma_W} \\ &+ \left(n \cdot \left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\nabla\mathbf{u}'\right), \Phi\right)_{\Gamma_W} \quad \forall\Phi \in \mathcal{H}. \end{aligned}$$

To eliminate the local shape derivatives of the gradient of the solution $\nabla\mathbf{u}'$ in the volume integral, one has to use a further integration by parts. Before this can be applied n, $\mathcal{F}^c_{\mathbf{u}}(\mathbf{u})$, $\mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})$ and $\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})$ are shifted to the other side of the products. Doing this also for the farfield boundary yields

$$\begin{aligned} 0 = &- \Big(\mathbf{u}', \left[\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}) - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right]^\top \nabla\Phi\Big)_\Omega + \Big(\nabla\mathbf{u}', \left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\Phi\Big)_\Omega^{\boxed{1}} \\ &+ \Big(\mathbf{u}', \left[n \cdot \left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}) - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)\right]^\top \Phi\Big)_{\Gamma\setminus\Gamma_W} \\ &- \Big(\nabla\mathbf{u}', \left[n \cdot \left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)\right]^\top \Phi\Big)_{\Gamma\setminus\Gamma_W} \\ &+ \left(n \cdot \left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\mathbf{u}' - \mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\nabla\mathbf{u}'\right), \Phi\right)_{\Gamma_W}^{\boxed{2}} \quad \forall\Phi \in \mathcal{H}. \end{aligned}$$

To simplify the following discussions, framed numbers $\boxed{1}$, $\boxed{2}$, $\boxed{3}$, ... are used to refer to certain terms. Integration by parts in the second volume integral $\boxed{1}$ and applying the chain rule backwards to the wall integral $\boxed{2}$ leads to

$$\begin{aligned}
0 = &-\left(\mathbf{u}', \left[\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}) - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right]^\top \nabla\Phi\right)_\Omega \\
&-\left(\mathbf{u}', \nabla\cdot\left[\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\Phi\right]\right)_\Omega^{\boxed{1}} + \left(\mathbf{u}', n\cdot\left[\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\Phi\right]\right)_\Gamma^{\boxed{1}} \\
&+\left(\mathbf{u}', \left[n\cdot\left(\mathcal{F}^c_{\mathbf{u}}(\mathbf{u}) - \mathcal{F}^v_{\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)\right]^\top \Phi\right)_{\Gamma\setminus\Gamma_W} \\
&-\left(\nabla\mathbf{u}', \left[n\cdot\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)\right]^\top \Phi\right)_{\Gamma\setminus\Gamma_W} \\
&+\left(n\cdot\left(\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right)', \Phi\right)_{\Gamma_W}^{\boxed{2}} \qquad \forall\Phi\in\mathcal{H}.
\end{aligned}$$

After reformulating the shape derivative of the pointwise Navier–Stokes equations, the final step is to use the adjoint conditions (3.9) and (3.10) to obtain

$$0 = \left(\mathbf{u}', n\cdot\left[\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\mathbf{z}\right]\right)_{\Gamma_W} + \left(n\cdot\left(\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right)', \mathbf{z}\right)_{\Gamma_W} \qquad \forall\mathbf{z}\in\mathcal{H}, \tag{3.11}$$

where the name of the dependent variable was changed from Φ to $\mathbf{z}$. At first glance, it might seem counter-intuitive to use the weak form adjoint equations (3.9) and (3.10) for the pointwise approach, but a pointwise interpretation of those does not effect the above equation.

Now the same process is applied to the variational Navier–Stokes equations from theorem 3.11. Using the adjoint equations (3.9) and (3.10) results in

$$\begin{aligned}
0 = &\left(\mathbf{u}', n\cdot\left[\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\mathbf{z}\right]\right)_{\Gamma_W} + \left(n\cdot\left(\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right)', \mathbf{z}\right)_{\Gamma_W} \\
&+\int_{\Gamma_W} \langle V, n\rangle \nabla\cdot\left(\left[\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right]\cdot\mathbf{z}\right) \mathrm{d}s \\
&-\left(\langle V, n\rangle\left[\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right], \nabla\mathbf{z}\right)_{\Gamma_W} \qquad \forall\mathbf{z}\in\mathcal{H}.
\end{aligned}$$

Applying the product rule to the divergence yields

$$\begin{aligned}
0 = &\left(\mathbf{u}', n\cdot\left[\left(\mathcal{F}^v_{\nabla\mathbf{u}}(\mathbf{u}, \nabla\mathbf{u})\right)^\top \nabla\mathbf{z}\right]\right)_{\Gamma_W} + \left(n\cdot\left(\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right)', \mathbf{z}\right)_{\Gamma_W} \\
&+\boxed{\left(\langle V, n\rangle\nabla\cdot\left[\mathcal{F}^c(\mathbf{u}) - \mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})\right], \mathbf{z}\right)_{\Gamma_W}} \qquad \forall\mathbf{z}\in\mathcal{H}.
\end{aligned} \tag{3.12}$$

Equation (3.11) differs from equation (3.12) only by the framed extra term. This term vanishes, if the Navier–Stokes equations are fulfilled pointwise anyway. In the following, only equation (3.12) will be used to avoid a fork in the derivation for the pointwise and the variational approach, but one has to keep in mind that all framed terms only occur in the variational approach.

3.4.3. Transformation to non-conservative variables

In this subsection, especially the no-slip condition at the wall $v = 0$ is used to reformulate the local shape derivative $\mathbf{u}'$ in equation (3.12) such that the local shape derivatives of the pressure p' and the viscous stress tensor τ' appear. These terms will later be used to eliminate their counterparts in the preliminary shape derivative of the cost function.

Starting with the first integral of equation (3.12), the local shape derivative $\mathbf{u}'$ reduces with the no-slip condition to

$$\mathbf{u}' = \begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho e \end{pmatrix}' = \begin{pmatrix} \rho' \\ \rho v_1' \\ \rho v_2' \\ (\rho e)' \end{pmatrix} \quad \text{on } \Gamma_W, \tag{3.13}$$

where, due to the vanishing velocity v, the relation

$$(yv)' = y'v + yv' = yv' \quad \text{on } \Gamma_W \tag{3.14}$$

for a general quantity y was used. This relation will also be used for all other terms containing the velocity v. To incorporate the no-slip condition also on the right hand side of the first integral, the so called homogeneity tensor is stated as

$$G = \left[G_{kl}^{ij}\right]_{kl}^{ij} = \frac{\partial \left(f_k^v\right)_i}{\partial \frac{\partial \mathbf{u}_j}{\partial x_l}},$$

where $(f_k^v)_i$ denotes the i-th component of the k-th viscous flux vector and $\frac{\partial \mathbf{u}_j}{\partial x_l}$ denotes the derivative of the j-th component of the vector of conservative variables with respect to x_l. In two space dimensions the i and j indices range from 1 to 4 and $k, l \in \{1, 2\}$. For a detailed discourse of the general homogeneity tensor see [35]. Using the no-slip condition the homogeneity tensor at the wall is given by

$$G_{11} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{4}{3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{\gamma}{Pr}e & 0 & 0 & \frac{\gamma}{Pr} \end{pmatrix}, \quad G_{12} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{2}{3} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$G_{21} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\frac{2}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad G_{22} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4}{3} & 0 \\ -\frac{\gamma}{Pr}e & 0 & 0 & \frac{\gamma}{Pr} \end{pmatrix}.$$

It is clear that the first rows of this tensor are all zero, since the viscous flux has no influence on the mass equation. Therefore, there is no part in the first integral of equation (3.12) containing the gradient of the first component of the adjoint variable $\nabla \mathbf{z}_1$. For the momentum equations, one introduces the so called **adjoint stress tensor** as

$$\Sigma := \mu \left(\nabla \mathbf{z}_{2,3} + (\nabla \mathbf{z}_{2,3})^\top - \frac{2}{3} (\nabla \cdot \mathbf{z}_{2,3})\, I \right).$$

With this and the local shape derivative of the state $\mathbf{u}'$ from equation (3.13), the second and third row of the first integral in equation (3.12) finally reduce to

$$\left(\mathbf{u}'_{2,3}, n \cdot \left((\mathcal{F}^v_{\nabla \mathbf{u}})^\top_{2,3} \nabla \mathbf{z}_{2,3} \right) \right)_{\Gamma_W} = \int_{\Gamma_W} v' \cdot (n \cdot \Sigma) \, \mathrm{d}s.$$

It remains the fourth component of $\nabla \mathbf{z}$, which correspond to the fourth rows of G, having non-zero entries only for $k = l$ and $j = 1$ or 4. Since the j-th column of the matrices G_{kl} are multiplied with the j-th component of the vector $\mathbf{u}$ the remaining expression becomes

$$\left(\mathbf{u}'_4, n \cdot \left((\mathcal{F}^v_{\nabla \mathbf{u}})^\top_4 \nabla \mathbf{z}_4 \right) \right)_{\Gamma_W} = \int_{\Gamma_W} \frac{\mu}{\rho} \frac{\gamma}{Pr} \left(-e\mathbf{u}'_1 + \mathbf{u}'_4 \right) n \cdot \nabla \mathbf{z}_4 \, \mathrm{d}s.$$

Using $\mathbf{u}'_1 = \rho'$ and $\mathbf{u}'_4 = (\rho e)' = \rho' e + \rho e'$ as well as

$$T'\kappa = \frac{\mu\gamma}{Pr} \left(e' - \left(\frac{1}{2} v^2 \right)' \right) = \frac{\mu\gamma}{Pr} e'$$

for $T\kappa = \frac{\mu\gamma}{Pr} \left(e - \frac{1}{2} v^2 \right)$ at the no-slip wall the above integral is simplified to

$$\left(\mathbf{u}'_4, n \cdot \left((\mathcal{F}^v_{\nabla \mathbf{u}})^\top_4 \nabla \mathbf{z}_4 \right) \right)_{\Gamma_W} = \int_{\Gamma_W} T'\kappa n \cdot \nabla \mathbf{z}_4 \, \mathrm{d}s.$$

Putting it all together, the first integral of equation (3.12) becomes

$$\left(\mathbf{u}', n \cdot \left((\mathcal{F}^v_{\nabla \mathbf{u}})^\top \nabla \mathbf{z} \right) \right)_{\Gamma_W} = \int_{\Gamma_W} v' \cdot (n \cdot \Sigma) \, \mathrm{d}s + \int_{\Gamma_W} T'\kappa n \cdot \nabla \mathbf{z}_4 \, \mathrm{d}s. \qquad (3.15)$$

For the second integral of equation (3.12) the local shape derivative of the convective and viscous fluxes at the no-slip wall are needed. Inserting the no-slip condition via the relation (3.14) into to convective and the viscous fluxes yields

$$(\mathcal{F}^c)' = \begin{pmatrix} \rho v_1 & \rho v_2 \\ \rho v_1^2 + p & \rho v_1 v_2 \\ \rho v_1 v_2 & \rho v_2^2 + p \\ \rho H v_1 & \rho H v_2 \end{pmatrix}' = \begin{pmatrix} \rho v'_1 & \rho v'_2 \\ p' & 0 \\ 0 & p' \\ \rho H v'_1 & \rho H v'_2 \end{pmatrix} \qquad \text{on } \Gamma_W$$

and

$$(\mathcal{F}^v)' = \begin{pmatrix} 0 & 0 \\ \tau'_{11} & \tau'_{12} \\ \tau'_{21} & \tau'_{22} \\ \sum_j \tau_{1j} v'_j + \kappa \frac{\partial T'}{\partial x_1} & \sum_j \tau_{2j} v'_j + \kappa \frac{\partial T'}{\partial x_2} \end{pmatrix} \quad \text{on } \Gamma_W.$$

Inserting these terms into the second integral of equation (3.12) results in

$$\begin{aligned} &\left(n \cdot \left((\mathcal{F}^c(\mathbf{u}))' - (\mathcal{F}^v(\mathbf{u}, \nabla \mathbf{u}))'\right), \mathbf{z}\right)_{\Gamma_W} \\ &= \int_{\Gamma_W} (p'n - \tau' n) \cdot \mathbf{z}_{2,3} + v' \cdot (\rho n \mathbf{z}_1 + (\rho H n - \tau n)\mathbf{z}_4) - \nabla T' \cdot n \kappa \mathbf{z}_4 \, \mathrm{d}s. \end{aligned} \tag{3.16}$$

Combining the results of (3.15) and (3.16), the shape derivative of the Navier–Stokes equations equation (3.12) becomes

$$\begin{aligned} 0 = &\int_{\Gamma_W} v' \cdot (n \cdot \Sigma) \, \mathrm{d}s + \int_{\Gamma_W} T' \kappa n \cdot \nabla \mathbf{z}_4 \, \mathrm{d}s \\ &+ \int_{\Gamma_W} (p'n - \tau' n) \cdot \mathbf{z}_{2,3} + v' \cdot (\rho n \mathbf{z}_1 + (\rho H n - \tau n)\mathbf{z}_4) - \nabla T' \cdot n \kappa \mathbf{z}_4 \, \mathrm{d}s \\ &+ \boxed{((V \cdot n)(\nabla \cdot (\mathcal{F}^c - \mathcal{F}^v)), \mathbf{z})_{\Gamma_W}}, \end{aligned} \tag{3.17}$$

where distinction between the pointwise and the variational approach is still present in the framed term. This formula now includes the no-slip boundary condition and the explicit variations of the primal variables. What remains is the subtraction of this equation from the preliminary shape derivative of the cost function and the incorporation of further primal and adjoint boundary conditions as well as local shape derivatives of boundary conditions to finally reach the Hadamard form.

3.4.4. Subtraction of the shape derivative of the Navier–Stokes equations from the preliminary shape derivative of the cost function

The last step to finally reach the Hadamard form of the shape derivative of the lift and drag coefficient is to subtract the shape derivative of the Navier–Stokes equations equation (3.17) from the preliminary shape derivative of the cost function (3.3):

$$
\begin{aligned}
&d\mathcal{J}(\Omega;V)\\
=&\frac{1}{C_\infty}\int_{\Gamma_W}(p'n-\tau'n)\cdot\theta^{\boxed{4}}+(V\cdot n)\operatorname{div}(p\theta-\tau\theta)^{\boxed{5}}\,\mathrm{d}s\\
&-\int_{\Gamma_W}v'\cdot(n\cdot\Sigma)\,\mathrm{d}s-\int_{\Gamma_W}T'\kappa n\cdot\nabla\mathbf{z}_4\,\mathrm{d}s\\
&-\int_{\Gamma_W}(p'n-\tau'n)\cdot\mathbf{z}_{2,3}{}^{\boxed{6}}+v'\cdot(\rho n\mathbf{z}_1+(\rho Hn-\tau n)\mathbf{z}_4)-\nabla T'\cdot n\kappa\mathbf{z}_4\,\mathrm{d}s\\
&-\boxed{((V\cdot n)(\nabla\cdot(\mathcal{F}^c-\mathcal{F}^v)),\mathbf{z})_{\Gamma_W}}^{\boxed{7}}.
\end{aligned}
$$

Again, framed numbers are used to label certain terms. If the adjoin boundary condition $\mathbf{z}_{2,3}=\frac{1}{C_\infty}\theta$ on Γ_W is fulfilled, the terms $\boxed{4}$ and $\boxed{6}$ cancel each other out. This adjoint boundary condition also implies that the $\mathbf{z}_{2,3}$-component of expression $\boxed{7}$ vanish with term $\boxed{5}$, because

$$
\begin{aligned}
(\nabla\cdot(\mathcal{F}^c-\mathcal{F}^v))_{2,3}\cdot\mathbf{z}_{2,3}&=(\nabla\cdot[(\rho v_iv_j)_{ij}+pI-\tau])\cdot\mathbf{z}_{2,3}\\
&=\operatorname{div}(pI-\tau)\cdot\frac{1}{C_\infty}\theta=\frac{1}{C_\infty}\operatorname{div}(p\theta-\tau\theta)
\end{aligned}
$$

is fulfilled due to $\frac{\partial\rho v_iv_j}{\partial x_k}=\frac{\partial\rho v_i}{\partial x_k}v_j+\rho v_i\frac{\partial v_j}{\partial x_k}=0$ at the no-slip boundary. In case of the pointwise approach, term $\boxed{7}$ is not present. One might think that in that case there is no possible way to cancel term $\boxed{5}$, but this term, being the conservation of momentum, equals zero anyway if the Navier–Stokes equations are fulfilled pointwise. Altogether, the only framed terms remaining from the above formula are the first and last component of term $\boxed{7}$. Marking them with $\boxed{7_{1,4}}$ one obtains

$$
\begin{aligned}
d\mathcal{J}(\Omega;V)=&-\int_{\Gamma_W}v'\cdot(n\cdot\Sigma)\,\mathrm{d}s-\int_{\Gamma_W}T'\kappa n\cdot\nabla\mathbf{z}_4\,\mathrm{d}s\\
&-\int_{\Gamma_W}v'\cdot(\rho n\mathbf{z}_1+(\rho Hn-\tau n)\mathbf{z}_4)-\nabla T'\cdot n\kappa\mathbf{z}_4\,\mathrm{d}s\\
&-\boxed{((V\cdot n)(\nabla\cdot(\mathcal{F}^c-\mathcal{F}^v))_{1,4},\mathbf{z}_{1,4})_{\Gamma_W}}^{\boxed{7_{1,4}}}.
\end{aligned}
\tag{3.18}
$$

The previous task to eliminate p' and τ' is accomplished, but only by introducing the local shape derivatives of the velocity and the temperature. At first it seems to be unrewarding to reformulate the equations to get rid of some local shape derivatives and at the same time introducing new ones. But the crucial difference is that only local shape derivatives of quantities remain, for which boundary conditions at the wall exist. Therefore, the variations of the no-slip, adiabatic and isothermal boundary conditions are needed. The shape derivatives for general Dirichlet or Neumann boundary conditions can be found in [85] and lead to the following expressions

$$\begin{aligned}
v' &= -(V \cdot n)\frac{\partial v}{\partial n} && \text{on } \Gamma_W,\\
T' &= (V \cdot n)\frac{\partial T_W - T}{\partial n} && \text{on } \Gamma_{iso},\\
\nabla T' \cdot n &= -(V \cdot n)\frac{\partial^2 T}{\partial n^2} + \nabla T \cdot \nabla_\Gamma (V \cdot n) && \text{on } \Gamma_{adia}.
\end{aligned}$$

Replacing the local shape derivatives in equation (3.18) with these conditions at the respective parts of the wall results in

$$\begin{aligned}
d\mathcal{J}(\Omega;V) = &\int_{\Gamma_W} (V \cdot n)\frac{\partial v}{\partial n} \cdot (n \cdot \Sigma)\,\mathrm{d}s\\
&- \int_{\Gamma_{iso}} (V \cdot n)\frac{\partial T_W - T}{\partial n}\kappa n \cdot \nabla \mathbf{z}_4\,\mathrm{d}s - \int_{\Gamma_{adia}} T'\kappa n \cdot \nabla \mathbf{z}_4\,\mathrm{d}s^{\boxed{8}}\\
&+ \int_{\Gamma_W} (V \cdot n)\frac{\partial v}{\partial n} \cdot (\rho n \mathbf{z}_1 + (\rho H n - \tau n)\mathbf{z}_4)\,\mathrm{d}s\\
&+ \int_{\Gamma_{iso}} \nabla T' \cdot n\kappa \mathbf{z}_4\,\mathrm{d}s^{\boxed{9}} - \int_{\Gamma_{adia}} (V \cdot n)\frac{\partial^2 T}{\partial n^2}\kappa \mathbf{z}_4\,\mathrm{d}s\\
&+ \int_{\Gamma_{adia}} \nabla T \cdot \nabla_\Gamma (V \cdot n)\kappa \mathbf{z}_4\,\mathrm{d}s^{\boxed{10}}\\
&- \boxed{((V \cdot n)(\nabla \cdot (\mathcal{F}^c - \mathcal{F}^v)_{1,4}), \mathbf{z}_{1,4})_{\Gamma_W}}^{\boxed{7_{1,4}}}.
\end{aligned}$$

Since at the isothermal wall there is only a condition for the temperature and not for its gradient of the temperature, still a $\nabla T'$ term remains at this part of the wall. The same holds for the adiabatic wall with T'. These last two local shape derivatives $\boxed{8}$ and $\boxed{9}$ vanish if the following adjoint boundary conditions

$$\mathbf{z}_4 = 0, \quad \text{on } \Gamma_{iso} \qquad \text{and} \qquad \nabla \mathbf{z}_4 \cdot n = 0, \quad \text{on } \Gamma_{adia}$$

are fulfilled. Herewith, the goal to remove all local shape derivatives from the shape derivative of the cost function is finished. It remains the reformulation into Hadamard form, that means into an expression of the form $\int (V \cdot n) \ldots \mathrm{d}s$. The only term not being of this structure is term $\boxed{10}$. Applying the tangential Green's formula from theorem 3.8 to this integral yields

$$\begin{aligned}\int_{\Gamma_{adia}} \nabla T\cdot\nabla_\Gamma(V\cdot n)\kappa\mathbf{z}_4\,\mathrm{d}s &\\ &= \int_{\Gamma_{adia}} (V\cdot n)K(\nabla T\cdot n)\kappa\mathbf{z}_4 - (V\cdot n)\operatorname{div}_\Gamma(\nabla T\kappa\mathbf{z}_4)\,\mathrm{d}s\\ &= -\int_{\Gamma_{adia}} (V\cdot n)\operatorname{div}_\Gamma(\nabla T\kappa\mathbf{z}_4)\,\mathrm{d}s,\end{aligned}$$

where the adiabatic wall condition $\nabla T\cdot n = 0$ was used in the second line.

Additionally, an expression for the term $\boxed{7_{1,4}}$ is given. Using again the no-slip condition at the wall one gets for the first component $\boxed{7_1}$

$$\nabla\cdot(\mathcal{F}^c - \mathcal{F}^v)_1 = \nabla\cdot(\rho v_1, \rho v_2) = \rho(\nabla\cdot v)$$

and for the forth component

$$\begin{aligned}\nabla\cdot(\mathcal{F}^c - \mathcal{F}^v)_4 &\\ &= \nabla\cdot(\rho H v_1, \rho H v_2) - \nabla\cdot\left(\sum_j \tau_{1j}v_j + \kappa\frac{\partial T}{\partial x_1}, \sum_j \tau_{2j}v_j + \kappa\frac{\partial T}{\partial x_2}\right)\\ &= \rho H(\nabla\cdot v) - \sum_{i,j}\tau_{ij}\frac{\partial v_j}{\partial x_i} - \kappa\Delta T.\end{aligned}$$

In total, the shape derivative of the lift or drag coefficient in Hadamard form is given by

$$\begin{aligned}d\mathcal{J}(\Omega;V) = &\int_{\Gamma_W}(V\cdot n)\frac{\partial v}{\partial n}\cdot(n\cdot\Sigma)\,\mathrm{d}s\\ &- \int_{\Gamma_{iso}}(V\cdot n)\frac{\partial T_W - T}{\partial n}\kappa n\cdot\nabla\mathbf{z}_4\,\mathrm{d}s\\ &+ \int_{\Gamma_W}(V\cdot n)\frac{\partial v}{\partial n}\cdot(\rho n\mathbf{z}_1 + (\rho H n - \tau n)\mathbf{z}_4)\,\mathrm{d}s\\ &- \int_{\Gamma_{adia}}(V\cdot n)\left(\frac{\partial^2 T}{\partial n^2}\kappa\mathbf{z}_4 + \operatorname{div}_\Gamma(\nabla T\kappa\mathbf{z}_4)\right)\mathrm{d}s \qquad (3.19)\\ &- \boxed{\int_{\Gamma_W}(V\cdot n)\rho(\nabla\cdot v)\mathbf{z}_1\,\mathrm{d}s}\\ &- \boxed{\int_{\Gamma_{adia}}(V\cdot n)\left(\rho H(\nabla\cdot v) - \sum_{i,j}\tau_{ij}\frac{\partial v_j}{\partial x_i} - \kappa\Delta T\right)\mathbf{z}_4\,\mathrm{d}s}.\end{aligned}$$

Although, there is only this single derivation of the Hadamard form for both approaches, the difference of the variational and the pointwise approach still exist. The framed terms in the last equation only appear when considering the Navier–Stokes equations in variational form. Referring to theorem 3.4, all terms inside the integrals of equation (3.19) except the vector field $(V \cdot n)$ build a scalar distribution

$$\begin{aligned} g(\Gamma) = & \frac{\partial v}{\partial n} \cdot (n \cdot \Sigma) - \frac{\partial T_W - T}{\partial n} \kappa n \cdot \nabla \mathbf{z}_4 \\ & + \frac{\partial v}{\partial n} \cdot (\rho n \mathbf{z}_1 + (\rho H n - \tau n) \mathbf{z}_4) - \left(\frac{\partial^2 T}{\partial n^2} \kappa \mathbf{z}_4 + \operatorname{div}_\Gamma (\nabla T \kappa \mathbf{z}_4) \right) \\ & - \boxed{\rho (\nabla \cdot v) \mathbf{z}_1 \, \mathrm{d}s} - \boxed{\left(\rho H (\nabla \cdot v) - \sum_{i,j} \tau_{ij} \frac{\partial v_j}{\partial x_i} - \kappa \Delta T \right) \mathbf{z}_4}, \end{aligned}$$

which are in the following called shape gradient for simplicity.

3.5. Numerical comparison

The differences between the pointwise and the variational approach are now investigated numerically. Therefore, the Discontinuous Galerkin solver PADGE, developed primarily at the German Aerospace Center (DLR), Braunschweig, is extended to calculate the above Hadamard form of the shape derivative, for both drag and lift coefficient. The major reason for choosing this code for the implementation of the shape derivative is that this code already includes a solver for the adjoint problem, which was originally implemented for error estimation in a *h*-/*p*-refinement framework. Here, the adjoint solution together with the primal solution is used to calculate the Hadamard form (3.19) directly.

3.5.1. High order grid generation and perturbation

The Hadamard form only depends on quantities at the wall and especially includes geometrical quantities like the normal vector. To gain maximum accuracy in this high order framework, it is therefore necessary to approximate the shape of the airfoil appropriately. Later on, calculations up to a polynomial degree of $N = 5$ will be shown which also require meshes of sufficient order. For this investigation the grid is set up in the following way: Basis is a very fine structured linear hexahedral mesh which only has straight faces. This fine grid is agglomerated to a spatially coarser grid where multiple cells are combined to a single element with curved boundaries, see figure 3.2. Even so the resulting grid is coarser in the sense of

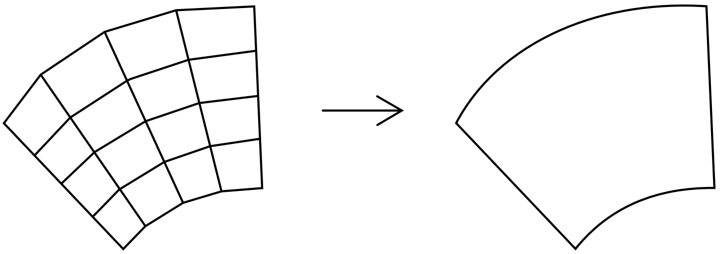


Figure 3.2.: Generation of curved meshes by agglomerating multiple straight grid cells to a single grid cell of fourth polynomial order.

number of elements, the geometrical information is preserved in the curvature of the elements. The flow solver internally builds a fourth order polynomial mapping for each coarse element which maps the reference element to the physical element. Hence, the boundary of each element is a polynomial of fourth order and the airfoil a piecewise curve of such polynomials.

The results obtained from the Hadamard form of the shape derivative of the cost function are validated with first order finite differences. They are computed by modifying the mappings from the reference element to the physical elements for each wall boundary element separately. For all modifications a separate flow solution is performed. Together with a baseline flow this enables the evaluation of finite differences for all edges at the airfoil. The mappings are transformed using a polynomial f satisfying at the wall boundary

$$\begin{aligned} f(0) = 0, \quad f'(0) = 0, \\ f(1) = 0, \quad f'(1) = 0, \\ f(0.5) = \varepsilon, \end{aligned}$$

where ε is the finite difference step-length and the line segment in physical dimensions is normalized to the interval $[0, 1]$ with midpoint 0.5. The conditions that the derivative f at the corners of the element should be zero ensure a smooth transition between the elements. The other conditions define the "height" of the deformation, which is the step size of the finite differencing in the middle of the edge and of course must be zero at the corners of the element. An exaggerated visualization of such a deformation "bump" is plotted in figure 3.3. Defining such a polynomial f for all elements at the airfoil enables the computation of finite differences for each edge individually.

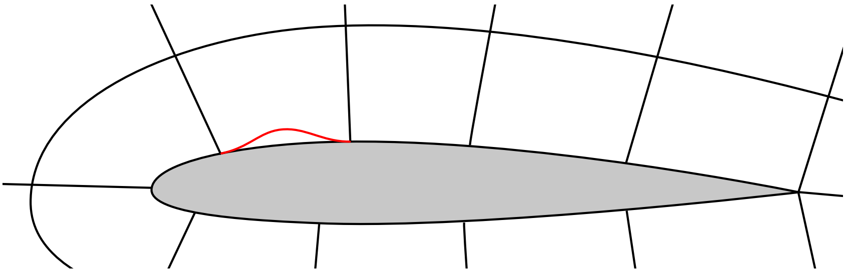


Figure 3.3.: Exaggerated visualization of the deformation "bump" of an airfoil along a single edge.

Since the Hadamard form in equation (3.19) and the cost function are boundary integrals, a Gauss quadrature rule is needed to evaluate them numerically. To preserve the overall accuracy, a quadrature rule of order eight is used which results in nine quadrature points along each edge of the airfoil. This very high order for the numerical quadrature allows precise integration even of products of multiple polynomials of order five or lower and is sufficient for all terms in the Hadamard form (3.19). The shape gradient in Hadamard form is evaluated at all these points, at first without multiplying it with the normal amount of the vector field $(V \cdot n)$. Since all other quantities in the Hadamard form are independent from the factor $(V \cdot n)$, only a single primal and a single adjoint solve are needed for this calculations. To compare this shape gradient with the results from the finite differences, each polynomial deformation f is transformed into an effective physical boundary movement, represented by a corresponding vector field V. For all edges of the airfoil the above shape gradient is then multiplied with the respective $(V \cdot n)$ term to build the Hadamard form, which can be compared to the finite difference results directly.

3.5.2. Test and verification setup

The verification of the shape derivatives in Hadamard form is performed on the ADIGMA MTC3 test case [86], where a flow of Mach $M = 0.5$ and a Reynolds number of $Re = 5000$ around a NACA0012 airfoil with an angle of attack $\alpha = 2.0^\circ$ is computed. This test case is already included in the PADGE framework and very suitable for this investigations since it is thoroughly verified. The computational grid consist of 1640 curved cells with 40 curved boundary edges of polynomial degree four representing the airfoil. The influence of accuracy of

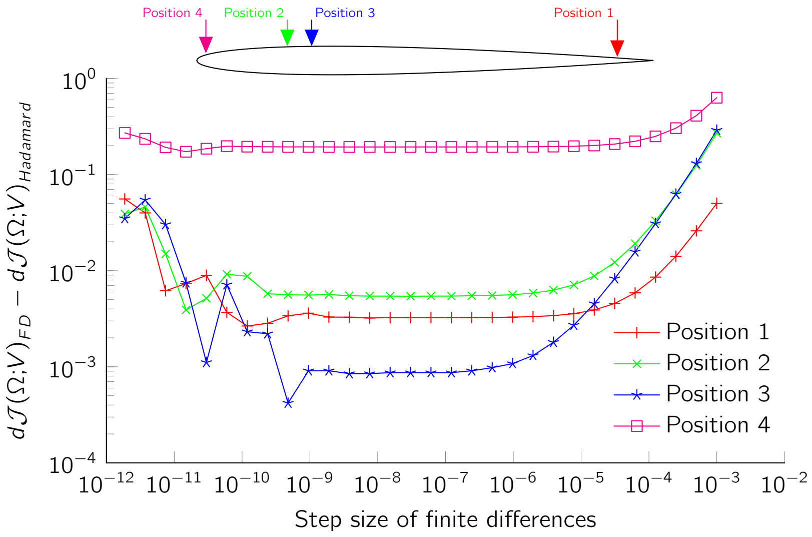


Figure 3.4.: Error of FD-shape gradient versus shape gradient in variational Hadamard form over different FD step sizes at four different positions.

the numerical solution to the Hadamard form is studied using different polynomial degrees for the Galerkin ansatz of the primal solution ranging from three to five. The adjoint solution is computed with a polynomial degree being always one degree higher, so four to six. For each polynomial degree the shape derivative in Hadamard form is verified with finite differences computed with the same polynomial degree, which are computed in the following way. With the above "bump"-functions f, each edge of the profile is disturbed individually and a flow computation is performed to calculate the drag and lift coefficient corresponding to a modification of the respective edge. An extra baseline computation gives the drag and lift coefficients for the original geometry, which is then used to build the finite differences for each edge. This process requires a big numerical effort, since it already needs 41 complete flow calculations for this simple example to build the shape gradient with finite differences. It is clear that evaluating the shape derivative in this way in a more complex or even in a three dimensional setting is impossible and already for this small example requires a lot of computational time.

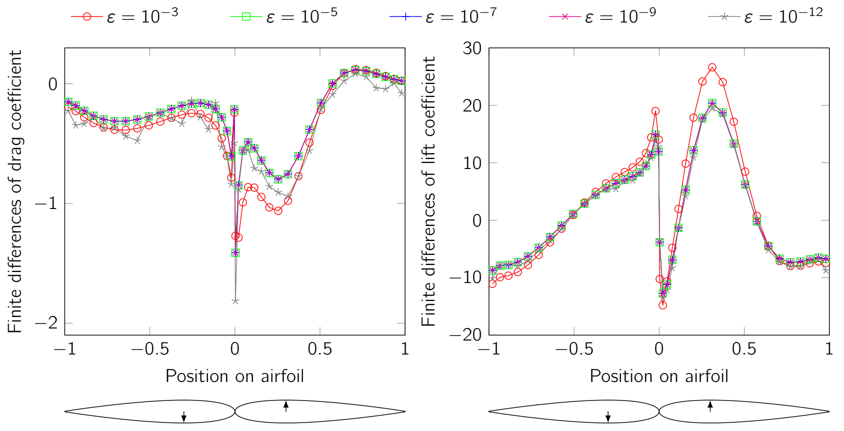


Figure 3.5.: Finite differences of the shape gradient for drag and lift coefficient at five different step sizes.

Another drawback of the finite difference approach is the step size ε. A priori, the absolute value one has to take is absolutely unknown and has to be determined with numerical experiments. Therefore, the absolute difference of the shape derivative in variational Hadamard form and the finite differences is computed for step sizes ranging from 10^{-12} to 10^{-3} at four different positions of the airfoil. Thus, this value can be interpreted as error of the finite differences with respect to the varying perturbation parameter ε. These errors are plotted in figure 3.4. Even so there is a quite substantial error at the nose of the airfoil, one can see that for ε between 10^{-9} and 10^{-5} all curves are flat. This means that the error in this region is independent from the chosen step size. However, the absolute values of these errors differ a lot between the four positions and it is per se unclear if the behavior is the same for other positions, for example on the bottom side of the airfoil. In figure 3.5, therefore, the finite differences of the shape gradient are plotted as a function of the spacial position to fixed perturbation sizes ε. To stay within the bounds of the possible numerical effort only five different step sizes are analyzed for all positions. In total this requires the computation of over 200 flow solutions. Since the curves for $\varepsilon = 10^{-5}, 10^{-7}$ and 10^{-9} perfectly coincide both for drag and lift coefficient the conclusions are the same as above. Consequently, the step size for the finite difference reference solution to validate the Hadamard form was chosen to be $\varepsilon = 10^{-7}$ which is between the lower and upper bound of the suitable range and is very robust with respect to cancellation errors.

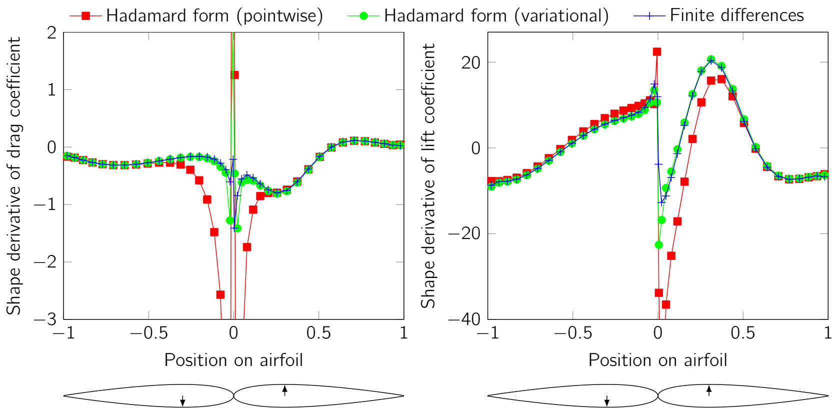


Figure 3.6.: Shape gradient of drag and lift coefficient for a polynomial degree of $N = 3$.

3.5.3. Comparison of the pointwise and the variational shape derivative

The shape derivatives of the drag and lift coefficient in Hadamard form for the variational and the pointwise approach are now compared to each other and to the finite differences. Figure 3.6 shows on the left hand side the shape derivative of the drag coefficient and on the right hand side the shape derivative of the lift coefficient. All calculations in this plot are performed using a polynomial degree of three for the primal solution. For both cost functions the variational approach very nicely matches with the finite differences while the pointwise approach shows noticeably deviations, especially at the leading edge of the airfoil.

Increasing the polynomial degree of the flow solution to $N = 4$, see figure 3.7, leads to a better match of both Hadamard forms to the finite differences. However, there is still an unmissable gap between the pointwise approach and the finite differences around the nose of the profile. In contrast to that, the agreement of the shape derivative stemming from the variational approach with the finite differences is remarkably good.

A further enhancement of the accuracy of the flow solution by increasing the polynomial to $N = 5$ retains this trend. In figure 3.8, an excellent match between the variational approach and the finite differences can be seen for both the drag and the lift coefficient. For the pointwise approach the shape derivative of the lift coefficient now also shows a good behavior. Nevertheless, for the drag coeffi-

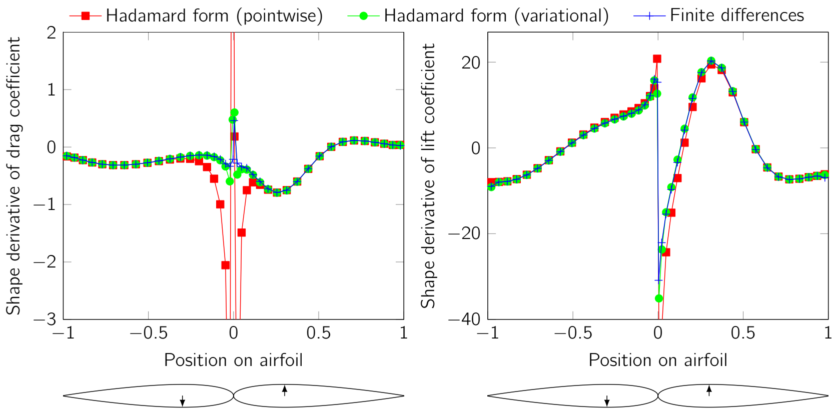


Figure 3.7.: Shape gradient of drag and lift coefficient for a polynomial degree of $N = 4$.

cient there is still an obvious difference, even so this difference diminished when compared to the lower degree cases. This behavior of the pointwise approach in contrast to the variational approach can be explained as follows. The primal and adjoint flow solver used to calculate the numerical solution is a Discontinuous Galerkin solver, which is based on the weak form of the Navier–Stokes equations equations. Hence, it is only capable of computing a solution that fulfills the governing equations in a weak sense and not in the strong pointwise manner. Therefore, it is clear that the comparison of the shape derivative of the variational approach with the finite differences already for a polynomial degree of $N = 3$ shows a very good agreement. In contrast to that, the shape derivative of the pointwise approach starts with a poor accordance and becomes better with each refinement step even so it does not reach the same accuracy as the variational approach. The reason for this improvement is that for higher polynomial degrees, the governing equations are computed more accurately and therewith, the solution becomes closer in fulfilling the pointwise equations.

Having accurate shape derivatives at hand, the natural next step is to apply them in a real design optimization. A recent trend in the field of optimization is the one-shot optimization method, where primal and dual iteration of the flow solver are executed simultaneously and even the design update is performed in each iteration [33, 74, 76]. Hence, shape gradients are evaluated for inexact and not fully converged flow and adjoint solutions until the residuals go to zero.

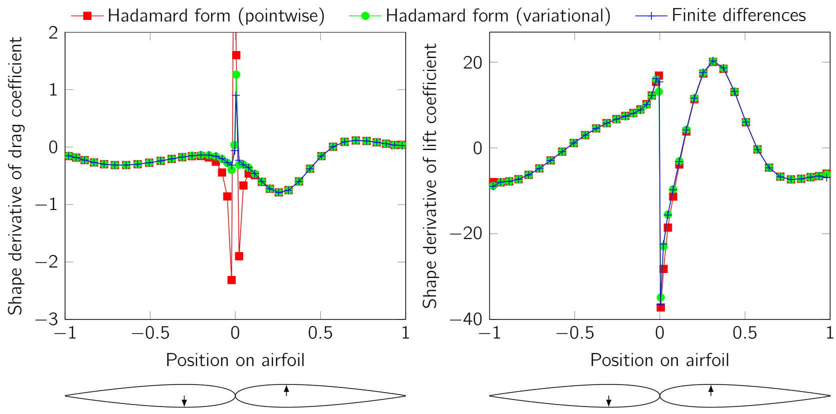


Figure 3.8.: Shape gradient of drag and lift coefficient for a polynomial degree of $N = 5$.

Therefore, using the appropriate shape derivative, which matches to the weak formulation the flow solver is based on, is of crucial importance. In [48] the here presented shape derivatives are applied to an one-shot airfoil optimization comparable to the above setting, where it is additionally coupled with an adaptive mesh refinement strategy. What remains is the application to flow situations where a strong solution is questionable and the full benefit of the variational approach should be visible. This is especially the case for high mach flows involving a shock or flow discontinuity. Due to limitations in the flow solver PADGE used for the above results, it is not possible to compute such cases. This would require a stable shock capturing for the Discontinuous Galerkin method as it will be presented in chapter 5.

4. Discontinuous Galerkin spectral element method

In the previous chapter shape derivatives for the Navier–Stokes equations have been presented. The numerical results were limited to subsonic cases since the used flow solver lacked a proper shock capturing for its Discontinuous Galerkin method. To overcome this restriction a shock capturing for the Discontinuous Galerkin method will be presented in chapter 5. This shock capturing is implemented into the open source flow solver *FLEXI*[1] developed in the Numerics Research Group (NRG) of the Institute of Aero- and Gasdynamics (IAG) at the University of Stuttgart. The flow solver *FLEXI* is based on the Discontinuous Galerkin spectral element method, which is briefly introduced in this chapter to give a suitable foundation for the formulation of the shock capturing afterwards.

The Discontinuous Galerkin method was originally proposed by Reed and Hill [68] in 1973. It is a combination of Finite Element (FE) and Finite Volume (FV), where all elements are independently approximated with local high order ansatz functions. Since this leads to possible discontinuous states at element interfaces, the elements are coupled using numerical fluxes, which are well-known from the Finite Volume method. This idea of coupling different domains weakling was even earlier, in 1971, introduced by Nitsche [61]. Later, Cockburn and Shu developed in a series of papers [15, 16, 18, 19, 20] a solid theoretical framework for the Discontinuous Galerkin scheme.

The Discontinuous Galerkin method is based on the weak formulation of the Navier–Stokes equations in equation (2.7), already transformed to the reference space. The main difference to the FE method is that the states are allowed to be discontinuous across element interfaces, which leads to double-valued states at the interface between two elements. To obtain an overall conservative scheme and handle these double valued states, the interface fluxes have to be uniquely defined for both adjacent elements and therefore, they are approximated with flux functions based on Riemann solvers known from the Finite Volume method. In the weak formulation of equation (2.7) the flux at the element interfaces is replaced with a numerical flux $((\mathcal{F}^c - \mathcal{F}^v) \cdot \mathcal{N})^*$, which is indicated with a superscript

[1]www.flexi-project.org

asterisks and leads to the following equation

$$\int_E J\mathbf{u}_t \Phi \,\mathrm{d}\xi - \int_E \mathcal{F} \cdot \nabla\Phi \,\mathrm{d}\xi + \int_{\partial E} (\mathcal{F} \cdot \mathcal{N})^* \,\Phi \,\mathrm{d}S_\xi = 0, \tag{4.1}$$

where additionally for simplicity the convective flux $\mathcal{F}^c$ and the viscous flux $\mathcal{F}^v$ are combined in a single flux $\mathcal{F} = \mathcal{F}^c - \mathcal{F}^v$. This equation is now discretized using polynomial tensor product ansatz functions.

Remark 4.1. *Comparing the Discontinuous Galerkin method to the Finite Element method, the double-valued states at the interface lead to a larger number of degrees of freedom. This introduces, especially for lower polynomial degrees of the ansatz and test functions, a significant extra cost, but has the following advantages. While the FE method has a global mass matrix, the mass matrix of the DG method is local and cheaper to invert. For the Discontinuous Galerkin spectral element method, as considered here, the mass matrix becomes diagonal and therefore trivial to invert. The second advantage of the non unique states is the ability to introduce interface fluxes which allow to prefer one state over the other depending on the flow direction, also known as upwinding.*

4.1. Numerical approximation

The solution inside each element E is approximated by a polynomial tensor product basis of degree N in each space direction

$$\mathbf{u}(\xi, t) \approx \sum_{i,j,k=0}^{N} \hat{\mathbf{u}}_{ijk}(t)\psi_{ijk}(\xi), \tag{4.2}$$

where $\hat{\mathbf{u}}_{ijk}(t)$ are the nodal degrees of freedom and

$$\psi_{ijk} = \ell_i(\xi^1)\ell_j(\xi^2)\ell_k(\xi^3)$$

are the tensor product basis functions consisting of the one-dimensional Lagrange interpolation polynomials

$$\ell_i(\xi) = \prod_{\substack{k=0 \\ k\neq i}}^{N} \frac{\xi - \xi_k}{\xi_i - \xi_k},$$

which are defined by the nodes of the Gauss quadrature rule $\{\xi_i\}_{i=0}^N$ in the interval $[-1, 1]$. The Lagrange interpolation polynomials fulfill the Lagrange property

$$\ell_i(\xi_j) = \delta_{ij} \quad \forall i, j = 0, \ldots, N. \tag{4.3}$$

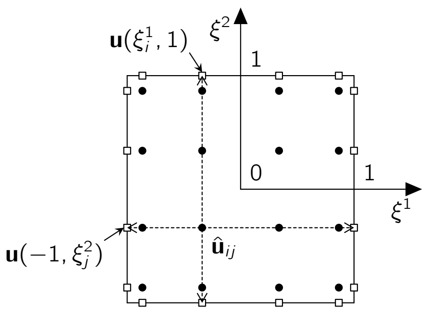


Figure 4.1.: DG reference element E in two space dimensions for a polynomial degree of $N = 3$ with Gauss points • and locations of the boundary fluxes ▫ at the DG interface.

This property will be used later several times to simplify the derivation of the discrete DG scheme. Since this is a Galerkin scheme, for the test function Φ the basis functions $\phi = \psi_{ijk}$ are used.

The same approximation as for the solution is used for each component of the contravariant fluxes $\mathcal{F}^m$

$$\mathcal{F}^m(\mathbf{u}, \nabla\mathbf{u}) \approx \sum_{i,j,k=0}^{N} \hat{\mathcal{F}}^m_{ijk} \psi_{ijk}(\xi), \tag{4.4}$$

where $\hat{\mathcal{F}}^m_{ijk}$ is the nodal interpolation of the m-th component of the flux at the ijk-th node, which is calculated by evaluating the physical flux with the nodal values and transforming it using the metric terms

$$\hat{\mathcal{F}}^m_{ijk} = \sum_{m=1}^{3} Ja^m_d(\xi^1_i, \xi^2_j, \xi^3_k) F_d(\mathbf{u}(\xi^1_i, \xi^2_j, \xi^3_k), \nabla\mathbf{u}(\xi^1_i, \xi^2_j, \xi^3_k)).$$

For the approximation of the fluxes at the element interfaces the solution at the element boundaries is needed. Since the Gauss interpolation points are all located inside of the element, the solution must be extrapolated to the element boundaries. Due to the tensor product structure of the ansatz functions this can be done one-dimensionally along the ξ^1-, ξ^2- and the ξ^3-lines. A two dimensional example is given in figure 4.1. With the boundary states of two adjacent elements, the numerical flux is then calculated in every flux point of their interface using Riemann solvers.

These approximations for the solution and the fluxes can now be inserted into the integrals of the weak formulation, which are approximated in the following by using the Gauss quadrature rule. Since the interpolation is already based on Gauss points, the integration can use the same points. With this collocation of interpolation and integration as well as the Lagrange property (4.3) the formulas will be significantly simplified. In the next three subsections the approximations are consecutively inserted into the three integrals of the weak formulation (4.1).

4.1.1. Time derivative integral

Inserting the approximation of the solution (4.2) into the integral of equation (4.1) containing the time derivative leads to

$$\frac{\partial}{\partial t}\int_E J\mathbf{u}\Phi \,\mathrm{d}\xi = \frac{\partial}{\partial t}\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} J(\xi)\sum_{m,n,o=0}^{N}\left(\hat{\mathbf{u}}_{mno}\psi_{mno}(\xi)\right)\psi_{ijk}(\xi)\,\mathrm{d}\xi^1\,\mathrm{d}\xi^2\,\mathrm{d}\xi^3 \qquad \forall i,j,k,$$

where the integration over the reference element is split into the three coordinate directions. Approximating these integrals with the Gauss quadrature and inserting the tensor product definition of the ansatz functions yields

$$\begin{aligned}
&=\frac{\partial}{\partial t}\sum_{\lambda,\mu,\nu=0}^{N} J(\xi_{\lambda\mu\nu})\sum_{m,n,o=0}^{N}\left(\hat{\mathbf{u}}_{mno}\,\underbrace{\ell_m(\xi^1_\lambda)}_{\delta_{m\lambda}}\,\underbrace{\ell_n(\xi^2_\mu)}_{\delta_{n\mu}}\,\underbrace{\ell_o(\xi^3_\nu)}_{\delta_{o\nu}}\right)\psi_{ijk}(\xi_{\lambda\mu\nu})\omega_\lambda\omega_\mu\omega_\nu\\
&=\frac{\partial}{\partial t}\sum_{m,n,o=0}^{N} J(\xi_{mno})\hat{\mathbf{u}}_{mno}\psi_{ijk}(\xi_{mno})\omega_m\omega_n\omega_o \qquad \forall i,j,k,
\end{aligned}$$

where $\{\omega_i\}_{i=0}^N$ are the weights of the Gauss quadrature. Inserting the same tensor product basis for the test functions and again using the Lagrange property gives

$$\begin{aligned}
&=\frac{\partial}{\partial t}\sum_{m,n,o=0}^{N} J(\xi_{mno})\hat{\mathbf{u}}_{mno}\,\underbrace{\ell_i(\xi^1_m)}_{\delta_{im}}\,\underbrace{\ell_j(\xi^2_n)}_{\delta_{jn}}\,\underbrace{\ell_k(\xi^3_o)}_{\delta_{ko}}\,\omega_m\omega_n\omega_o\\
&=J(\xi_{ijk})\frac{\partial\hat{\mathbf{u}}_{ijk}}{\partial t}\omega_i\omega_j\omega_k \qquad \forall i,j,k.
\end{aligned}\tag{4.5}$$

4.1.2. Volume integral

For the volume integral of the weak formulation (4.1), the scalar product can be written as the sum of the three contravariant flux components

$$\int_E \mathcal{F} \cdot \nabla \Phi \, \mathrm{d}\xi = \sum_{d=1}^{3} \int_E \mathcal{F}^d \frac{\partial \phi}{\partial \xi^d} \, \mathrm{d}\xi.$$

Since all three integrals of this sum have the same structure one can exemplarily consider only the integral in the first direction $d = 1$. After inserting the approximation of the fluxes (4.4) and the test function one gets

$$\int_E \mathcal{F}^1 \frac{\partial \phi}{\partial \xi^1} \, \mathrm{d}\xi = \int_E \sum_{m,n,o=0}^{N} \hat{\mathcal{F}}^1_{mno} \psi_{mno}(\xi) \frac{\partial \psi_{ijk}(\xi)}{\partial \xi^1} \, \mathrm{d}\xi.$$

Using the tensor product basis, the Gauss quadrature and the Lagrange property yields

$$\begin{aligned}
\int_E \mathcal{F}^1 \frac{\partial \phi}{\partial \xi^1} \, \mathrm{d}\xi &= \sum_{\lambda,\mu,\nu=0}^{N} \sum_{m,n,o=0}^{N} \hat{\mathcal{F}}^1_{mno} \underbrace{\ell_m(\xi^1_\lambda)}_{\delta_{m\lambda}} \underbrace{\ell_n(\xi^2_\mu)}_{\delta_{n\mu}} \underbrace{\ell_o(\xi^3_\nu)}_{\delta_{o\nu}} \frac{\partial \psi_{ijk}(\xi_{\lambda\mu\nu})}{\partial \xi^1} \omega_\lambda \omega_\mu \omega_\nu \\
&= \sum_{\lambda,\mu,\nu=0}^{N} \hat{\mathcal{F}}^1_{ijk} \frac{\partial \psi_{ijk}(\xi_{\lambda\mu\nu})}{\partial \xi^1} \omega_\lambda \omega_\mu \omega_\nu \\
&= \sum_{\lambda,\mu,\nu=0}^{N} \hat{\mathcal{F}}^1_{ijk} \left. \frac{\partial \ell_i(\xi^1)}{\partial \xi^1} \right|_{\xi=\xi^1_\lambda} \underbrace{\ell_j(\xi^2_\mu)}_{\delta_{j\mu}} \underbrace{\ell_k(\xi^3_\nu)}_{\delta_{k\nu}} \omega_\lambda \omega_\mu \omega_\nu \\
&= \omega_j \omega_k \sum_{\lambda=0}^{N} \hat{\mathcal{F}}^1_{\lambda jk} \left. \frac{\partial \ell_i(\xi^1)}{\partial \xi^1} \right|_{\xi=\xi^1_\lambda} \omega_\lambda .
\end{aligned} \tag{4.6}$$

Defining the differentiation matrix

$$D_{ab} = \left. \frac{\partial \ell_b(\xi)}{\partial \xi} \right|_{\xi=\xi_a} \quad \text{with } a, b = 0, \dots, N$$

one can interpret the last sum in equation (4.6) as one-dimensional matrix vector product. Repeating the last steps also for $d = 2$ and $d = 3$ the whole volume integral becomes

$$\begin{aligned}
&\int_E \mathcal{F} \cdot \nabla \Phi \, \mathrm{d}\xi \\
&= \omega_j \omega_k \sum_{\lambda=0}^{N} \hat{\mathcal{F}}^1_{\lambda jk} D_{\lambda i} \omega_\lambda + \omega_i \omega_k \sum_{\mu=0}^{N} \hat{\mathcal{F}}^2_{i\mu k} D_{\mu j} \omega_\mu + \omega_i \omega_j \sum_{\nu=0}^{N} \hat{\mathcal{F}}^3_{ij\nu} D_{\nu k} \omega_\nu .
\end{aligned} \tag{4.7}$$

4.1.3. Surface integral

Similar to the volume integral, the surface integral can be split into the different directions in reference space and be evaluated at $\xi^1 = \{-1, 1\}$, $\xi^2 = \{-1, 1\}$ and $\xi^3 = \{-1, 1\}$ independently

$$\begin{aligned} \int_{\partial E} (\mathcal{F} \cdot \mathcal{N})^* \, \Phi \, \mathrm{d}S_\xi = & \left[\int_{-1}^{1} \int_{-1}^{1} (\mathcal{F} \cdot \mathcal{N})^* \, \Phi \, \mathrm{d}\xi^2 \, \mathrm{d}\xi^3 \right]_{\xi^1=-1}^{1} \\ & + \left[\int_{-1}^{1} \int_{-1}^{1} (\mathcal{F} \cdot \mathcal{N})^* \, \Phi \, \mathrm{d}\xi^1 \, \mathrm{d}\xi^3 \right]_{\xi^2=-1}^{1} \\ & + \left[\int_{-1}^{1} \int_{-1}^{1} (\mathcal{F} \cdot \mathcal{N})^* \, \Phi \, \mathrm{d}\xi^1 \, \mathrm{d}\xi^2 \right]_{\xi^3=-1}^{1} . \end{aligned} \tag{4.8}$$

To keep the derivation simple only the $\xi^1 = 1$ face is investigated. All other faces are treated the same way. At the $\xi^1 = 1$ face, the outward pointing normal vector in the reference element is given by $\mathcal{N} = (1, 0, 0)^\top$ and the numerical flux reduces to

$$\mathcal{F} \cdot \mathcal{N} = \mathcal{F}^1.$$

Given the surface element $\hat{s}$ and the unit normal vector in physical space n by

$$\hat{s} = \sqrt{\sum_{d=1}^{3} \left(Ja_d^1(1, \xi^2, \xi^3) \right)^2}, \quad n_d = \frac{Ja_d^1}{\hat{s}}$$

and with the transformation from equation (2.6) this flux can be written as

$$\mathcal{F}^1 = \sum_{d=1}^{3} Ja_d^1(1, \xi^2, \xi^3) F_d(1, \xi^2, \xi^3) = (F \cdot n)\hat{s},$$

where F is the physical flux. Using Riemann solvers $f^*(\mathbf{u}_L, \mathbf{u}_R, n)$ in the physical normal direction n, the numerical flux can be expressed as

$$(\mathcal{F} \cdot \mathcal{N})^* = (F \cdot n)^* \hat{s} = f^*(\mathbf{u}_L, \mathbf{u}_R, n)\hat{s},$$

which only depends on the left and right state at the interface and the unit normal vector. Again this flux is approximated at Gauss points, but this time in Gauss points at the respective interface as depicted in figure 4.1

$$(\mathcal{F} \cdot \mathcal{N})^* = \sum_{m,n=0}^{N} \left[f^*(\mathbf{u}_L, \mathbf{u}_R, n)\hat{s} \right]_{m,n}^{+\xi^1} \ell_m(\xi^2) \ell_n(\xi^3), \tag{4.9}$$

where the superscript $+\xi^1$ indicates the $\xi^1 = 1$ face and the subindices m, n denote the evaluation of the Riemann solver at the m, n-th Gauss point, with the respective states and physical normal vector in these positions. Inserting equation (4.9) and the test function into corresponding part of equation (4.8) and using Gauss quadrature as well as the Lagrange property (4.3) results in

$$\begin{aligned}
\int_{-1}^{1}\int_{-1}^{1} (\mathcal{F}\cdot\mathcal{N})^* \,\Phi \,\mathrm{d}\xi^2\,\mathrm{d}\xi^3\Big|_{\xi^1=1}
&= \sum_{\mu,\nu=0}^{N}\left(\sum_{m,n=0}^{N} [f^*\hat{s}]^{+\xi^1}_{m,n}\, \underbrace{\ell_m(\xi^2_\mu)}_{\delta_{m\mu}}\underbrace{\ell_n(\xi^3_\nu)}_{\delta_{n\nu}}\right) \ell_i(1)\,\underbrace{\ell_j(\xi^2_\mu)}_{\delta_{j\mu}}\underbrace{\ell_k(\xi^3_\nu)}_{\delta_{k\nu}}\,\omega_\mu\omega_\nu \\
&= [f^*\hat{s}]^{+\xi^1}_{j,k}\,\ell_i(1)\omega_j\omega_k,
\end{aligned}$$

where the arguments of the Riemann solver where omitted. Repeating these steps for all other faces finally gives the whole surface integral

$$\begin{aligned}
\int_{\partial E} (\mathcal{F}\cdot\mathcal{N})^*\,\Phi\,\mathrm{d}S_\xi = \quad &\left([f^*\hat{s}]^{+\xi^1}_{j,k}\,\ell_i(1) - [f^*\hat{s}]^{-\xi^1}_{j,k}\,\ell_i(-1)\right)\omega_j\omega_k \\
+&\left([f^*\hat{s}]^{+\xi^2}_{i,k}\,\ell_j(1) - [f^*\hat{s}]^{-\xi^2}_{i,k}\,\ell_j(-1)\right)\omega_i\omega_k \\
+&\left([f^*\hat{s}]^{+\xi^3}_{i,j}\,\ell_k(1) - [f^*\hat{s}]^{-\xi^3}_{i,j}\,\ell_k(-1)\right)\omega_i\omega_j.
\end{aligned} \tag{4.10}$$

4.1.4. Semi-discrete formulation

Putting equations (4.5), (4.7) and (4.10) together, the semi-discrete formulation of the weak formulation, where the time is still continuous, becomes

$$\begin{aligned}
\frac{\partial \hat{\mathbf{u}}_{ijk}}{\partial t} = -\frac{1}{J_{ijk}}\Bigg[\quad &\sum_{\lambda=0}^{N} \hat{\mathcal{F}}^1_{\lambda jk}\hat{D}_{i\lambda} + \left([f^*\hat{s}]^{+\xi^1}_{j,k}\,\hat{\ell}_i(1) - [f^*\hat{s}]^{-\xi^1}_{j,k}\,\hat{\ell}_i(-1)\right) \\
+&\sum_{\mu=0}^{N} \hat{\mathcal{F}}^2_{i\mu k}\hat{D}_{j\mu} + \left([f^*\hat{s}]^{+\xi^2}_{i,k}\,\hat{\ell}_j(1) - [f^*\hat{s}]^{-\xi^2}_{i,k}\,\hat{\ell}_j(-1)\right) \\
+&\sum_{\nu=0}^{N} \hat{\mathcal{F}}^3_{ij\nu}\hat{D}_{k\nu} + \left([f^*\hat{s}]^{+\xi^3}_{i,j}\,\hat{\ell}_k(1) - [f^*\hat{s}]^{-\xi^3}_{i,j}\,\hat{\ell}_k(-1)\right)\Bigg],
\end{aligned} \tag{4.11}$$

where the precomputed one-dimensional operators are defined by

$$\hat{\ell}_i = \frac{\ell_i}{\omega_i}, \quad \hat{D}_{ij} = -\frac{\omega_i}{\omega_j}D_{ij}, \qquad i,j = 0,\dots,N.$$

Each of the lines of equation (4.11) corresponds to the operator for one space dimension and the extension to higher space dimensions or the restriction to the two or one-dimensional case is straightforward. This dimension-by-dimension structure of the semi-discrete operator is a direct consequence of the tensor product ansatz in conjunction with the collocation of interpolation and integration points.

4.1.5. Gradient approximation for second order equations

For the above derivation of the Discontinuous Galerkin spectral element method, the convective flux $\mathcal{F}^c(\mathbf{u})$ and the viscous flux $\mathcal{F}^v(\mathbf{u}, \nabla\mathbf{u})$ were combined to a single flux $\mathcal{F}$ to keep the notation simple. As long as the governing equation is a first order equation that only depends on the solution $\mathbf{u}$ and not on the gradient of the solution $\nabla\mathbf{u}$, like the Euler equations, the semi-discrete formulation (4.11) is sufficient for the approximation of this equation and can directly be implemented. Nevertheless, the viscous flux of the Navier–Stokes equations depends on the gradient of the solution. The idea is to rewrite the second order equation into a first order system and then apply the same DG discretization also for the additional gradient quantity. This procedure is called lifting and, therewith, leads to a DG scheme inside of the original DG scheme. In this work for the numerical flux of the gradient equation the BR1 or BR2 scheme of Bassi and Rebay [7, 8] is used. A short summary of the BR1 discretization for DGSEM can be found in [39]. For a more detailed view, including a comparison of BR1 and BR2, one may read [38].

4.2. Time integration

So far, time is still a continuous quantity and the whole derivation of the Discontinuous Galerkin method above is independent of the time integration used to advance the solution numerically in time. In general, any suitable time integration, implicit or explicit, can be used. The implementation in this work is restricted to the explicit Runge-Kutta time integration. Based on the work of Williamson [91], who developed two-register Runge-Kutta schemes, Kennedy et al. [49] derived five-stage fourth order methods that have optimized coefficients, especially for the compressible Navier-Stokes equations. Low storage explicit Runge-Kutta schemes of third and fourth order are used in this work. These have time step restrictions depending on the spatial discretization, the physical element sizes and fastest signal velocities.

4.2.1. Time step restriction

The time step of the explicit Runge-Kutta time integration is a direct result of the CFL condition by Courant, Friedrichs and Lewy [21] and reads

$$\Delta t \leq \mathrm{CFL} \cdot \alpha_{RK}(N) \frac{\Delta x}{\lambda^c (2N+1)}, \tag{4.12}$$

where CFL is the CFL number, Δx the physical element size, λ^c the maximal signal velocity and $\alpha_{RK}(N)$ a scaling factor of the Runge-Kutta time integration for the Discontinuous Galerkin method that depends on the polynomial degree N of the spatial discretization. The factor $\frac{1}{2N+1}$ takes into account the fact that the DOFs inside an element are not distributed equidistantly [17]. The CFL number for this explicit method has to be chosen smaller than 1 to get a stable solution. The maximal signal velocity λ^c is given by the eigenvalues of the inviscid Euler equations and depends on the fluid velocity v and the sound speed c

$$\lambda^c = \max(v - c, v, v + c).$$

In the implementation, the above inequality (4.12) is transformed to the reference space and therewith the minimum element size becomes 2, which is the size of the reference element in each space direction. This transformation of course changes the fluid velocity and the sound speed by the metric terms, but the overall structure remains the same.

The time step restriction in equation (4.12) is only valid for convection dominated problems, described by the hyperbolic part of the Navier–Stokes equations, the Euler equations. For the parabolic terms of the Navier–Stokes equations one has to consider an additional viscous time step restriction [27]

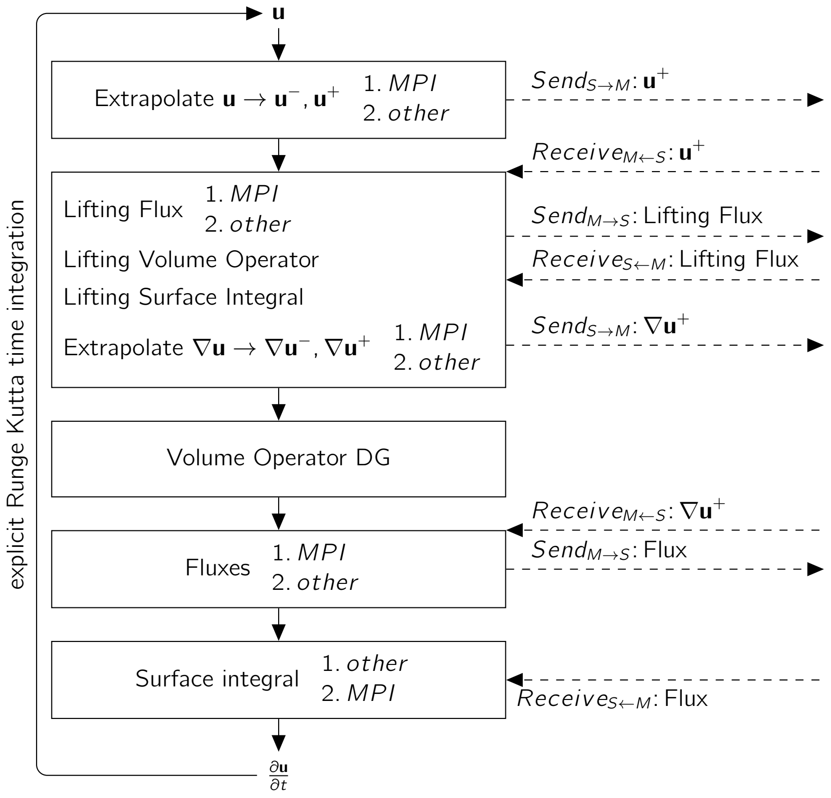
$$\Delta t \leq \mathrm{DFL} \cdot \beta_{RK}(N) \frac{\Delta x^2}{\lambda^v (2N+1)^2}, \tag{4.13}$$

where λ^v denotes the maximal eigenvalue of the diffusion matrix, DFL is the counterpart of the CFL number for diffusion and $\beta_{RK}(N)$ is a scaling factor of the Runge-Kutta time integration for the diffusion.

To guarantee stability one has to take the global minimum of these two time steps.

4.3. Overview of the implementation

The Discontinuous Galerkin scheme using explicit Runge-Kutta time integration can be implemented as shown in figure 4.2. The outer loop is the time integration



$\mathbf{u}$
Extrapolate $\mathbf{u} \rightarrow \mathbf{u}^-, \mathbf{u}^+$
1. *MPI*
2. *other*
$Send_{S\rightarrow M}$: $\mathbf{u}^+$
$Receive_{M\leftarrow S}$: $\mathbf{u}^+$
Lifting Flux
1. *MPI*
2. *other*
$Send_{M\rightarrow S}$: Lifting Flux
$Receive_{S\leftarrow M}$: Lifting Flux
Lifting Volume Operator
Lifting Surface Integral
Extrapolate $\nabla\mathbf{u} \rightarrow \nabla\mathbf{u}^-, \nabla\mathbf{u}^+$
1. *MPI*
2. *other*
$Send_{S\rightarrow M}$: $\nabla\mathbf{u}^+$
Volume Operator DG
$Receive_{M\leftarrow S}$: $\nabla\mathbf{u}^+$
$Send_{M\rightarrow S}$: Flux
Fluxes
1. *MPI*
2. *other*
Surface integral
1. *other*
2. *MPI*
$Receive_{S\leftarrow M}$: Flux
$\frac{\partial \mathbf{u}}{\partial t}$
explicit Runge Kutta time integration

Figure 4.2.: Flow chart of the Discontinuous Galerkin operator.

loop. Based on the solution $\mathbf{u}$, the DG operator calculates the derivative of this state $\frac{\partial \mathbf{u}}{\partial t}$ with respect to time, which is used in the Runge-Kutta time integration to advance in time. Step-by-step, the DG operator computes the single terms of the semi-discrete formulation (4.11). This will be explained for a parallel execution of the code where each processor computes a certain portion of all elements in the domain. The communication between the processors is performed by the Message Passing Interface (MPI). First of all, the solution is extrapolated from the inner volume Gauss points to the interfaces of the elements. This is done for all so called slave MPI interfaces at first so that the solution at these interfaces can be transmitted immediately from this processor to the adjacent one. The labels 'master' and 'slave' are assigned to the two adjacent elements of an interface. Even so there are two states at the interface, one extrapolated from the master and one from the slave element of an interface, the flux at the interface is of course computed only once. The flux evaluation is always performed on the master side. Hence, the slave side must provide the extrapolated solution at the interface first. To distinguish the master and slave states they are marked with superscripts $\cdot^{-}$ and $\cdot^{+}$. At MPI interfaces the slave state must be transmitted to the processor holding the master element. After starting this non-blocking communication, the solution is extrapolated to all the other sides. This division of the extrapolation operator into MPI and non-MPI interface enables the start of the communication as early as possible and therewith hides the communication of the face solution behind the calculation of the extrapolation to the non-MPI interface. This technique of split and hide is necessary to achieve a good parallel performance and is applied to the subsequent communications as well.

The next block contains the lifting operator which calculates the gradients of the solution using the BR1 or BR2 scheme of Bassi and Rebay. As stated above, this requires a Discontinuous Galerkin method for the gradient equation which consists of the same components as the overall DG method. First, the flux for the lifting is computed. This is done using the already received face states of the adjacent elements. Again this is done only on one of the two sides of an interface and therefore the lifting flux has to be sent from the master to the slave elements. During this transmission the volume operator of the lifting is evaluated to hide the communication. Afterwards, the lifting surface integral can be calculated. By this, the gradient is given as a polynomial defined on the same Gauss interpolation points as the solution, but for the calculation of the fluxes it is later also needed at the interfaces. Therefore, the last step of the lifting is to extrapolate the gradients to the interfaces and again send them from the slave to the master elements.

While the transmission of the gradients is going on the volume operator, which is the evaluation of the volume integral from equation (4.7), is performed.

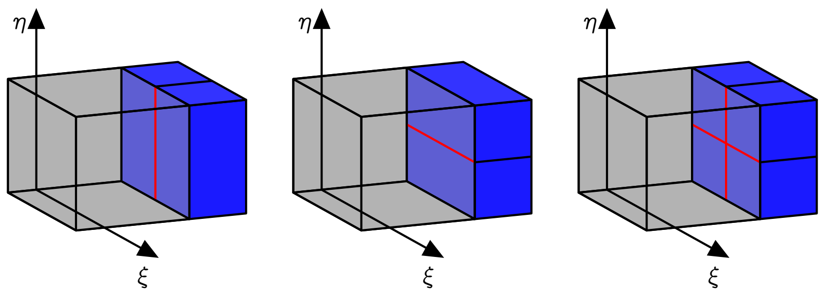
After that, the face gradients of the solution should be received and the fluxes can be calculated. This is done using Riemann solvers and corresponds to the evaluation of the f^* terms in equation (4.10). Again, first for the MPI interfaces to start the transmission of the fluxes from the master to the slave elements as early as possible.

The last step is to integrate the fluxes over the surface, as in equation (4.10). This time this is done for the non-MPI interfaces first since this gives some extra time for the flux transfer.

4.4. Nonconforming meshes using mortar interfaces

The application of numerical methods to more realistic examples beyond academic test cases introduces several additional restrictions. Besides difficulties like nontrivial boundary conditions or initial conditions, the approximation of rather complex geometries is an issue that especially becomes crucial in the mesh generation process. On the one hand the Discontinuous Galerkin method allows, due its high order, the use of quite large elements in comparison to low order methods. On the other hand fine geometrical structures require small cells to resolve the surface of obstacles. This conflict can partly be resolved by using curved elements to represent the geometry. Besides this, the most interesting parts of the flow often arise due to the interaction of the fluid with the objects. Hence, quite small elements are needed around the geometry to properly resolve these flow features. In contrast to that, the farfield normally is quite smooth and can be handled with large elements. For a computational efficient method it is therefore necessary to reduce the costs by coarsening the grid in the smooth parts of the domain and only refine the regions of interest. To achieve this, the ability of handling non-conforming interfaces gives the user a huge benefit and a big freedom in the mesh generation process.

For the discontinuous spectral element method Kopriva [52] introduced the mortar technique to handle non-conforming interfaces. The idea is to split the face of a large element into smaller faces of the adjacent little elements. Restricting the division to only halving the big face one ends up with at most three different types of mortar faces in 3D. A big face can be halved vertical or horizontal or in both directions, see figure 4.3. These types on the one hand keep the implementation as simple as possible and on the other hand are very common in mesh generators for non-conforming, purely hexahedral meshes.



(a) Type 1 **(b)** Type 2 **(c)** Type 3

Figure 4.3.: All three possible mortar types. The first and second types only have a different reference direction by which the interface between the big and the two small elements is split. The last type is a combination of the types 1 and 2.

4.4.1. Mortar interfaces for DG

The introduction of mortar interfaces into DGSEM, where the possible non-conforming interfaces are restricted to the three cases of figure 4.3, is quite easy. The basic idea is to interpolate the solution of the big side to virtual small sides which are conforming with the small sides of the adjacent smaller elements. Due to the fact that the virtual face and the face of the small element match, the face solution of the small element and the solution from the big element (on the virtual face) are conforming and can directly be used to calculate the flux on the small side. This is done using the same Riemann solvers as for the conform case. On each small side this flux can then be applied to the small side element immediately. For the big side the small side fluxes are projected to the big side and then applied to the big element. In short this reads:

1. Interpolation of big side solution to the small virtual sides.

2. Use Riemann solvers to calculate fluxes on small conform sides.

3. Apply fluxes to small elements.

4. Project small side fluxes to big side and apply to big element.

As explained above, the steps 2 and 3 are preformed on conform faces and do not differ from the respective steps of a solver without mortar interfaces. Therefore,

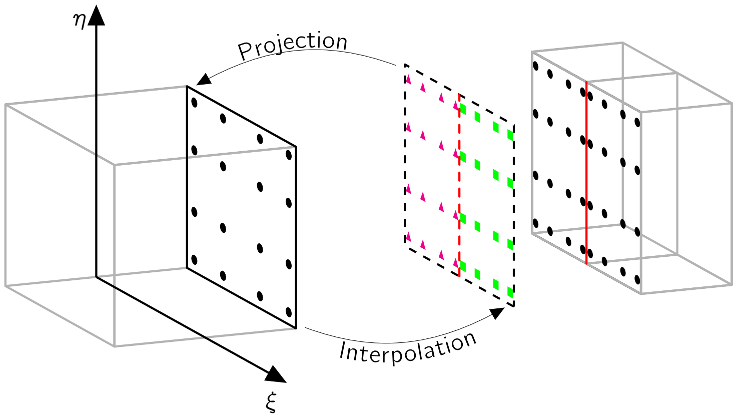


Figure 4.4.: Interpolation points at a mortar interface. Fluxes are always computed on the points of the small sides.The solution of the big side (at black dots) must be interpolated to both small sides individually. Therefore, virtual small sides (magenta triangles and green squares) are introduced which store the interpolated solution of the big side. The flux is computed on the conforming small sides and then projected back to the big side.

only the steps 1 and 4 are of interest and will be explained in the following. Besides this, due to the tensor product structure, it is sufficient to concentrate on type 1 only. For type 2 one just applies the operators in the other space direction. Type 3 finally is the concatenation of type 1 and 2 operators.

In figure 4.4 the interpolation points of the big side (black dots) and of the two adjacent small sides as well as the virtual sides (magenta triangles, green squares) for a type 1 mortar interface, where the big side is split in ξ direction, are visualized. It is clear that the positions in η direction for all three involved faces are the same. Therefore, the solution must only be interpolated to new positions in ξ direction. The Gauss points in the reference interval $[-1,1]$ are mapped to the left and right half of this interval using the mappings

$$\begin{aligned} [-1,1] &\to [-1,0] \\ \xi &\mapsto \frac{\xi-1}{2} \end{aligned} \qquad \text{and} \qquad \begin{aligned} [-1,1] &\to [0,1] \\ \xi &\mapsto \frac{\xi+1}{2}. \end{aligned}$$

With these mappings, the ξ-coordinates of the interpolation points ξ^L and ξ^R of the two small sides in the coordinate system of the big side are given by

$$\xi^L \in \left\{ \frac{\xi_i^B - 1}{2} : i = 0, \dots, N \right\}, \qquad \xi^R \in \left\{ \frac{\xi_i^B + 1}{2} : i = 0, \dots, N \right\},$$

where the superscripts B, L and R denote the big, left and right sides. Using the same superscripts also for the nodal degrees of freedom one can interpolate the discrete big side solution $\mathbf{u}^B$ to the positions of the interpolation points of the left (magenta triangles) and right (green squares) side, which gives the discrete solutions in the respective small coordinate system

$$\begin{array}{c|c} \begin{aligned} \mathbf{u}^L[i,j] &= \sum_{\lambda,\mu=0}^{N} \mathbf{u}^B[\lambda,\mu]\ell_\lambda(\xi_i^L)\underbrace{\ell_\mu(\eta_j)}_{\delta_{\mu j}} \\ &= \sum_{\lambda=0}^{N} \mathbf{u}^B[\lambda,j]\ell_\lambda(\xi_i^L) \end{aligned} & \begin{aligned} \mathbf{u}^R[i,j] &= \sum_{\lambda,\mu=0}^{N} \mathbf{u}^B[\lambda,\mu]\ell_\lambda(\xi_i^R)\underbrace{\ell_\mu(\eta_j)}_{\delta_{\mu j}} \\ &= \sum_{\lambda=0}^{N} \mathbf{u}^B[\lambda,j]\ell_\lambda(\xi_i^R), \end{aligned} \end{array}$$

where the notation $[i,j]$ is used to access the i,j-th entry of an array. Defining the matrices

$$\begin{aligned} I_{i\lambda}^L &= \ell_\lambda(\xi_i^L), \qquad \xi_i^L \in \left\{ \frac{\xi_i^B - 1}{2} : i = 0, \dots, N \right\}, \\ I_{i\lambda}^R &= \ell_\lambda(\xi_i^R), \qquad \xi_i^R \in \left\{ \frac{\xi_i^B + 1}{2} : i = 0, \dots, N \right\}, \end{aligned}$$

one can write the interpolation of the nodal values from the big side to the nodal values of the two small side as

$$\begin{aligned} \mathbf{u}^L[i,j] &= \sum_{\lambda=0}^{N} I^L[i,\lambda]\mathbf{u}^B[\lambda,j] \\ \mathbf{u}^R[i,j] &= \sum_{\lambda=0}^{N} I^R[i,\lambda]\mathbf{u}^B[\lambda,j], \end{aligned}$$

where for each fixed j this is a matrix vector multiplication. Using the colon to denote all entries of the specific index this can be written short as matrix vector multiplications for fixed j's

$$\mathbf{u}^L[:,j] = I^L[:,:] \cdot \mathbf{u}^B[:,j], \qquad \mathbf{u}^R[:,j] = I^R[:,:] \cdot \mathbf{u}^B[:,j] \qquad \forall j. \tag{4.14}$$

Due to the fact that the matrices I^L and I^R only depend on the one-dimensional interpolation to the left/right half, these matrices can directly be used for the type 2 mortar interface as well, where the big side is halved in η-direction. Using the superscript U (up) and D (down) to label the upper and lower half one gets

$$\mathbf{u}^D[i,:] = I^L[:,:] \cdot \mathbf{u}^B[i,:], \qquad \mathbf{u}^U[i,:] = I^R[:,:] \cdot \mathbf{u}^B[i,:] \qquad \forall i,$$

where for the η-direction the i-index is fixed. The only difference between these two types is the direction in which the operators are applied or in a discrete sense on which index the matrix-vector multiplications are performed.

Concatenating the type 1 and type 2 operations yields the interpolation to the four type 3 quadrants. For example, the interpolation to the upper left quadrant can be achieved by applying the I^L matrix to the already interpolated upper half solution $\mathbf{u}^U$.

After interpolating the solution of the big side to the small sides, the flux can be evaluated using the Riemann solvers of the conform case. The functions for the flux computation do not even have to know that the data originally comes from a non-conform interface. The fluxes then can directly be integrated over the two small sides separately and be applied to the DOFs of the respective small elements. For the big element side the fluxes of the two small side must first be projected to the big side. This is done using L^2 projection, following the ideas of [52]. Again only the type 1 case, where the ξ-direction is halved, is investigated. In the coordinate system of the big side, the L^2 projection corresponds to finding the flux that satisfies

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{0} \left(f^B(\xi^B,\eta) - 2\cdot f^L(2\xi^B+1,\eta)\right)\Phi \,\mathrm{d}\xi^B\,\mathrm{d}\eta \\ +&\int_{-1}^{1}\int_{0}^{1} \left(f^B(\xi^B,\eta) - 2\cdot f^R(2\xi^B-1,\eta)\right)\Phi \,\mathrm{d}\xi^B\,\mathrm{d}\eta = 0, \qquad \forall\Phi,\end{aligned}$$

where f^B, f^L and f^R denote the fluxes on the big side and the left and right small sides. The factor 2 in front of the small side fluxes f^L and f^R is the scaling factor between the reference coordinate systems of the big and the respective reference coordinate system of the small sides. Shifting the terms containing f^L and f^R to the right hand side and rewriting them in the coordinate system of the respective small side yields

$$\begin{aligned}\int_{-1}^{1}\int_{-1}^{1} f^B(\xi^B,\eta)\Phi\,\mathrm{d}\xi^B\eta = &\int_{-1}^{1}\int_{-1}^{1} f^L(\xi^L,\eta)\Phi\left(\tfrac{\xi^L-1}{2},\eta\right)\mathrm{d}\xi^L\,\mathrm{d}\eta \\ &+\int_{-1}^{1}\int_{-1}^{1} f^R(\xi^R,\eta)\Phi\left(\tfrac{\xi^R+1}{2},\eta\right)\mathrm{d}\xi^R\,\mathrm{d}\eta, \qquad \forall\Phi.\end{aligned}$$

Inserting the approximation of the fluxes and the function Φ as well as replacing the integration with Gauss quadrature leads to

$$\begin{aligned}
&\sum_{\lambda,\nu=0}^{N}\sum_{i,j=0}^{N} f_{ij}^{B}\underbrace{\ell_i(\xi_\lambda^B)}_{\delta_{i\lambda}}\underbrace{\ell_j(\eta_\nu)}_{\delta_{j\nu}}\underbrace{\ell_m(\xi_\lambda^B)}_{\delta_{m\lambda}}\underbrace{\ell_n(\eta_\nu)}_{\delta_{n\nu}}\omega_\lambda\omega_\nu \\
&\qquad = \sum_{\lambda,\nu=0}^{N}\sum_{i,j=0}^{N} f_{ij}^{L}\underbrace{\ell_i(\xi_\lambda^L)}_{\delta_{i\lambda}}\underbrace{\ell_j(\eta_\nu)}_{\delta_{j\nu}}\ell_m\left(\tfrac{\xi_\lambda^L-1}{2}\right)\underbrace{\ell_n(\eta_\nu)}_{\delta_{n\nu}}\omega_\lambda\omega_\nu \\
&\qquad\quad + \sum_{\lambda,\nu=0}^{N}\sum_{i,j=0}^{N} f_{ij}^{R}\underbrace{\ell_i(\xi_\lambda^R)}_{\delta_{i\lambda}}\underbrace{\ell_j(\eta_\nu)}_{\delta_{j\nu}}\ell_m\left(\tfrac{\xi_\lambda^R+1}{2}\right)\underbrace{\ell_n(\eta_\nu)}_{\delta_{n\nu}}\omega_\lambda\omega_\nu, \qquad \forall m,n.
\end{aligned}$$

Due to the Lagrange property (4.3) this reduces to

$$f_{ij}^{B} = \sum_{\lambda=0}^{N} f_{\lambda j}^{L}\ell_i\left(\tfrac{\xi_\lambda^L-1}{2}\right)\frac{\omega_\lambda}{\omega_i} + \sum_{\lambda=0}^{N} f_{\lambda j}^{R}\ell_i\left(\tfrac{\xi_\lambda^R+1}{2}\right)\frac{\omega_\lambda}{\omega_i}, \qquad \forall i,j. \tag{4.15}$$

Defining the projection matrices

$$P_{i\lambda}^{L} = \ell_i\left(\tfrac{\xi_\lambda-1}{2}\right)\frac{\omega_\lambda}{\omega_i} \qquad \text{and} \qquad P_{i\lambda}^{R} = \ell_i\left(\tfrac{\xi_\lambda+1}{2}\right)\frac{\omega_\lambda}{\omega_i} \qquad \forall i,\lambda = 0,\dots,N$$

the projection (4.15) reads as matrix-vector multiplication

$$f^{B}[:,j] = P^{L}[:,:]\cdot f^{L}[:,j] + P^{R}[:,:]\cdot f^{R}[:,j], \qquad \forall j, \tag{4.16}$$

which must be applied to all indices j separately. Again, one can see that as a result of the tensor product ansatz the projection is an one-dimensional operation which can be applied dimension by dimension. For the type 2 mortar interface, where the split is in η-direction, the projection matrices must be just applied to the second index. Finally the type 3 mortar interface is again the concatenation of these to operations. For a more general view on the mortar interfaces, including different polynomial degrees at the interface, the reader is referred to [52].

5. Shock capturing for the Discontinuous Galerkin method

High order methods in general, but especially the Discontinuous Galerkin method presented in the previous chapter, have restricted capabilities to resolve shock waves or other flow discontinuities. They can be extended to such flow situations with a variety of different techniques which are generally summarized under the term shock capturing. An improper shock capturing in the DG implementation of the PADGE code prevented the application of the shape derivative from chapter 3 to flow cases including shocks. Even so, the Discontinuous Galerkin method allows discontinuities particularly at the element boundaries, the high order polynomials representing the solution are unable to resolve inner cell jumps without oscillations. To show an example, figure 5.1 presents the density of the Sod shock tube problem after a single time step. This example is initialized with piecewise constant data in the left and right half of the domain, where the discontinuity is perfectly located at an element boundary. After a single time step of the DG method, the polynomial of degree $N = 5$ immediately starts to oscillate since the scheme is not able to resolve the propagation of the developing elementary waves.

To circumvent this issue different shock capturing approaches for the Discontinuous Galerkin scheme are available, but they all have the aim in common to diminish the oscillations that are generated from the high order polynomials for discontinuous data. One technique is to apply so called artificial viscosity [64, 6, 66], which adds viscosity locally to the original equations with the intention of smearing the solution such that it can be properly resolved by the polynomials. This approach was originally proposed by von Neumann and Richtmyer [90] for finite difference schemes. An adaption of this to high order Discontinuous Galerkin schemes, which eliminates the high frequencies without widening the shock over a several cells, was invented by Persson and Peraire [64].

Another approach is to combine the DG method with weighted essentially non-oscillatory (WENO) schemes. This idea was introduced by Qiu and Shu [67], who used a Hermite WENO method as limiter for Runge-Kutta DG schemes in shock regions. With the same concept Balsara et al. [5] extended this to hybrid RKDG+HWENO schemes, where they used indicators on sub-cells to detect the troubled zones. One limitation of these schemes, especially for parallel compu-

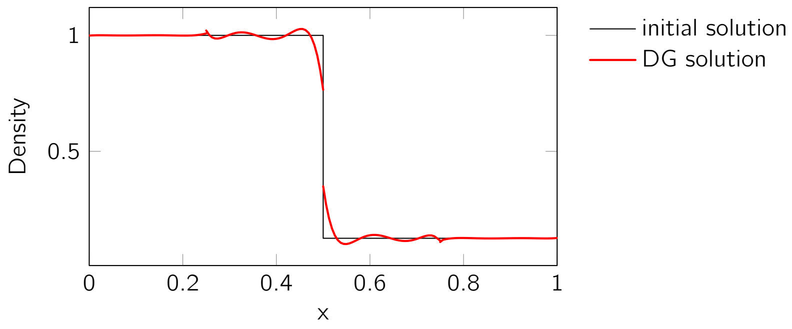


Figure 5.1.: The density of the Sod shock tube example for the pure DG scheme immediately starts oscillating after a single time step ($t = 0.005$).

tations, is that the size of their stencil increases with an increasing polynomial degree and hence, the amount of required communication between processors may become a crucial factor.

A further technique for shock capturing it to reduce the polynomial degree of the elements containing a discontinuity to low order. This is often combined with a local mesh refinement to keep the overall resolution, leading to the so called *h*-/*p*-refinement [3, 9]. This idea of spatial refinement with lower order ansatz functions was also exploited by Huerta et al. [41], where they extend the polynomial ansatz space with piecewise constant functions (low order) in sub-cells (*h*-refinement) of the original DG element. The basic concept of the *h*-/*p*-refinement is that the reduction to lower order polynomials decreases the oscillations and the spatial refinement compensates the loss in overall resolution.

A special variant of the last approach is presented in this section. The general idea is to replace the DG method in troubled elements with the Finite Volume (FV) scheme, which is perfectly suited to resolve shocks. For the FV method the solution in a cell is represented as a single integral mean value. Hence, only replacing the DG method in shock containing elements with the FV scheme would lead to a big loss of resolution, since the information of all DOFs of a DG element would be compressed into a single state. Therefore, the DG element is subdivided into several FV sub-cells such that each sub-cell contains exactly one nodal DOF of the DG scheme, see figure 5.2. This leads to $(N+1)^d$ sub-cells in the d-dimensional space and there are two main reasons for choosing exactly this subdivision of a DG element. First, the usage of the same number of FV sub-cells as the number of DOFs of a DG element enables to reuse the same data structures for both methods. Therewith, the solution can be stored in the same array independent of the scheme which is used to update an element. This technique is mandatory for an efficient implementation in a massive parallel environment. The second reason also concerns the efficiency of the scheme in a parallel environment. One big issue for a good scaling of computations on multiple processors is an even distribution of load among the processors. Therefore, the computational effort needed to update a DG element or a FV sub-cells element should be the same. Otherwise, travelling shocks would lead to strong load imbalances which would require a redistribution of the load among the processors. The costs for such load balancing steps cannot be neglected and they affect the overall performance gravely. Therefore, the aim is to build a shock capturing that does not require a load balancing.

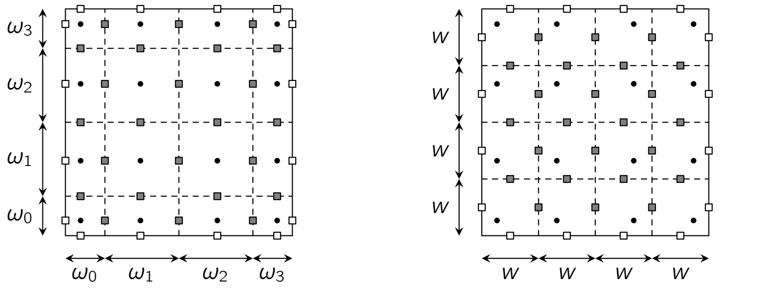
Remark 5.1 (Finite Volume sub-cells element)**.** *The entirety of Finite Volume sub-cells a DG element is split into, are called a "FV sub-cells element". This can be abbreviated in the following by omitting either "FV", "sub-cells" or "element". From the context or the plural of sub-cells it should always be clear that the whole block of Finite Volume sub-cells inside a DG element is meant. Especially the term "FV element" is not referring to a single FV sub-cell, but to the entire original element split up into sub-cells. If only a specific single FV sub-cell is addressed, the plural 's' is of course missing as well as the word "element".*

The derivation of the FV sub-cell shock capturing for Discontinuous Galerkin methods in this chapter is given for the two dimensional case, even though the implementation is so far 3D only. This keeps the notations and explanations simple and since everything is based on the tensor product structure of the DG method, the extension of the theory to any higher space dimension is straightforward.

5.1. Finite Volume method on sub-cells

The Finite Volume sub-cells method, like the Discontinuous Galerkin method in chapter 4, is derived in reference space. The reference element of the DG method is split into $(N+1)^2$ Finite Volume sub-cells such that for each Gauss point of the original DG interpolation there is one Finite Volume sub-cell. In general, the distribution of the sub-cells inside the reference element can be arbitrary, but here only two natural choices are investigated, see figure 5.2. The first way is to use the weights $\{\omega_i\}_i$ of the Gauss integration from the DG method as widths of the sub-cells in reference space. The second distribution is equidistant where all widths are of the same size and are given by

$$w = \frac{2}{N+1},$$



(a) Gaussian distributed FV sub-cells **(b)** Equidistant distributed FV sub-cells

Figure 5.2.: DG reference element split into FV sub-cells --- with Gauss points • of the original DG reference element, locations of the inner ■ and the interface □ boundary fluxes as well as the sizes ω_i and w of the sub-cells.

where 2 is the edge length of the reference element ranging from -1 to 1. In both cases the sub-cell corresponding to the ij-th Gauss point is labeled with κ_{ij}. Even so the sub-cells for the equidistant case are of the same size in reference space, they can become differently sized under the mapping to physical space, e.g. due to an inner cell stretching or a curving of the mesh. The Gaussian distribution of FV sub-cells was already covered in [84] and hence the focus will be on the equidistant case in the following.

Based on the transformed conservation law in reference space from equation (2.5), the Finite Volume sub-cells method is now formulated. Each sub-cell κ_{ij} becomes a control volume in the finite volume context and after integration over this sub-cell the equation reads

$$\int_{\kappa_{ij}} J\mathbf{u}_t \,\mathrm{d}\xi + \int_{\kappa_{ij}} \nabla_\xi \cdot \mathcal{F}(\mathbf{u}) \,\mathrm{d}\xi = 0 \quad \forall \kappa_{ij} \in E.$$

To get rid of the divergence in the second integral, the divergence theorem is applied to this term

$$\int_{\kappa_{ij}} J\mathbf{u}_t \,\mathrm{d}\xi + \int_{\partial\kappa_{ij}} \mathcal{F}(\mathbf{u}) \cdot \mathcal{N} \,\mathrm{d}S_\xi = 0 \quad \forall \kappa_{ij} \in E. \tag{5.1}$$

Until now the solution $\mathbf{u}$ is still continuous and will be discretized in space for the numerical scheme in the following.

Remark 5.2 (Block unstructured FV method)**.** *The Finite Volume sub-cells inside a DG element can be interpreted as a block of Finite Volumes sub-cells. Each DG element then becomes a block and using FV sub-cells everywhere one ends up with a blockwise Finite Volume method on unstructured curved hexahedral blocks. The blocks all have the same number of Finite Volume elements and are coupled through Riemann solvers to each other.*

5.1.1. Discretization

During the computation a shock may travel into a DG element and the algorithm must switch this DG element to Finite Volume sub-cells. To get an overall conservative scheme, the switching from DG to FV and back must not lead to a loss or growth in the conservative quantities. Therefore, the Finite Volume discretization must ensure to keep the integral mean value of the solution over each DG element constant during this switching process. This is especially the case if the integral mean value

$$\int_E \mathbf{u}\,\mathrm{d}\xi = \int_E \mathbf{u}_{DG}\,\mathrm{d}\xi = \int_E \mathbf{u}_{FV}\,\mathrm{d}\xi \tag{5.2}$$

of both discretization, DG and FV, is the same. For a numerical scheme this property must be enforced discretely. Inserting the approximation of the DG method from equation (4.2) into equation (5.2) this reads

$$\begin{aligned}\int_E \mathbf{u}\,\mathrm{d}\xi &= \int_E \sum_{i,j=0}^{N} \hat{\mathbf{u}}_{ij}\psi_{ij}(\xi)\,\mathrm{d}\xi \\ &= \sum_{\lambda,\mu=0}^{N}\sum_{i,j=0}^{N} \hat{\mathbf{u}}_{ij}\underbrace{\ell_i(\xi^1_\lambda)}_{\delta_{i\lambda}}\underbrace{\ell_j(\xi^2_\mu)}_{\delta_{j\mu}}\omega_\lambda\omega_\mu = \sum_{i,j=0}^{N}\hat{\mathbf{u}}_{ij}\omega_i\omega_j,\end{aligned} \tag{5.3}$$

where the integration over the reference element is performed with Gauss quadrature.

The Finite Volume formulation in equation (5.1) is now discretized for the two investigated distributions of FV sub-cells, Gaussian and equidistant. New degrees of freedom, which take the role of the average cell values in the Finite Volume context, are introduced and labeled with a double-dot hat

$$\hat{\ddot{\mathbf{u}}}_{ij}$$

to distinguish them from the DOFs with a single hat $\hat{\mathbf{u}}_{ij}$ of the DG method. The approximation of the Finite Volume formulation has to be chosen such that the Finite Volume DOFs $\hat{\ddot{\mathbf{u}}}_{ij}$ fulfill the above integral mean value condition in equation (5.2).

5.1.1.1. Finite Volume discretization for Gaussian sub-cells

For the discretization of Gaussian sub-cells, the sizes of the sub-cells in reference space correspond to the weights of the Gauss integration rule used in the Discontinuous Galerkin method. As explained above, the integral mean value is of major importance for conservation. Assuming that the state inside each sub-cell is constant and using the midpoint rule for integration leads to

$$\int_E \mathbf{u}\,\mathrm{d}\xi = \sum_{i,j=0}^{N} \int_{\kappa_{ij}} \hat{\mathbf{u}}_{ij}\,\mathrm{d}\xi = \sum_{i,j=0}^{N} \hat{\mathbf{u}}_{ij}\omega_i\omega_j, \tag{5.4}$$

where the average value inside each sub-cell is the corresponding DOF $\hat{\mathbf{u}}_{ij}$ and the sizes of the sub-cell κ_{ij} are ω_i in ξ-direction and ω_j in the η-direction, respectively. Comparing equations (5.3) and (5.4) one directly sees that the degrees of freedom of the FV sub-cells method are the same as for the DG method. The nodal representation $\{\hat{\mathbf{u}}_{ij}\}_{i,j=0}^{N} = \{\hat{\hat{\mathbf{u}}}_{ij}\}_{i,j=0}^{N}$ can be interpreted as DG solution and as FV sub-cells solution at the same time, which is the main advantage of the Gaussian distributed sub-cells over the equidistant distributed sub-cells.

5.1.1.2. Finite Volume discretization for equidistant sub-cells

In contrast to the Gaussian distribution, the degrees of freedom for the equidistant FV sub-cells differ from the DOFs of the DG method. The conversion between the FV and the DG discretization can be reduced to matrix-vector multiplications, which are applied dimension by dimension in a tensor product fashion. Therefore, and due to simplicity, the following derivation of the involved matrices can be performed in the one-dimensional case. In this case the reference element of the DG method reduces to the interval $[-1, 1]$ and for all terms/variables including indices i, j, the j-index is omitted. The Finite Volume sub-cells of the reference interval are then given by

$$[-1 + k \cdot w, -1 + (k+1) \cdot w] \quad \forall k = 0, \ldots, N,$$

where $w = \frac{2}{N+1}$ is the width of all sub-cells. The integral mean value of the solution is now split into these parts of the reference element

$$\int_E \mathbf{u}\,\mathrm{d}\xi = \int_{-1}^{1} \mathbf{u}\,\mathrm{d}\xi = \sum_{k=0}^{N} \int_{-1+k\cdot w}^{-1+(k+1)\cdot w} \mathbf{u}\,\mathrm{d}\xi.$$

Inserting the DOFs of the FV scheme and numerical integrating over each FV sub-cell individually with the midpoint rule then gives

$$\int_E \mathbf{u}\,\mathrm{d}\xi = \sum_{k=0}^{N} \int_{-1+k\cdot w}^{-1+(k+1)\cdot w} \mathbf{u}\,\mathrm{d}\xi \overset{!}{=} \sum_{k=0}^{N} w\hat{\mathbf{u}}_k. \tag{5.5}$$

To ensure that this approximation with FV sub-cells has the same integral mean value as the Discontinuous Galerkin method, the DG approximation is inserted into the sub-intervals on the left hand side of equation (5.5)

$$\sum_{k=0}^{N} \int_{-1+k\cdot w}^{-1+(k+1)\cdot w} \sum_{i=0}^{N} \hat{\mathbf{u}}_i \psi_i(\xi)\,\mathrm{d}\xi \overset{!}{=} \sum_{k=0}^{N} w\hat{\mathbf{u}}_k. \tag{5.6}$$

The last step is to resolve the integrals over the sub-intervals on the left hand side by numerical integration. Since the DG approximation is a polynomial of degree N, a Gauss integration with $N+1$ points on the sub-interval is exact. The respective Gauss integration points are defined by linearly mapping the original Gauss points $\{\xi_i\}_i$ on the reference interval $[-1,1]$ to every sub-cell

$$\left\{\xi_i^k\right\}_{i=0}^{N} = \left\{-1 + k\cdot w + (\xi_i + 1)\cdot \frac{w}{2}\right\}_{i=0}^{N} \quad \forall k = 0,\dots,N,$$

where $\frac{w}{2}$ is the scaling factor between the reference interval $[-1,1]$ and the sub-intervals, which all have the length w. Integrating numerically the sub-interval in equation (5.6) with these sets of Gauss points one obtains

$$\frac{w}{2}\sum_{k=0}^{N}\sum_{\lambda=0}^{N}\sum_{i=0}^{N} \hat{\mathbf{u}}_i \psi_i(\xi_\lambda^k)\omega_\lambda \overset{!}{=} \sum_{k=0}^{N} w\hat{\mathbf{u}}_k,$$

where ω_λ are the Gauss integration weights. Rearranging this equation leads to

$$\sum_{k=0}^{N} w\left[\sum_{i=0}^{N}\left\{\frac{1}{2}\sum_{\lambda=0}^{N}\omega_\lambda\psi_i(\xi_\lambda^k)\right\}\hat{\mathbf{u}}_i\right] \overset{!}{=} \sum_{k=0}^{N} w\hat{\mathbf{u}}_k.$$

Fulfilling this equation directly ensures discrete conservation and therefore this equation is used to define the conversion from the DOFs of the Discontinuous Galerkin method to the DOFs of the Finite Volume sub-cells method

$$\hat{\mathbf{u}}_k := \left[\sum_{i=0}^{N}\left\{\frac{1}{2}\sum_{\lambda=0}^{N}\omega_\lambda\psi_i(\xi_\lambda^k)\right\}\hat{\mathbf{u}}_i\right] \quad \forall k = 0,\dots,N. \tag{5.7}$$

The curly bracket in this definition can be interpreted as matrix where the k-th row defines the conversion to the DOF of the k-th sub-cell and is given by

$$Vdm_{FV} := \left\{ \frac{1}{2} \sum_{\lambda=0}^{N} \omega_\lambda \psi_i(\xi_\lambda^k) \right\}_{k,i=0}^{N} . \tag{5.8}$$

With this Vandermonde matrix, the conversion from Discontinuous Galerkin degrees of freedom to Finite Volume reduces to a simple matrix-vector multiplication in one space dimension. For higher space dimensions this operation can be performed dimension by dimension, resulting in multiple matrix-vector multiplications. Nevertheless, the matrix is always fix and must only be precomputed once for the polynomial degree N of the DG method before the simulation. In the following matrices that interpolate or project solutions from one basis to another are called Vandermonde matrices.

Multiplying equation (5.7) with the inverse $Vdm_{FV}{}^{-1}$ of the transformation matrix directly gives the definition of the switching from a Finite Volume solution back to a DG representation. In contrast to the Gaussian distributed sub-cells, the DOFs for both methods are not the same and switching a DG element to FV requires the application of Vdm_{FV}. These switching operations introduce on the one hand additional computational tasks, but on the other hand the distribution of Finite Volume sub-cells is uniformly. This overall regular pattern has advantageous properties which are exemplarily shown in figure 5.3. Here a closeup view of the curved shock front of the forward facing step example is presented for two computations where only the distribution of the FV sub-cells is different. The irregular pattern of the grid in the case of Gaussian distributed FV sub-cells leads to small disturbances of the density in this case while the equidistant FV sub-cells show a much smoother profile. This effect is related to the overall mesh resolution and more or less vanishes for highly resolved simulations.

After deriving the degrees of freedom of the FV sub-cells and the respective conversion matrices from Discontinuous Galerkin DOFs to Finite Volume DOFs and vice versa in one space dimension, the following explanations are again for the two dimensional case.

5.1.2. Approximation of the Finite Volume formulation

The Finite Volume discretization is now inserted into the Finite Volume formulation from equation (5.1) and the integrals of this equation are approximated numerically. First, the time derivative integral is approximated. Since the integral mean value $\hat{\mathbf{u}}_{ij}$ inside the Finite Volume sub-cell κ_{ij} is constant, the numerical

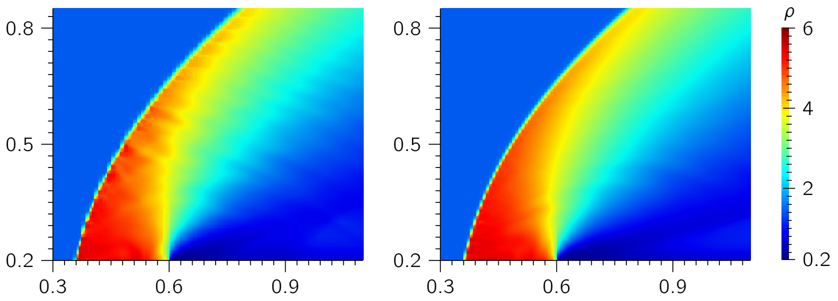


Figure 5.3.: Closeup view of curved shock front of the forward facing step example. The uneven distribution of the FV sub-cells for the Gaussian case (left) introduces small disturbances into the solution. The density for the equidistant FV sub-cells on the right is much smoother.

integration reduces to

$$\int_{\kappa_{ij}} J\mathbf{u}_t \, \mathrm{d}\xi = \omega_i \omega_j J_{ij} \frac{\partial \hat{\mathbf{u}}_{ij}}{\partial t},$$

where J_{ij} is the integral mean value of the original Jacobian determinant J inside the sub-cell κ_{ij}. Here, the more general case of the Gaussian distributed FV sub-cells is used. To obtain the corresponding equations for the equidistant case, one has to replace each of the widths ω_i and ω_j of the Gaussian distributed sub-cell with the equidistant width w. All other parts remain the same.

The boundary integral of equation (5.1) is separated into the four edges of the sub-cell. Integration along these edges is performed numerically with the midpoint rule, resulting in

$$\begin{aligned}\int_{\partial\kappa_{ij}} \mathcal{F}(\mathbf{u}) \cdot \mathcal{N} \, \mathrm{d}S_\xi = \quad & \omega_j \left(f_{i-\frac{1}{2},j}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) + f_{i+\frac{1}{2},j}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) \right) \\ & + \omega_i \left(f_{i,j-\frac{1}{2}}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) + f_{i,j+\frac{1}{2}}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) \right),\end{aligned}$$

where e.g. $f\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right)_{i-\frac{1}{2},j}$ is the flux at the left edge of the ij-th sub-cell. Putting the last two equations together, the Finite Volume sub-cells method reads

$$\begin{aligned}\omega_i \omega_j J_{ij} \frac{\partial \hat{\mathbf{u}}_{ij}}{\partial t} = & -\omega_j \left[f_{i-\frac{1}{2},j}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) + f_{i+\frac{1}{2},j}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) \right] \\ & -\omega_i \left[f_{i,j-\frac{1}{2}}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) + f_{i,j+\frac{1}{2}}\left(\mathbf{u}^-, \mathbf{u}^+, \mathcal{N}\right) \right].\end{aligned} \tag{5.9}$$

From this equation the extension to three space dimensions is easy to see. On the left hand side one has to add ω_k and add a third index k to ij. Besides doing this for the right hand side as well, one has to insert a third square bracket term for $k \pm \frac{1}{2}$. Keep in mind that for the equidistant case all ω_i, ω_j and ω_k have to be replaced by w.

In both cases, equidistant and Gaussian distributed FV sub-cells, the time derivative of the DOFs is integrated in time using the same Runge-Kutta time integration as for the Discontinuous Galerkin method.

5.2. Time step restriction

In section 4.2.1 the time step restriction of the Discontinuous Galerkin method for the Runge-Kutta time integration was derived. In this section, the influence of the shock capturing with Finite Volume sub-cells on the time step restriction is discussed. For other shock capturing techniques, for example the artificial viscosity approach, it is well known that they can tremendously decrease the overall time step [1]. Hence, a strict analysis for the here presented scheme is essential. For the Discontinuous Galerkin Spectral Element Method with an explicit Runge-Kutta time integration, the time step restriction was already given in equation (4.12) and is repeated here

$$\Delta t \leq \mathrm{CFL} \cdot \frac{\alpha_{RK}(N)}{2N+1} \frac{\Delta x_{DG}}{\lambda^c}.$$

This time step restriction is derived from the CFL condition. The same can be done for the time step restriction of the Finite Volume method which then reads

$$\Delta t \leq CFL \cdot \alpha_{RK}(0) \frac{\Delta x_{FV}}{\lambda^c}, \tag{5.10}$$

where Δx_{FV} is the size of the FV sub-cells. The factor $\alpha_{RK}(0)$ has to be inserted for the explicit Runge-Kutta scheme additionally, and in contrast to the DG method it does not depend on the polynomial degree, but in fact is the value for the DG scheme with a polynomial degree of $N = 0$.

To make both time step restrictions comparable to each other, the element size of the (smallest) FV sub-cell is written as fraction of the DG element size. For the two different distributions of the FV sub-cells this reads

$$\Delta x_{FV} = \frac{\Delta x_{DG}}{N+1} \quad \text{and} \quad \Delta x_{FV} = \frac{\min(\omega)}{2} \Delta x_{DG}$$

for the equidistant and the Gaussian case, respectively, where $\min(\omega)$ is the minimal weight of the Gauss integration rule. The extra division by 2 is needed since

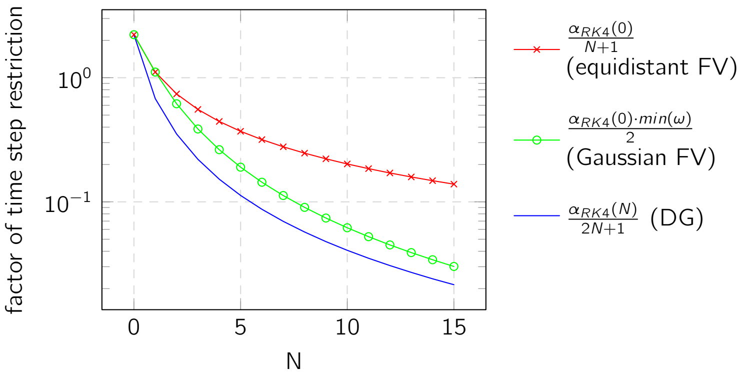


Figure 5.4.: Factors in the time step restriction for the FV sub-cells method and the DG scheme. For all polynomial degrees and both distributions of the FV sub-cells, the time step is bigger than the one for the DG method.

the Gauss weights sum up to $\sum_{i=0}^{N}\omega_i = 2$. Substituting this into the time step restriction of the FV method in equation (5.10) this condition becomes

$$\Delta t \leq CFL \cdot \frac{\alpha_{RK}(0)}{N+1}\frac{\Delta x_{DG}}{\lambda} \quad \text{or} \quad \Delta t \leq CFL \cdot \alpha_{RK}(0)\frac{\min(\omega)}{2}\frac{\Delta x_{DG}}{\lambda}.$$

Comparing them to the time step restriction of the DG method in equation (4.12), the only differences are the following factors

DG	equidistant FV	Gaussian FV
$\dfrac{\alpha_{RK}(N)}{2N+1}$	$\dfrac{\alpha_{RK}(0)}{N+1}$	$\alpha_{RK}(0)\dfrac{\min(\omega)}{2}$.

In figure 5.4, these factors are plotted over the polynomial degree for a fourth order explicit Runge-Kutta time integration.

In all cases the factor of the FV sub-cells method is greater than the factor of the DG scheme. Hence, the shock capturing with the Finite Volume sub-cells does not diminish the time step, but indeed can lead to bigger time steps if the highest signal velocities λ^c are only present in FV sub-cell regions. For equidistant sub-cells one might consider to increase the number of FV sub-cells to $2N+1$ in each direction as this would still give larger time steps than for the DG method. Dumbser et al. use this number of sub-cells in [24], but this has the disadvantage of a completely changed data structure.

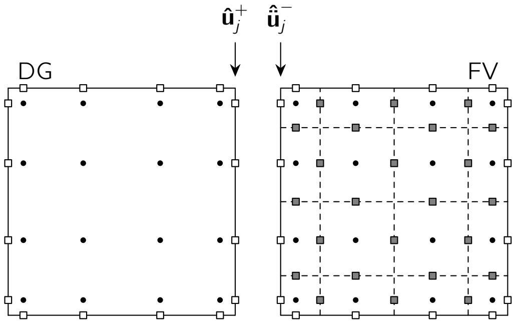


Figure 5.5.: Coupling of a Discontinuous Galerkin element with Gaussian distributed Finite Volume sub-cells.

5.3. Coupling of Discontinuous Galerkin elements and Finite Volume sub-cells

The Finite Volume method is dedicated to resolve shocks or high gradients in the flow solution and should therefore only be applied in such regions. The troubled zones are detected using indicators which will be explained in section 5.4. In all other elements, the smooth parts of the domain, the Discontinuous Galerkin method should be applied to benefit from the high order of this method. Therefore in general, some parts of the domain are discretized with DG others with FV sub-cells. Since both methods use numerical fluxes to couple elements, it is natural to use them to couple regions with the two different methods.

The two different discretizations of the Finite Volume method in equidistant and Gaussian distributed sub-cells need different couplings to DG elements. First, the Gaussian distributed sub-cells are considered. In figure 5.5 a DG element and an adjacent FV sub-cells element are plotted with a small gap between them, even so they are connected through a single face. In this case, the flux points of two adjacent elements are both distributed like the Gauss points of the interpolation/integration and therefore match. The flux computed with left and right states $\{\hat{\mathbf{u}}_j^+\}_j$ and $\{\hat{\hat{\mathbf{u}}}_j^-\}_j$ can directly be used to update the DG element on the left hand side and the FV sub-cells next to the interface on the right hand side.

In the case of an equidistant FV sub-cells element, the coupling with a DG element is shown in figure 5.6. Here, the left and right states at the interface are distributed differently. The left states $\{\hat{\mathbf{u}}_j^+\}_j$ are distributed like the Gauss integration points along the interfaces while the right states $\{\hat{\hat{\mathbf{u}}}_j^-\}_j$ are spread uniformly.

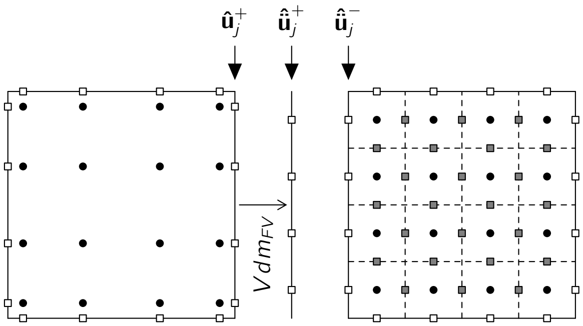


Figure 5.6.: Coupling of a Discontinuous Galerkin element with equidistant distributed Finite Volume sub-cells.

Since they do not match, the DG side has to be converted to match with the FV side. This is done with the same Vandermonde matrix given in equation (5.8), which is used to convert Discontinuous Galerkin DOFs to Finite Volume DOFs. The only difference is that here it is applied to face data only and not to the whole volume data. After this operation, the left and right states are given at the same positions and the numerical flux can be evaluated using the Riemann solver. Since these fluxes are computed at equidistant points, they can only be used on the Finite Volume side directly to update the sub-cells next to the interface. For the Discontinuous Galerkin side of the interface, the fluxes must first be projected to Gaussian distributed points. This is done using the inverse operator Vdm_{FV}^{-1} of the conversion of the state from DG to FV. All these steps written in pseudo code read

```
$\hat{\mathbf{u}}^+ := Vdm_{FV} \cdot \hat{\mathbf{u}}^+$    (Interpolate DG solution to equidistant FV points)
  $\hat{\hat{f}} := \text{Riemann solver}\left(\hat{\mathbf{u}}^-, \hat{\mathbf{u}}^+, \mathcal{N}\right)$    (Calculate numerical flux in FV points)
  $\hat{f} := Vdm_{FV}^{-1} \cdot \hat{\hat{f}}$    (Project flux from FV to DG points).
```

Therewith, all basic ingredients for a Discontinuous Galerkin shock capturing using first order Finite Volume sub-cells are described. The only remaining component to build a numerical scheme is the detection of the troubled DG elements that should be treated with the FV sub-cells.

5.4. Indicators

The Finite Volume sub-cells method is designed to be integrated into the high order Discontinuous Galerkin algorithm with the aim to handle shocks or instabilities that will produce non-physical solutions and finally blow up the high order computation. This requires the knowledge when and where the DG method reaches its limits and the FV sub-cells should take over the work. Basically, there are two different approaches to detect the troubled elements: A posteriori and a priori. The latter approach is used in [24] which is based on the MOOD approach [14]. In general, the idea is to compute each time step with the high order method for all elements and mark the troubled cells, i.e. with a non-physical solution, afterwards repeat this time step, but now update the marked cells with a low order scheme. It is clear that this approach introduces a large additional amount of extra work and requires a dynamic load balancing to resolve load imbalances in a parallel setting. Therefore, in this work a posteriori indicators are used which mark all elements that may become problematic after each time step. On the one hand this might lead to unnecessarily marked elements, but on the other hand it does not require a recomputation of every single time step.

The indicators used in this work are described in the following. All these functions are modified in such a way that they return a low value in smooth regions of the solution and indicate a shock with a high return value. The general idea is to specify a threshold value which decides where to use the high order DG method and where to used the shock capturing FV sub-cells. If the indicator value for a Discontinuous Galerkin element becomes greater than the threshold, the solution is converted to a Finite Volume sub-cells element. Or the other way round, if the indicator value falls below the threshold, a FV sub-cells element becomes a DG element again. Numerical studies showed that a single threshold value might introduce numerical problems which occur if the indicator value is near the threshold. In this case, after switching from DG to FV sub-cell and performing one time step, the new indicator value may be on the opposite side of the threshold again and directly lead to a switching back of the solution to the beforehand type. This could introduce an alternating switching back and forth between the two numerical methods which might introduce spurious artifacts. Therefore, the single threshold is replaced by an upper and a lower threshold as visualized by the range in figure 5.7. If the indicator value of a DG element becomes greater then the upper threshold the solution of this element is switched to the FV sub-cells method. The element keeps updating with the FV scheme until the indicator value falls below the lower threshold. The range between the lower and upper threshold prevents a permanent switching between the two methods.

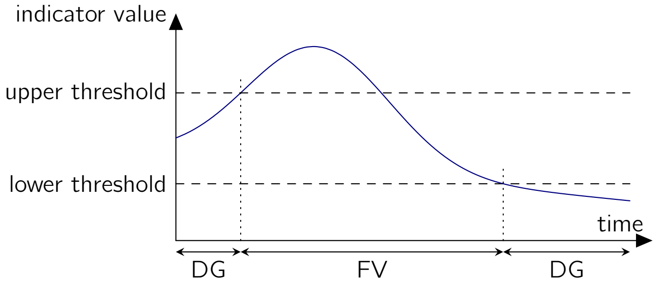


Figure 5.7.: To prevent a permanent switching between the DG and the FV subcells method two thresholds are used. The solution switches from DG to FV if the indicator value becomes greater then the upper threshold and is only switched back to DG if the indicator value falls below the lower threshold.

5.4.1. Persson indicator

In [64] Persson and Peraire developed a shock capturing for DG methods where artificial viscosity should vanish in smooth regions. Therefore, they proposed a sensor which is based on a representation of the solution in terms of a hierarchical family of orthogonal polynomials. The basic idea is that for a smooth solution the coefficients or modes decay very quickly. Since the approximation of a discontinuity with polynomials leads to oscillations, they propose to take the amount of the highest mode as a smoothness indicator. The indicator compares the amount of solution represented only in the highest polynomial mode with the whole solution up to this mode. Per element this indicator is defined as

$$S = \frac{(U - \widetilde{U}, U - \widetilde{U})_{L_2}}{(U, U)_{L_2}},$$

where $(\cdot, \cdot)_{L_2}$ is the standard L_2 inner product of an element and the numerical solution U is given in a modal hierarchical basis of polynomials up to degree p. The truncated expansion of this solution is $\widetilde{U}$ and these expansions can be expressed by

$$U = \sum_{i=1}^{N(p)} u_i \Psi_i \quad \text{and} \quad \widetilde{U} = \sum_{i=1}^{N(p-1)} u_i \Psi_i,$$

where $N(p)$ denotes the total number of polynomial basis functions Ψ up to the polynomial degree p. Defining the truncation to modes between a and b

$$[\widetilde{U}]_a^b := \sum_{i=N(a-1)+1}^{N(b)} u_i \Psi_i$$

the indicator can be rewritten to

$$S = \frac{([\widetilde{U}]_p^p, [\widetilde{U}]_p^p)_{L_2}}{([\widetilde{U}]_1^p, [\widetilde{U}]_1^p)_{L_2}}.$$

Taking not only the amount of information in the highest mode into account but also comparing the second highest modes and so on, the indicator can be generalized to

$$\mathcal{I}_{\text{Persson}} = \max\left\{ \frac{([\widetilde{U}]_i^i, [\widetilde{U}]_i^i)_{L_2}}{([\widetilde{U}]_1^i, [\widetilde{U}]_1^i)_{L_2}}; i = p-k, \dots, p \right\},$$

which takes the maximal indicator value for all polynomial degrees of the k highest modes. For the numerical examples in chapter 6, the Persson indicator refers to this generalization with $k = 2$.

5.4.2. JST indicator

This indicator originates from the switching function of the Jameson-Schmidt-Turkel scheme [45]. A three dimensional adaption to ijk-th Finite Volume sub-cells or to the ijk-th DOFs of a Discontinuous Galerkin element reads

$$\mathcal{I}_{\text{JST}}(i,j,k) = \frac{p_{min,ijk} - 2p_{ijk} + p_{max,ijk}}{p_{min,ijk} + 2p_{ijk} + p_{max,ijk}} \tag{5.11}$$

where $p_{min,ijk} = \min(p_{i\pm\delta_{id},j\pm\delta_{dj},k\pm\delta_{dk}}, d = 1,2,3)$ is the minimal pressure of the neighboring nodal values of the node ξ_{ijk} and $p_{max,ijk}$ the respective maximal value. Equation (5.11) gives an indicator value for each FV sub-cell or DG degree of freedom, but a single indicator value for the whole DG element is needed. Using the volume weighted mean value of all nodal values, the single indicator value of an element reads

$$\mathcal{I}_{\text{JST}} = \frac{1}{V_{Element}} \sum_{i,j,k=0}^{N} \mathcal{I}_{\text{JST}}(i,j,k) V_{ijk},$$

where $V_{Element}$ denotes the volume of the whole DG element and V_{ijk} is the volume of the ijk-th FV sub-cell. The JST indicator can of course be evaluated for other variables instead of the pressure, for example the density is also used for the Navier–Stokes equations. In literature, the JST indicator is often called Jameson indicator.

5.4.3. Ducros indicator

The JST indicator detects shocks by looking on the surrounding pressure differences of a point. In turbulent flow this may lead to wrong detection of turbulent structures. To prevent the JST indicator from this misbehavior, Ducros et al. [23] developed an additional sensor

$$\mathcal{I}_{\text{Ducros}} = \frac{(\nabla \cdot v)^2}{(\nabla \cdot v)^2 + (\nabla \times v)^2 + \varepsilon},$$

where $\nabla \cdot v$ is the divergence of the velocity vector field, $\nabla \times v$ the vorticity and $\varepsilon = 1^{-15}$ a small constant that prohibits division by zero. This sensor is zero in weakly compressible regions and goes to 1 near shocks. It can therefore be used directly as a shock indicator or, as Ducros et al. suggested, be multiplied to the JST indicator to exclude triggering of turbulence from the JST indicator.

5.5. Second order reconstruction

The major drawback of the Finite Volume method in contrast to the Discontinuous Galerkin method is the order of accuracy. While the DG method can easily be built for any polynomial degree (in theory, numerically there are some restrictions due to the floating point arithmetic) the Finite Volume method, as derived above, is only a first order method. Due to this poor accuracy, the results of this method include numerical errors that might not be negligible. Especially the quite large numerical dissipation smoothens the solution and destroys small flow features.

Therefore, the Finite Volume method should only be applied to regions where it is unavoidable and this method has it strengths. This is the case in regions where shocks or high gradients occur and where the total variation diminishing (TVD) property guarantees stability by not introducing new extrema. The guaranteed stability follows from the proof of Harten [34] that a total variation diminishing scheme is monotonicity preserving. Since the overall stability of the scheme is of major importance, any enhancement of the accuracy of the Finite Volume scheme must retain the TVD property.

Nevertheless, the first order Finite Volume method can be extended to a second order scheme without losing the TVD property. In the 1970's van Leer introduced

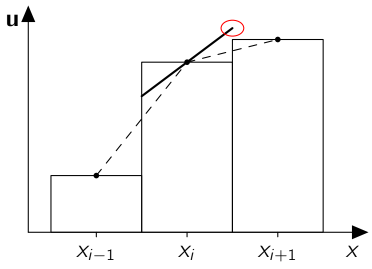


Figure 5.8.: Reconstruction of a linear polynomial using the slopes from cell center to the cell centers of the adjacent cells (dashed lines). Using the arithmetic mean value of these slopes as slope of the reconstructed solution, generates a new maximum in this case. This violates the total variation diminishing property of the scheme which may cause instabilities.

in a series of papers [58, 54, 55, 56, 57] slope limiters to build higher order methods. The general concept is to approximate the solution inside a Finite Volume cell as a linear polynomial instead of constant cell averages. The integral mean values of the neighboring cells are used to reconstruct a linear solution within each cell. Figure 5.8 shows a cell and its two neighboring cells in one space dimension. The slopes from cell center to cell center of the adjacent elements are plotted as dashed lines. Using the arithmetic mean of the two slopes as slope of the linear reconstruction, generates in this case a solution that introduces a new maximum. Therewith, the solution is not guaranteed to be stable, since the new maximum violates the total variation diminishing property of the scheme.

Remark 5.3 (Reconstruction in primitive quantities)**.** *The reconstruction of slopes can be performed directly on the numerical solution given in conservative quantities. Due to the nonlinearity of the conversion between primitive and conservative quantities this has the drawback that even so the conservative quantities do not generate new extrema, primitive quantities, for example the pressure, can drop below zero. This negative pressure then enters the numerical flux computation by the Riemann solver which will produce invalid results. Therefore, the limiting can instead be performed on primitive quantities. For the numerical results shown in chapter 6 density, velocity, pressure and temperature are limited and reconstructed.*

5.5.1. Slope limiters

To ensure the total variation diminishing property of a second order Finite Volume scheme, slope limiters are used. They prevent the generation of new extrema as follows. Let x_{i-1}, x_i and x_{i+1} be the cell centers of three cells in one space dimension as shown in figure 5.8. The solution at these points are given by $\mathbf{u}_{i-1}, \mathbf{u}_i, \mathbf{u}_{i+1}$ and $\Delta x_{i\pm 1}$ denote the distances between the cell centers. The slopes of the linear reconstruction to the left and right of the i-th cell are then given by

$$s_{i-1} = \frac{\mathbf{u}_i - \mathbf{u}_{i-1}}{\Delta x_{i-1}} \qquad \text{and} \qquad s_{i+1} = \frac{\mathbf{u}_{i+1} - \mathbf{u}_i}{\Delta x_i}.$$

The ratio of this successive slopes

$$r_i = \frac{\frac{\mathbf{u}_i - \mathbf{u}_{i-1}}{\Delta x_{i-1}}}{\frac{\mathbf{u}_{i+1} - \mathbf{u}_i}{\Delta x_{i+1}}} \tag{5.12}$$

is plugged into a limiter function $\Lambda(r_i)$ which is then used to limit the right slope s_i in such a way that no new extrema occur. In general, the slope of the i-th cell is computed by

$$s_i = \Lambda(r_i) s_{i+1}.$$

Remark 5.4. *On an equidistant mesh the distances between the cell centers are the same and the ratio of the slopes reduces to* $r_i = \frac{\mathbf{u}_i - \mathbf{u}_{i-1}}{\mathbf{u}_{i+1} - \mathbf{u}_i}$*, which is more often used in literature to define limiter functions. Since here the mesh is arbitrary the general definition from equation* (5.12) *is used.*

In figure 5.9 the desired region for the slope of a limited FV cell is visualized. The following description is for the limitation of the i-th cell in figure 5.9. The three cells at x_{i-1}, x_i and x_{i+1} are not of the same size here to demonstrate the general case. With $\Delta x_{i-\frac{1}{2}}$ and $\Delta x_{i+\frac{1}{2}}$, the distances from the cell center of the i-th cell to its left and right boundary are denoted. Outgoing from the center of the middle cell two different slopes to both neighboring cells are plotted. The red slopes mark the maximal slopes that are allowed such that the slopes do not generate new extrema on the boundary of the respective adjacent cell. The limiter is prohibited to generate a slope that is steeper than any of the two involved red lines. In contrast to that the green slopes, which are used to calculate the ratio r_i, only mark a bound that should at least be reached to reconstruct a linear solution. As a consequence of this, limiters fulfill

$$\Lambda(1) = 1$$

on a locally linear solution, which is the case when both slopes to the left and the right are the same and the ratio r_i becomes one. Due to the restriction of the

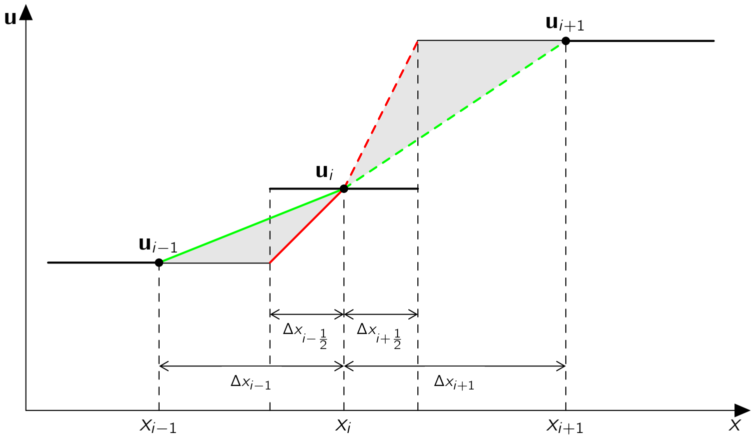


Figure 5.9.: Admissible region of the slopes to the left and right adjacent cell. The reconstructed slope must not be steeper than any of the red lines, but it is allowed to be flatter than the green lines.

upper bound (red), every limiter restricts the slope to zero at (local) minima or maxima. This is the case when the ratio r_i becomes negative as the signs of the associated slopes differ

$$\Lambda(r) = 0 \qquad \forall r \leq 0.$$

Following this explanation, the most simple limiter is the MinMod limiter [70] which takes always the left or right slope with the smaller absolute value and zero in case of different signs of these slopes

$$\Lambda_{MinMod}(r) = \max(0, \min(1, r)).$$

Due to its simplicity and great numerical stability, the MinMod limiter is widely used for computations involving shocks.

Another very common limiter is the Sweby limiter [88] which includes a parameter $\beta \in [1, 2]$. On an equidistant mesh, the limiter is given by

$$\Lambda(r) = \max(0, \min(\beta r, 1), \min(r, \beta)). \tag{5.13}$$

The parameter allows to vary the steepness of the limited slope. For $\beta = 1$ the Sweby limiter coincides with the MinMod limiter and for $\beta = 2$ it is the Superbee limiter [69]. To transform this limiter to non-equidistant meshes one has to understand the single parts of equation (5.13). If the ratio r is negative, both minimum functions are negative and the maximum function will select the zero. In figure 5.9, the red lines indicate the maximal allowed slopes to the right and left of the i-th cell such that it remains total variation diminishing to the neighboring cell. On an equidistant mesh the red slopes are twice as much as the respective green ones and therefore the β parameter of the Sweby limiter is bounded by 2. For a β between 1 and 2 the Sweby limiter blends between the MinMod and the Superbee limiter by reducing the red slopes to only $1.x$ times the green slopes. The first minimum function of equation (5.13) chooses the smaller of the red solid and the dashed green slope in figure 5.9. For $\min(\beta r, 1) = 1$ it takes the green dashed slope and for $\min(\beta r, 1) = \beta r$ the red solid one. Therefore, it is clear that a modification of equation (5.13) for non-equidistant meshes must not touch the 1 and only has to rescale the β in the first minimum function for different cell sizes. On an equidistant mesh, $\beta = 2$ is just the ratio

$$\frac{\Delta x_{i-1}}{\Delta x_{i-\frac{1}{2}}} = 2.$$

Therefore, one has to rescale this β with $\frac{\Delta x_{i-1}}{2\Delta x_{i-\frac{1}{2}}}$ for arbitrary cell sizes. The second minimum function in equation (5.13) chooses between the green solid and the red dashed line of figure 5.9. By the same reasoning only the β has to be rescaled and the limiter function of the Sweby limiter on non-equidistant meshes becomes

$$\Lambda_{Sweby}(r) = \max\left(0, \min\left(\beta r \frac{\Delta x_{i-1}}{2\Delta x_{i-\frac{1}{2}}}, 1\right), \min\left(r, \beta \frac{\Delta x_{i+1}}{2\Delta x_{i+\frac{1}{2}}}\right)\right).$$

5.5.2. Reconstruction on curved meshes

In multiple space dimensions, a second order reconstruction requires the computation of slopes in every space direction. The basic approach is to build the slopes to the neighboring elements in each space direction. This becomes even more complicated if the mesh is stretched or curved. In contrast to the formulation of the Finite Volume scheme on the reference element, the calculation of the slopes must be done in physical space. This is necessary due to the slope limiting, where the two slopes entering the calculation must be of the same scale. Otherwise, a comparison like in the MinMod-limiter, which chooses the smaller slope, does not make sense. Figure 5.10 illustrates the influence of the mapping of the equidistant sub-cells of one element into physical space on the distances between FV sub-cell centers. In this example lines in ξ-direction of the reference space are mapped to spherical lines in physical space that are additionally stretched along

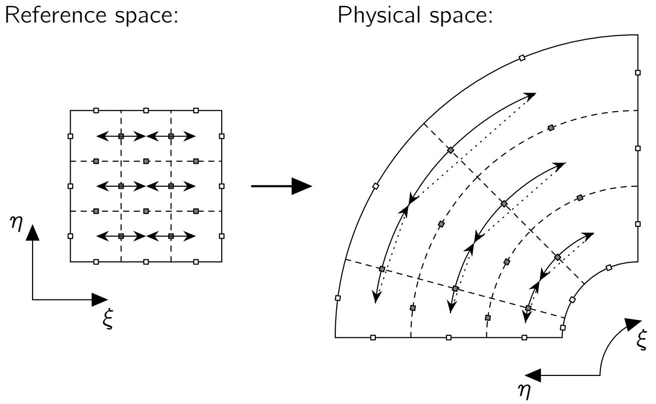


Figure 5.10.: A nonlinear mapping, e.g. due to a stretching, of a reference FV sub-cells element to the physical space changes the ratio of lengths and distances. Therefore, the limiting of slopes for the second order reconstruction has to be built on the physical distances instead of reference distances.

this direction. Due to the stretching, even so the reference element is divided into equidistant sub-cells, the distances between two neighboring cell centers and the distances from the cell centers to the intermediate faces are not the same.

All these required distances and lengths can be computed once in a preprocessing step. Since the FV sub-cell algorithm inherits the tensor product structure of the underlying DG method, the calculation of the distances can also be done dimension by dimension. In the example of figure 5.10 the distances between the cell centers in ξ-direction are visualized. These distances can be computed for each index j of the η-direction by integrating along the ξ-direction. Let

$$\begin{aligned} X_\eta(\xi): \quad & L \to C \\ & [-1,1] \mapsto X(\xi,\eta) \end{aligned}$$

be the mapping that maps the line $L(\xi) = \{(\xi,\eta) | \xi \in [-1,1], \eta = \text{fix}\}$ in ξ-direction of the reference element to a curve C in the physical space. Using definition 2.2, the physical length of the curve C can be computed with

$$\int_C 1 \, \mathrm{d}s = \int_{-1}^{1} 1 \left\| \frac{\partial X_\eta(\xi)}{\partial \xi} \right\|_2 \mathrm{d}\xi.$$

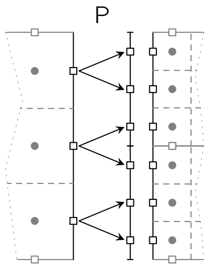


Figure 5.11.: Interpolation from the big side to two small virtual sides. The values are just copied from a big side FV sub-cell to the two adjacent small sub-cells.

The same can be done in any other coordinate direction of the reference space. Of course integrating piecewise only parts of these lines is possible as well. Therewith, all distances and lengths can be computed from the Jacobian matrix of the mapping X, which maps the reference space to the physical space by simply integrating the norm of the covariant basis vectors from equation (2.4) along one-dimensional lines in the reference space.

5.6. Mortar interfaces

In section 4.4, mortar interfaces have been introduced into the Discontinuous Galerkin method to handle nonconforming meshes. To be able to apply the Finite Volume sub-cells method in every element that needs a shock capturing, regardless if it is placed at a mortar interface or not, it is necessary to define mortar interfaces for FV sub-cell interfaces as well. Additionally, mixed mortar interfaces, where on one side of the interface a FV sub-cells element and on the other DG elements are located or vice versa, must be handled. Even quite complicated settings are imaginable, where on the small elements side of a mortar interface different types, DG of FV sub-cells, may coexist. In all cases the general strategy is the same as for pure DG mortar interfaces from section 4.4. The solution from the big side is interpolated to small virtual sides and on these small sides the flux is evaluated. Afterwards, the small side fluxes are projection back to the big side. In the following, mortar interfaces for pure FV/FV interfaces are introduced first and thereafter all possible mixed interfaces are described.

Following the approach for DG mortar interfaces, one must define only one-dimensional interpolation and projection operators which are applied for the mortar types 1 and 2 in the respective coordinate direction, or for type 3 are concatenated in both directions. Since everything takes place in the reference space, there are two interpolation operators which interpolate the solution $\mathbf{u}^B$ from the reference interval $[-1, 1]$ either to the solution $\mathbf{u}^L$ on the left half $[-1, 0]$ or to $\mathbf{u}^R$ on the right interval $[0, 1]$ of the big element. The mean value inside a Finite Volume solution is just copied to the adjacent two small sub-cells as shown in figure 5.11 and can be written like the DG case in equation (4.14) as matrix-vector product

$$\mathbf{u}^L = I^L_{FV} \cdot \mathbf{u}^B, \qquad \mathbf{u}^R = I^R_{FV} \cdot \mathbf{u}^B.$$

The corresponding interpolation matrices I^L_{FV} and I^R_{FV} can be defined by taking the unit matrix and doubling each line. The upper half of this $(2N \times N)$-matrix is the interpolation matrix to the left interval $[-1, 0]$ and the lower half to the right interval $[0, 1]$. For the example, in figure 5.11 the matrices are given by

$$I^L_{FV} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad I^R_{FV} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.14}$$

In three space dimension the interpolation operators are applied to the appropriate dimensions of the three different mortar types. A detailed view on the three dimensional case, including the three different mortar types, is already covered in the Discontinuous Galerkin framework in section 4.4. The only difference between DG and FV sub-cells mortar interfaces are the interpolation and projection matrices that are applied. On the small (virtual) sides, the flux is computed in the general conform framework as before. These fluxes can be directly added to the FV sub-cells of the small elements, but must be projected back to the big side.

The projection of the small side fluxes f^L and f^R to the big side is quite simple. For one big FV sub-cell, the flux over the mortar interface is just the sum of the two fluxes over the matching small sides. Thereby, the projection becomes a matrix-vector multiplication as in equation (4.16)

$$f^B = P^L_{FV} \cdot f^L + P^R_{FV} \cdot f^R.$$

The projection matrices P^L_{FV} and P^R_{FV} are just the transposed interpolation matrices. Altogether, the mortar interfaces for FV sub-cells elements are treated the same way as DG mortar interfaces with the only difference in the used matrices. The last open question is the procedure at mixed DG and FV sub-cells mortar interfaces.

5.6.1. Mixed DG and FV sub-cells mortar interfaces

The basic strategy at mortar interfaces is to interpolate the solution to small virtual sides and then compute the flux only at conform interfaces. At this level, the coupling of DG and FV sub-cells is already given in section 5.3 and can be summarized as follows

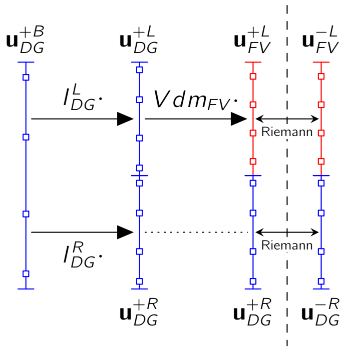
$$\begin{aligned} \mathbf{u}^+_{FV} &:= Vdm_{FV} \cdot \mathbf{u}^+_{DG} \qquad \text{(Interpolate DG solution to equidistant FV points)} \\ f_{FV} &:= \text{Riemann solver}(\mathbf{u}^-_{FV}, \mathbf{u}^+_{FV}, \mathcal{N}) \qquad \text{(Calculate numerical flux in FV points)} \\ f_{DG} &:= {Vdm_{FV}}^{-1} \cdot f_{FV} \qquad \text{(Project flux from FV to DG points),} \end{aligned}$$

where $\mathbf{u}^+_{DG}$ denotes the DG solution on the plus side and $\mathbf{u}^-_{FV}$ the FV solution on the minus side. The solution from the DG side must be converted to a FV sub-cells solution $\mathbf{u}^+_{FV}$, since the flux will be computed in FV interface points. This flux f_{FV} can be added directly to the FV sub-cells element, but for the DG element it has to be projected back to the DG representation f_{DG} first. The approach for coupling adjacent conform DG and FV elements is applied the same way at mortar interfaces.

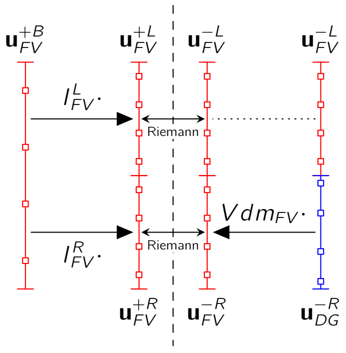
The procedure for all different possible combinations of DG and FV elements at a mortar interface is summarized in figure 5.12. In general, there are two different settings. Either the big side element is a DG element (figure 5.12a), or a FV sub-cells element (figure 5.12b). In both cases the solution is interpolated to the small virtual sides, regardless of the adjacent element types. On the small sides level, the interfaces are conforming and the coupling between DG and FV is done as described in section 5.3. As long as a side of a FV sub-cells element is involved, the flux must be computed in FV points. In this case a DG side solution must be converted to FV points first using the Vandermonde matrix Vdm_{FV}. After computing the fluxes on all small side, they must be projected back to the big side. Therefore, the paths in figure 5.12 are taken backwards. But this time the inverse Vandermonde matrix ${Vdm_{FV}}^{-1}$ and the projection matrices instead of the interpolation matrices are used.

In case of a DG element at the big side, see figure 5.12a, the solution on the big face is interpolated with the operators I^L_{DG} and I^R_{DG} to DG solutions on the small virtual faces. For the left small virtual face, the adjacent element is a FV sub-cells element and therefore the DG solution must be converted to FV using the matrix Vdm_{FV}.

In the other case, the big side element is a FV sub-cells element, see figure 5.12b. The face solution from this element is interpolated to the small virtual faces using the FV interpolation operators I^L_{FV} and I^R_{FV}. For the right small virtual



(a) DG on the big side



(b) FV on the big side

Figure 5.12.: Procedure at mixed DG and FV sub-cell mortar interfaces. Basically, there are two different setting: The big side element is a DG element (a) or it is a FV sub-cells element (b). In each case the first step is to interpolate the big side solution to the small virtual sides with the appropriate operator. The flux computation is performed on conforming small interfaces. If one side is a FV sub-cells side, the flux is computed in FV points. In such a case, a DG side must be interpolated to FV sub-cell points using the Vandermonde matrix Vdm_{FV}.

face in this example, the adjacent element is a DG element and the DG solution must be converted to FV using the Vandermonde Vdm_{FV}.

5.6.2. Edge local reconstruction at mortar interfaces

In the previous section the solution from a big side Finite Volume sub-cell at a mortar interface is just copied to the two adjacent small virtual sides, see figure 5.14a. This shifts the average cell value from the cell center of the big side to the cell centers of the small virtual sides. As long as the solution of the big side element has a non-zero slope along the mortar interface this leads to spurious artifacts due to the different mesh resolutions, see the left plot of figure 5.13. To reduce this problem the big side solution can be reconstructed along the mortar interface to improve its resolution. With the aim to keep the following explanations simple, the derivation is restricted to equidistantly distributed FV sub-cells. The process for Gaussian distributed FV sub-cells follows the same approach.

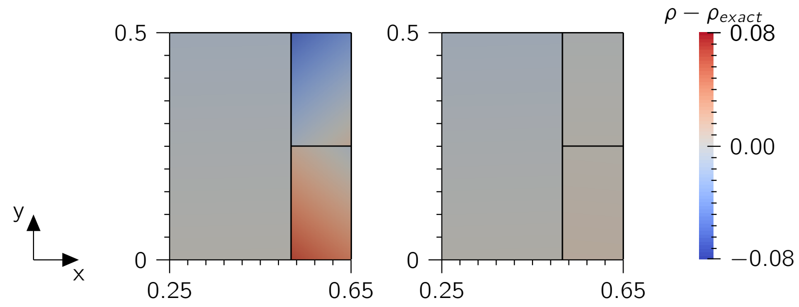


Figure 5.13.: Comparison of Finite Volume mortar interfaces without and with the non-TVD reconstruction. The exact solution is constant in the x- and linear in the y-direction and is advected in x-direction. The difference between the numerical and the exact solution after a single time step, which should be zero, is plotted. Left: without reconstruction along the mortar interface. Right: with a non-TVD reconstruction along the mortar interface.

Instead of only copying the big side solution to the small virtual sides, see figure 5.14a, the big side solution is reconstructed along the mortar interface using the inner slopes of the element. This is explained for the example in figure 5.14b, where for a polynomial degree of $N = 2$ the solution of the three sub-cells at the big side of the mortar interface have the values a, b and c. Assuming the flux points of the big side have a distance of Δy, the slopes along the mortar interface from the first to the second and from the second to the third sub-cell are given by

$$\nabla_{ab} = \frac{b-a}{\Delta y} \quad \text{and} \quad \nabla_{bc} = \frac{c-b}{\Delta y}.$$

These slopes can also be computed by a matrix-vector product of the vector $(a, b, c)^\top$ with the matrix

$$M_\nabla := \frac{1}{\Delta y}\begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \nabla_{ab} \\ \nabla_{bc} \end{pmatrix} = M_\nabla \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix}. \tag{5.15}$$

For a TVD reconstruction one would now take these slopes and also the slopes to the neighboring outer sub-cells of a and c respectively and limit them using for example the MinMod limiter. Not only are the slopes to the neighboring outer sub-cells not uniquely defined for unstructured meshes, this would also lead to a quite complicated communication structure over vertices of the mesh. Therefore, the reconstruction proposed here is element local which leads to a loss of the TVD property along the mortar interface. But in return, it can be condensed into a single element local matrix. For the slopes in the outer sub-cell, there is no choice except taking the slopes to the next inner sub-cell. To keep the overall scheme symmetric, the slopes for inner sub-cells, here only b, are the mean values of the two slopes to the neighboring sub-cells. Denoting with ∇_a, ∇_b and ∇_c the slopes of the respective sub-cells along the mortar interface, this can be written as the following matrix-vector multiplication

$$\begin{pmatrix}\nabla_a\\ \nabla_b\\ \nabla_c\end{pmatrix} = M_{\text{mean}} \cdot \begin{pmatrix}\nabla_{ab}\\ \nabla_{bc}\end{pmatrix} \qquad \text{with} \qquad M_{\text{mean}} := \begin{pmatrix}1 & 0\\ 0.5 & 0.5\\ 0 & 1\end{pmatrix}. \tag{5.16}$$

These slopes are now used to calculate a reconstructed solution at the small side points, see figure 5.14b. They are $\pm\frac{1}{4}\Delta y$ apart from the big side flux point. For the sub-cell a this is $a - \frac{1}{4}\Delta y \cdot \nabla_a$ and $a + \frac{1}{4}\Delta y \cdot \nabla_a$. Again, this is a matrix-vector multiplication and the interpolation of the big side solution to the small virtual left and right side including the non-TVD reconstruction can be written in total as

$$\mathbf{u}^L = \begin{pmatrix}a\\ a\\ b\end{pmatrix} + M_L \cdot \begin{pmatrix}\nabla_a\\ \nabla_b\\ \nabla_c\end{pmatrix} \qquad \text{and} \qquad \mathbf{u}^R = \begin{pmatrix}b\\ c\\ c\end{pmatrix} + M_L \cdot \begin{pmatrix}\nabla_a\\ \nabla_b\\ \nabla_c\end{pmatrix},$$

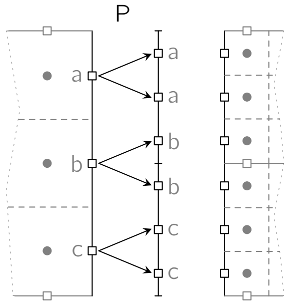
where the matrices M_L and M_R are given by

$$M_L := \Delta y \begin{pmatrix}-\frac{1}{4} & 0 & 0\\ \frac{1}{4} & 0 & 0\\ 0 & -\frac{1}{4} & 0\end{pmatrix} \qquad \text{and} \qquad M_R := \Delta y \begin{pmatrix}0 & \frac{1}{4} & 0\\ 0 & 0 & -\frac{1}{4}\\ 0 & 0 & \frac{1}{4}\end{pmatrix}. \tag{5.17}$$

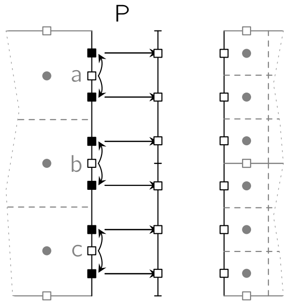
The vectors $(a, a, b)^\top$ and $(b, c, c)^\top$ are just the result of the interpolation operators I^L_{FV} and I^R_{FV} from equation (5.14) without the non-TVD reconstruction. Inserting this and putting equations (5.15) to (5.17) together, the new interpolation from the big side to the small virtual sides that includes a non-TVD reconstruction along the mortar interface reads

$$\mathbf{u}^L = \left(I^L + M_L \cdot M_{\text{mean}} \cdot M_\nabla\right)\mathbf{u}^B \qquad \text{and} \qquad \mathbf{u}^R = \left(I^R + M_R \cdot M_{\text{mean}} \cdot M_\nabla\right)\mathbf{u}^B.$$

The terms inside the parenthesis are $(N + 1) \times (N + 1)$ matrices that are solution independent and can therefore be computed once before the simulation. Figure 5.13 gives a good impression how well this reconstruction can diminish the numerical errors that are introduced due to a jump in mesh resolution. The extension to arbitrary polynomial degree N is straightforward.



(a) without reconstruction



(b) with reconstruction

Figure 5.14.: Mortar interpolation of the solution from the big side to the small virtual sides. Without a reconstruction along the mortar interface this reduces to a simple copy of the solution to the adjacent small virtual sides (a). This can be improved by using the slopes between the sub-cells of the big side element to reconstruct a solution at the flux points of the small virtual sides (b).

5.7. Overview of the implementation

In this section the integration of the Finite Volume sub-cell scheme into the algorithm of the Discontinuous Galerkin scheme from section 4.3 is explained. Comparing the semi-discrete formulations of the DG method in equation (4.11) and the FV sub-cell method in equation (5.9) one can see that they share common parts like the numerical fluxes, but overall they are quite different. The difficulty of the incorporation of the shock capturing with FV sub-cells into the existing DG implementation is to change the structure of the code as little as possible to keep its good numerical performance. Therefore, common blocks of the FV scheme and the DG method must be identified. Matching components of the FV scheme should then be implemented into the respective routines of the DG operator.

The DG operator consists of two main parts, the volume integral, which is an element local operation, and the surface integral, which couples adjacent DG elements via the numerical fluxes. In contrast to that, for an update of a FV sub-cell no volume integral but only surface integrals are computed. Nevertheless, the special structure of the FV sub-cells inside a DG element can be exploited. The faces of all sub-cells inside a DG element are of two different types. There are the

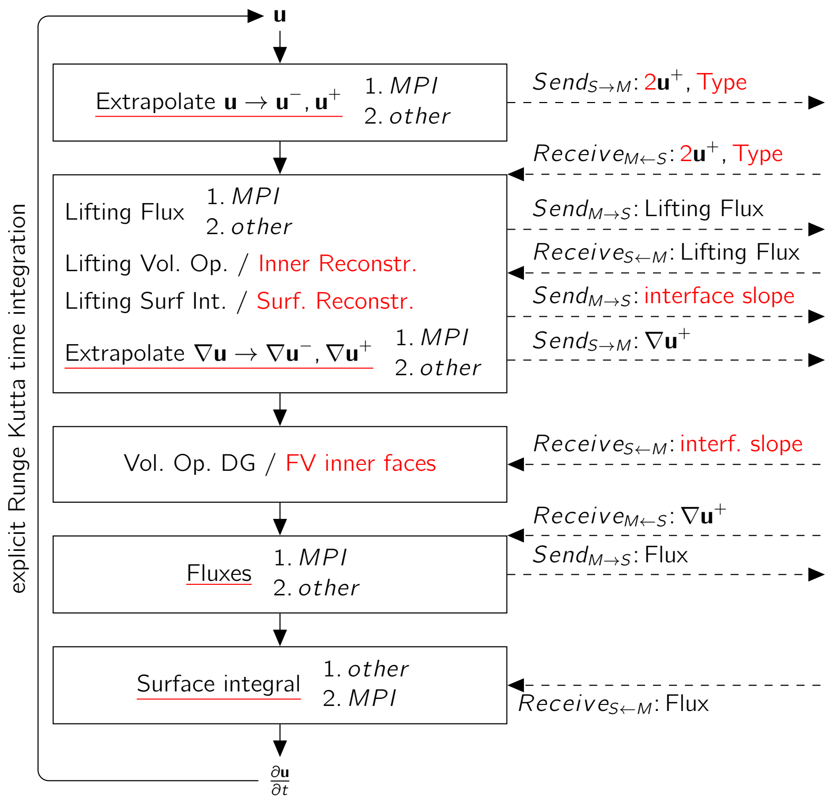

$\mathbf{u}$
Extrapolate $\mathbf{u} \rightarrow \mathbf{u}^-, \mathbf{u}^+$
1. *MPI*
2. *other*
$Send_{S\rightarrow M}$: 2$\mathbf{u}^+$, Type
$Receive_{M\leftarrow S}$: 2$\mathbf{u}^+$, Type
Lifting Flux
1. *MPI*
2. *other*
$Send_{M\rightarrow S}$: Lifting Flux
Lifting Vol. Op. / Inner Reconstr.
$Receive_{S\leftarrow M}$: Lifting Flux
Lifting Surf Int. / Surf. Reconstr.
$Send_{M\rightarrow S}$: interface slope
Extrapolate $\nabla\mathbf{u} \rightarrow \nabla\mathbf{u}^-, \nabla\mathbf{u}^+$
1. *MPI*
2. *other*
$Send_{S\rightarrow M}$: $\nabla\mathbf{u}^+$
Vol. Op. DG / FV inner faces
$Receive_{S\leftarrow M}$: interf. slope
$Receive_{M\leftarrow S}$: $\nabla\mathbf{u}^+$
Fluxes
1. *MPI*
2. *other*
$Send_{M\rightarrow S}$: Flux
Surface integral
1. *other*
2. *MPI*
$Receive_{S\leftarrow M}$: Flux
$\frac{\partial \mathbf{u}}{\partial t}$
explicit Runge Kutta time integration

Figure 5.15.: Flow chart of the hybrid Discontinuous Galerkin/Finite Volume sub-cells operator. Procedures underlined in red are modified to perform their specific task either for DG or FV elements. At the same time the BR1/2 lifting of the DG elements is computed, the 2nd order reconstruction of the FV sub-cells is built. The counterpart to the DG volume integral are the fluxes over inner FV sub-cell interfaces. Additional communication is required. Besides the information of which type (DG or FV) an element at a MPI interfaces is, for the FV reconstruction a second array of face data has to be transmitted, which is indicated by the red numbers 2.

faces that coincide with the faces of the DG element, from now on called outer faces, and the remaining inner faces. All the FV sub-cells inside a DG element can be understood as a block, see remark 5.2, where the numerical fluxes of the outer faces connect the block to other blocks or DG elements and the inner faces are a sort of volume operation of the block.

In figure 5.15 the modified algorithm is shown. The basic structure is the same as for the DG scheme in figure 4.2. Therefore, only the differences to the DG algorithm, marked in red, are explained. The extrapolation of the solution to the interfaces of the elements changes for FV elements. Here the average cell values from the sub-cells next to the interface are just copied to the boundary without a reconstruction since the slope computation over the interface requires the states of the adjacent elements which may not yet be available in a parallel setting. But in preparation for the reconstruction over these interfaces, an additional face array is filled which contains the slopes between the first and second sub-cell next to the interface, or for DG elements the nodal values at the first Gauss points next to the interface. As for the pure DG algorithm, the face state $\mathbf{u}^-$ must be transmitted from the slave to the master and additionally the array for the reconstruction. Therefore, the amount of communication doubles, which is indicated by the red number 2 in figure 5.15. Besides this, a single logical per interface is sent to tell the master if the slave element is a DG element or a FV sub-cells element.

The next block contains the lifting operation for the DG method which consists of a volume and a surface term. Since the lifting computes the gradients of the DG elements it is appropriate to build the slopes for the second order reconstruction in these routines. At the same time the routine of the volume operator of the lifting is performed for DG elements, all inner slopes of the FV elements are computed. Both operations are completely element local and are used to hide the communication of the lifting flux. During the evaluation of the lifting surface integral, the slopes over the interfaces can be computed using the additional array that was filled beforehand. After that, the gradients are extrapolated to the element interfaces, which needs the same modifications as for the states in the first step. In fact, the same extrapolation procedure is used for the state and the gradient. Additionally to the communication of the extrapolated gradients from the slave to the master, the reconstructed slopes over the element interfaces must be sent from the master to the slave, since the slave side needs them to limit the slopes inside the FV sub-cells next to the interface.

While this communication proceeds, the volume operator of the DG elements is evaluated to hide the transmission. For the Finite Volume sub-cells at all inner faces, the numerical fluxes are computed and directly integrated over the respective inner faces.

Next, the numerical fluxes at the element interfaces are evaluated. At mixed interfaces the state from the DG side must be interpolated to FV flux points first. The flux computation with the Riemann solver in this routine does not change since this just requires the left and right states. Then the fluxes have to be projected back to DG points for the DG side.

In the last step the fluxes are integrated along the interfaces. This routine has to be adapted for the Finite Volume sub-cells, but the basic method is the same. All in all, the incorporation of the FV sub-cell method into the DG algorithm does not affect the overall structure of the scheme. Many routines must be adapted to be able to handle both types of discretizations. But due to the logical subdivision of a DG element into a structured block of FV sub-cells there are always analogous parts to the components of the DG operator. Even so the communication in a parallel setting increases, in comparison to the pure DG method, the additional amount of data that must be transmitted is minimal and to hide these communications the same techniques are used.

6. Numerical results

The shock capturing for the Discontinuous Galerkin Spectral Element Method presented in the previous chapter is implemented in the open source computational fluid dynamics (CFD) solver *FLEXI*[1]. The fundamental numerical method of this solver is the DG scheme as given in chapter 4. Numerical results with the Discontinuous Galerkin method of this code can, for example, be found in [11, 10, 26]. They are limited to flow scenarios where no shocks or strong flow discontinuities occur. Such cases, which require a shock capturing, are presented in the following.

The *FLEXI* code is mainly developed in the Numerics Research Group (NRG) of the Institute of Aero- and Gasdynamics (IAG) and is published under the GNU General Public License v3. All meshes used for simulations in this chapter are built with the high order preprocessor *HOPR*[2], also supplied by the NRG as open source. Both software are executed from the command line and are controlled with parameter files. In appendix A *HOPR* parameter files as well as *FLEXI* parameter files for all investigated examples are provided. They can be used to repeat the computations. The only exception from this is the shock boundary layer interaction in section 6.4.4, which was simulated with a previous version of *FLEXI* not published as open source. The reason for this is that this computation is really large and expensive ($\approx$ 240 million DOFs, more than 90,000 processors), but it is expected that the simulation can be recomputed using the open source version of *FLEXI*. This example is also an exception in another way. Only this simulation is performed on Finite Volume sub-cells, which are distributed by the weights of the Gauss integration. This distribution of the FV sub-cells is investigated in detail in [84]. Therefore, all other examples in this chapter are performed on equidistant FV sub-cells but for most of them an equivalent simulation on Gaussian FV sub-cells can be found in the paper mentioned.

The numerical examples in this chapter are used to show the shock capturing with FV sub-cells. First, one-dimensional test cases validate the basic properties of the scheme. Next, the influence of the Finite Volume sub-cells on the order of convergence of the DG method is examined. In section 6.3 scaling tests show the parallel efficiency of the hybrid DG/FV sub-cells scheme for high performance

[1]www.flexi-project.org
[2]www.hopr-project.org

computations on thousands of processors. After proving all these essential properties, the method is applied to complex examples involving shocks.

6.1. Validation examples

In this section the shock capturing using Finite Volume sub-cells for the Discontinuous Galerkin method is applied to one-dimensional test cases and a three dimensional freestream example. In one space dimension, waves can only travel left or right but cannot interact with each other under an angle. This makes it possible to study the basic capabilities and show the advantages of the scheme. The influences of the polynomial degree N of the Discontinuous Galerkin method and the indicator thresholds are investigated. With the indicator thresholds, the amount of Finite Volume sub-cells used to resolve discontinuities can be influenced and the sensitivity of the simulation regarding this user defined parameters will be shown.

6.1.1. Sod shock tube

The Sod shock tube problem [81] is one of the most famous test cases for shock capturing. An initial discontinuity is located in the middle of the computational domain $\Omega = [0, 1]$, where the state on the left side is $\rho = 1, v_1 = 0, p = 1$ and on the right side $\rho = 0.125, v_1 = 0, p = 0.1$. This test case is used to analyze the behavior of the hybrid DG/FV sub-cell scheme with respect to the polynomial degree N of the DG elements. To obtain comparable results for the different polynomial degrees, the overall computational work is kept nearly the same by adjusting the number of grid cells.

The numerical setup is as follows. With the Jameson indicator on the pressure, the shock cells are marked; the upper and lower thresholds are chosen to 0.015 and 0.014. An additional Persson indicator with a threshold of -6.7 is used to avoid switching of FV sub-cell elements to DG elements if they contain high mode oscillations. The MinMod limiter is used to limit the slopes of the second order reconstruction. For an even number of grid cells, the discontinuity is perfectly located at a grid line and, depending on the indicator thresholds, the initial discontinuity may not be detected since the solution inside the DG elements is constant and perfectly smooth. Therefore, the very first time step is computed with enforced Finite Volume sub-cells in the whole domain. The CFL number is set to 0.8 and the HLLC Riemann solver is used for the numerical fluxes. At both boundaries of the domain Dirichlet boundary conditions with the initial states are used.

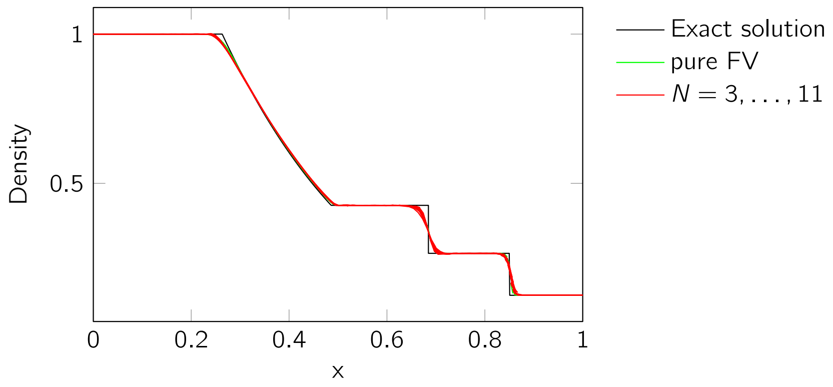


Figure 6.1.: Density of Sod shock tube problem at $t = 0.2$ with a DG polynomial degree ranging between $N = 3, \dots, 11$. To yield a total amount of work comparable for all polynomial degrees, the number of DG elements ranges between 44 elements for $N = 3$ and 10 elements for $N = 11$. The pure Finite Volume computation uses 252 FV sub-cells, which leads to nearly the same number of time steps times DOFs.

In figure 6.1, the density profiles for computations with nine different polynomial degrees $N = 3, \dots, 11$ are plotted. Besides this, the solution for a pure Finite Volume computation and an exact solution are shown. To achieve a numerical effort comparable for all cases, the number of DG elements of the equidistant grid is adjusted such that the DOFs multiplied with the number of time steps is nearly the same. The coarsest grid for $N = 11$ has 10 elements. With the above numerical setup this leads to 141 time steps until the end time $t = 0.2$. The total numerical effort for this computation can be quantified with

$$DOFs \cdot \#timesteps = 12 \cdot 10 \cdot 141 = 16920,$$

where 12 is the number of DOFs per element. For a fixed end time, the number of time steps only depends on the time step itself which in turn depends on the grid size, the CFL number and the polynomial degree, see equation (4.13). Hence, for the other polynomial degrees the number of elements directly influences the number of time steps via the grid size which must be adjusted such that a comparable load is obtained. For all investigated cases, the number of elements and therewith time steps are listed in table 6.1. Except the change of the polynomial

N	3	4	5	6	7	8	9	10	11
#elements	44	33	26	22	18	16	13	12	10
#time steps	100	107	113	127	127	136	126	149	141
≈ % FV	9.9	8.9	10.4	10.8	12.1	13.8	19.2	19.1	22.1
#FV elems	4.3	2.9	2.7	2.4	2.2	2.2	2.5	2.3	2.2

Table 6.1.: Numerical properties for the investigation of a varying polynomial degree on the solution of the Sod shock tube example. To obtain a comparable total computational effort, the grid has to be adjusted which results in the number of time steps for a CFL scale of 0.8. The amount of elements that are in total updated with the FV sub-cells method rises with increasing polynomial degree.

degree and the number of grid cells, the numerical setups of all computations are absolutely the same.

Figure 6.1 shows that for all polynomial degrees the solution is quite the same, regardless of the polynomial degree. Since all computations are comparable in terms of the numerical costs, one might conclude that there is no benefit of the underlying higher order Discontinuous Galerkin scheme. Therefore, the results of the hybrid scheme must be analyzed in detail. First of all, the pure Finite Volume scheme benefits from a weaker time step restriction which leads to bigger time steps and thereby to more grid cells to keep the numerical costs the same as for the hybrid scheme. To distinguish the nine coupled DG/FV sub-cells computations, a closer look at the contact discontinuity, where some deviations occur, is helpful. The closeup view in figure 6.2 shows that this wave is for polynomial degrees up to $N = 6$ quite comparably resolved. For larger N, the contact discontinuity is more and more flattened and the largest investigated polynomial degree of $N = 11$ shows the most smear. However, the differences are so small that any general conclusions would be rather disputable. Instead, the similarities of the solutions for different polynomial degrees are further investigated. One of the main features of high order methods is their small numerical dissipation. That means that for smooth solutions a higher polynomial degree with the same numerical costs in total should give better results. Indeed, the Sod shock tube example is not smooth but dominated by discontinuities which require the shock capturing using FV sub-cells. This method is designed to introduce dissipation to blur the sharp wave fronts such that they are numerically resolvable. These two contrary dissipative mechanisms must be in balance to produce stable results. An analysis of the amount of the FV sub-cells that are used to update all elements over all time steps, see table 6.1, shows that the higher the polynomial degree is, the greater

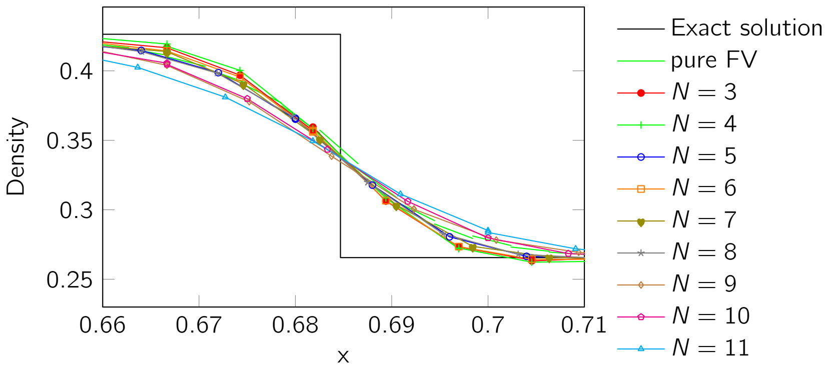


Figure 6.2.: Closeup view of the contact discontinuity of the Sod shock tube for select polynomial degrees. Increasing the polynomial degree of the DG elements from 3 to 5 improves the sharpness of the contact discontinuity. Further increment to $N = 11$ leads to bigger smear of this wave.

the percentage of elements that are updated with the FV sub-cells method. This behavior becomes quite clear with the following argument. To resolve a wave front, at least one DG element is switched to the FV sub-cell method. Since for large N there are only a few elements, a single FV sub-cells element leads to a greater amount of FV sub-cells than for a grid with more elements. For example, with ten elements, for the $N = 11$ case, two marked elements already lead to about 20% FV sub-cells. In the last row of table 6.1 the different FV amounts are converted to the absolute number of elements that are updated with the FV method during the whole computation. Except for $N = 3$ these numbers range between 2 and 3 which is remarkable low for two sharp wave fronts and the kinks at the bounds of the rarefaction wave. Since all numerical results nearly look the same, the following can be concluded from table 6.1. A higher polynomial degree leads to a bigger amount of FV sub-cells which introduces more dissipation into the computations than for the lower polynomial degrees. This is compensated by the smaller numerical dissipation of the DG scheme for higher polynomial degrees. It seems that these contrary dissipative mechanisms are in balance for a varying polynomial degree in this case. Altogether, the Discontinuous Galerkin shock capturing with Finite Volume sub-cells works very well for all N in this example, although a true benefit of the high order scheme is not visible in this case.

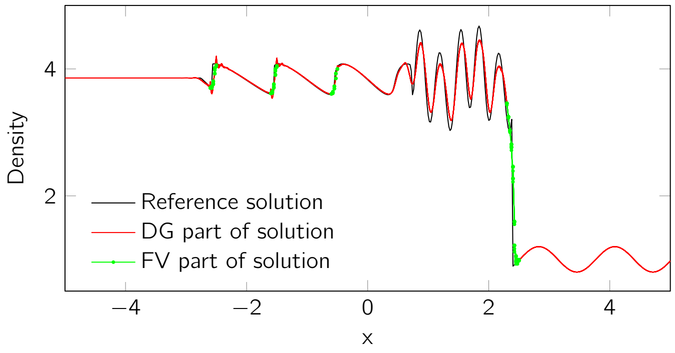


Figure 6.3.: Density of Shu-Osher fluctuations shock wave interaction problem at $t = 1.8$ with a polynomial degree of $N = 3$ on a grid with 100 elements. The results of 10 different computations with an upper threshold value of the indicator based switching between DG and FV sub-cells, varying from 0.007 to 0.12. The lower threshold is 0.005.

6.1.2. Shu-Osher density fluctuations shock wave interaction problem

Shu and Osher proposed in [79, 80] a test case for the interaction of smooth data with a discontinuity. A shock wave travels with Mach = 3 into a sinusoidal density wave. The domain is $[-5, 5]$ and the initial shock is placed at $x = -4$. To the left and right of this shock, the initial conditions are given by

$$(\rho, v_1, p) = \begin{cases} (3.857143, 2.629369, 10.33333) & x < -4 \\ (1.0 + 0.2 \cdot \sin(5x), 0.0, 1.0) & x \geq 4. \end{cases}$$

In section 5.4 an upper and lower threshold for the indicator, which manages the switching between DG elements and FV sub-cells, is introduced. This test case is used to investigate the sensitivity of these thresholds on the solution.

The numerical setup is as follows. The mesh consists of 100 elements and the polynomial degree is $N = 3$. With the JST indicator on the pressure, the elements containing a shock are marked and the Roe Riemann solver is used as numerical flux. Figure 6.3 shows the influence of the upper threshold. In general, the total variation diminishing property of the FV shock capturing introduces a noticeable amount of dissipation to resolve the shock without over- or under-shoots. This leads to the gap in amplitudes in the region behind the shock front in comparison

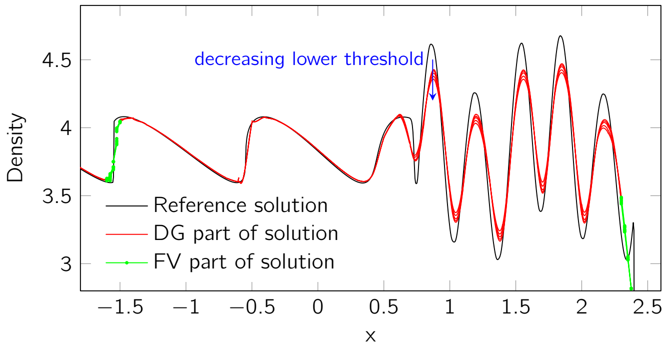


Figure 6.4.: Close-up view of the density of Shu-Osher fluctuations shock wave interaction problem at $t = 1.8$ with a polynomial degree of $N = 3$ on a grid with 100 elements. The results of 5 different computations with a lower threshold value of the indicator based switching between DG and FV sub-cells varying from 0.001 to 0.008. The upper threshold is 0.011.

to the reference solution, which was produced with a pure second order Finite Volume method on a very fine grid. In figure 6.3 ten computations with a varying upper threshold value between 0.007 and 0.12 are plotted. The lower threshold is fixed to 0.005 for all cases. Increasing the upper threshold further than the bound 0.12 leads to unstable solutions. Nevertheless, the range of the upper threshold is quite large and does not affect the solution much. All computations directly lie on each other, except for the three shock fronts between $x = -3$ and 0, where noticeable differences occur. Here, small oscillations appear for too large upper thresholds, but they do not influence the stability of the computation. This is only the case if those steps are not resolved with FV sub-cells. A single Finite-Volume sub-cell at these steps prevents the oscillations. Beginning with an upper threshold of 0.012 and larger, in more and more time steps these sharp parts are computed with DG and therefore introduce oscillations. The total amount of elements that are treated with the FV sub-cells method ranges between $\approx 2.5\%$ (for 0.007) and $\approx 1.6\%$ (for 0.12). Altogether, the sensitivity of the upper threshold on the solution is quite small.

The same numerical setup is now used to examine the influences of the lower threshold. In this case the upper threshold is fixed to 0.011 and the lower threshold varies between 0.001 and 0.008. In figure 6.4 the influences on the solution are

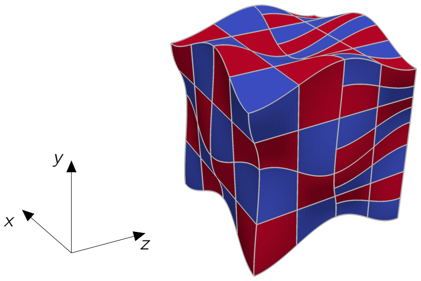


Figure 6.5.: Curved periodic mesh of the freestream preservation example. The right half in z-direction is refined in both other directions. The distribution of the DG and FV elements is manually fixed, such that every second element along the space filling curve is either FV (red) or DG (blue). The solution is initialized with a constant state which should not change over time or by mesh induced effects.

visualized. The solution for only five different lower thresholds (0.001, 0.002, 0.004, 0.006 and 0.008) are plotted to keep them distinguishable. Since the only difference can be seen right behind the shock front, figure 6.4 shows a close-up view of this region. All solutions are similar, except the amplitudes between $x = 0.5$ and $x = 2.5$ differ. Here, smaller thresholds lead to smaller amplitudes which can be explained as follows. The lower threshold triggers the switching from FV elements to DG sub-cells and if this value is smaller this is happening later when the shock wave leaves an element. Therewith, the overall amount of FV sub-cells that is used to resolve the shock front increases and directly leads to more introduced dissipation which diminishes the amplitudes. Varying the lower threshold between 0.001 and 0.008 results in an amount of FV sub-cells which ranges between $\approx 3.1\%$ and $\approx 2.0\%$, which confirms the hypothesis.

6.1.3. Freestream preservation

In contrast to the examples above, this is a fully three dimensional test case, which proofs the numerical conservation property of the hybrid DG/FV sub-cells method. To show this, a constant freestream solution is initialized, which should not change over time for periodic or consistent Dirichlet boundary conditions. After several time steps, the error in L_2- and L_∞-norm is analyzed. The grid is chosen quite complex to test several features of the code and their interactions at the same time.

To examine the freestream preservation on curved meshes the cubical domain is deformed by

$$\begin{aligned}\Omega &\to \Omega_{def}\\ x &\mapsto x + 0.1 \cdot \sin(x_1) \cdot \sin(x_2) \cdot \sin(x_3).\end{aligned}$$

The mesh consists of two blocks in z-direction where one block has twice the resolution in x- and y- direction than the other block. Thereby, mortar interfaces are introduced in the mesh. Every second element along the space filling curve is manually switched from a DG element to a FV sub-cells element, resulting in a checkerboard like distribution of the DG and FV elements. This is visualized in figure 6.5, where the curved elements are approximated with a geometrical polynomial degree of $N_{geo} = 3$. For freestream preservation, the polynomial degree of the approximation of the solution must be $N \geq 2 \cdot N_{geo}$ and hence the polynomial degree is chosen to $N = 6$. In the whole domain the constant state $\mathbf{u} = (1,1,1,1,1)^\top$ is initialized and periodic boundaries are used. After more than 300 time steps ($t = 0.5$), the error norms are

	$\mathbf{u}_1$	$\mathbf{u}_2$	$\mathbf{u}_3$	$\mathbf{u}_4$	$\mathbf{u}_5$
L_2	$4.49e^{-15}$	$6.12e^{-15}$	$6.04e^{-15}$	$5.89e^{-15}$	$1.86e^{-14}$
L_∞	$4.57e^{-13}$	$3.92e^{-13}$	$3.74e^{-13}$	$4.92e^{-13}$	$1.72e^{-12}$

They are in the region of the machine precision and therewith the freestream preservation of the shock capturing with FV sub-cells for the DG method is proven on curved meshes including mortar interfaces.

6.2. Order of convergence

In this section the order of convergence of the pure DG and the coupled DG/FV sub-cells method is investigated. One of the main properties of the Discontinuous Galerkin method is the ability to use high polynomial degrees N for the ansatz and test functions. The theory then states that the scheme is of high order, namely of order $N+1$. But since the Finite Volume method with reconstruction is of theoretical order 2, one cannot expect higher orders for the coupled method. This section focuses on the convergence rates for the hybrid DG/FV sub-cells scheme including mortar interfaces and is restricted to structured linear meshes. On unstructured curved 3D meshes, comparable convergence rates for the DGSEM implementation in the *FLEXI* code were found by Hindenlang et al. [39].

In a first step the convergence rate of the pure DG method is verified. Therefore, a simple test case of a periodic diagonal density sine wave, which is advected in direction $(1,1,1)^\top$, is used. The computational domain is a cartesian

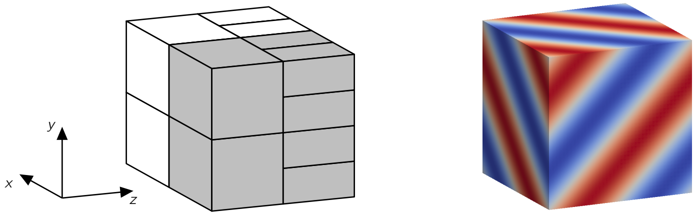


Figure 6.6.: Exemplary mesh with a baseline resolution of 2 elements per direction for the investigation of the order of convergence. The upper half in *z*-direction has always twice the resolution in *x*- and *y*-direction, leading to mortar interfaces of type 3. FV sub-cell elements are used in the gray half, while the other half are DG elements. The right cube shows the initial diagonal sine wave of the density.

poly. degree	baseline cells	L^2 error	L^2 order	L^∞ error	L^∞ order	theor. order
N=2	12	2.13e-04		2.12e-03		3
	16	8.96e-05	3.01	9.05e-04	2.96	
	20	4.58e-05	3.01	4.65e-04	2.98	
	24	2.65e-05	3.01	2.70e-04	2.99	
N=3	12	6.81e-06		7.98e-05		4
	16	2.16e-06	4.00	2.59e-05	3.91	
	20	8.85e-07	3.99	1.07e-05	3.94	
	24	4.27e-07	3.99	5.23e-06	3.95	
N=4	8	1.35e-06		1.78e-05		5
	12	1.77e-07	5.01	2.43e-06	4.90	
	16	4.19e-08	5.01	5.84e-07	4.96	
	20	1.37e-08	5.01	1.92e-07	4.98	
N=5	8	4.43e-08		6.02e-07		6
	12	3.87e-09	6.02	5.68e-08	5.82	
	16	6.85e-10	6.01	1.03e-08	5.92	
	20	1.79e-10	6.01	2.74e-09	5.95	

Table 6.2.: Errors and convergence rates of the density for a 3D advected sinus wave for the pure DG method with a polynomial degree ranging from $N = 2$ to $N = 5$. The upper half in *z*-direction is refined in *x*- and *y*-direction to twice the resolution, which requires mortar interfaces. The number of baseline cells denotes the number of unrefined elements in each space direction of the periodic cube.

box $[-1, 1]^3$, discretized with two structured blocks, one for the lower half and one for the upper half in z-direction. The division of the domain into two separate blocks is used to additionally test the mortar interfaces. Therefore, the upper block uses twice the grid resolution in x- and y-direction as the lower block. In figure 6.6 the mesh for a baseline resolution of 2 elements and the initial density sine wave is visualized. The specifications of the number of grid cells in tables 6.2 and 6.3 refer to this baseline resolution and lead to $(m^2 + (2m)^2) \cdot m$ elements in total for a baseline resolution of m elements.

For the polynomial degrees $N = 2, \ldots, 5$ the number of grid cells in each space direction is increased in each run by four elements to obtain the spatial order. To prevent a masking of the errors with a temporal error, the CFL number is reduced to 0.8. Table 6.2 lists the errors and convergence rates of the pure DG method. For all polynomial degrees, the theoretical order is accurately matched. Increasing the polynomial degree further requires time integration with an appropriate order or a very low CFL number because otherwise the spatial order is hidden behind the time error. In [84] the same convergence test for $N = 11$ required a CFL number of 0.005 to see the spatial convergence even so the computation is also stable for $CFL = 1$.

The convergence rate for the coupled DG/FV sub-cells method is investigated with the same test problem. Since this problem is smooth, the use of FV sub-cells is enforced in the right half in x-direction. The left half $x < 0$ is computed with the DG method. Therewith, mixed DG/FV interfaces as well as mortar interfaces are included in this study. Again, the tests are performed for polynomial degrees of $N = 2, \ldots, 5$ for the DG elements. But since the theoretical order of the Finite Volume method, including the presented reconstruction, is limited to two, one cannot expect higher numerical orders of convergence than two. Actually, this theoretical order is diminished even further by the use of the TVD limiters. Therefore, the tests are executed for three different setups. First, without any reconstruction which should yield a first order convergence. Subsequently then with the MinMod limiter and with a central limiter. The latter one is not supposed to be stable in general, but for this example it works and shows the full theoretical order. In table 6.3 the results are summarized. To keep the table compact this time only the L^2 error norms and convergence rates are listed, since the L^∞ values show the same behavior. The case without a reconstruction shows a convergence rate of nearly 1 for all polynomial degrees, as expected. For the MinMod limiter, the convergence rate is only about 1.6. This loss in convergence rate, compared to the theoretical order, is well known and is due to the TVD property of this limiter [71, 70]. By using more advanced limiters like the van Leer or the Sweby limiter for example, the convergence rates can be improved slightly, but do not

poly. degree	baseline cells	no reconstruction error	no reconstruction order	MinMod error	MinMod order	central error	central order
N=2	12	4.05e-02		7.88e-03		1.31e-03	
	16	3.17e-02	0.85	5.01e-03	1.58	7.37e-04	2.01
	20	2.61e-02	0.88	3.49e-03	1.61	4.71e-04	2.01
	24	2.21e-02	0.90	2.60e-03	1.62	3.27e-04	2.00
N=3	12	3.18e-02		5.01e-03		7.36e-04	
	16	2.46e-02	0.88	3.15e-03	1.61	4.14e-04	2.00
	20	2.01e-02	0.91	2.19e-03	1.62	2.65e-04	2.00
	24	1.70e-02	0.92	1.63e-03	1.64	1.84e-04	2.00
N=4	8	3.71e-02		6.72e-03		1.06e-03	
	12	2.61e-02	0.87	3.50e-03	1.61	4.71e-04	2.01
	16	2.01e-02	0.91	2.20e-03	1.62	2.65e-04	2.00
	20	1.64e-02	0.93	1.52e-03	1.64	1.69e-04	2.00
N=5	8	3.18e-02		5.02e-03		7.38e-04	
	12	2.21e-02	0.89	2.60e-03	1.62	3.27e-04	2.01
	16	1.70e-02	0.92	1.63e-03	1.63	1.84e-04	2.00
	20	1.38e-02	0.94	1.13e-03	1.64	1.18e-04	2.00

Table 6.3.: L^2 errors and convergence rates of the density for a 3D advected sinus wave for the coupled DG/FV sub-cells method with a polynomial degree of the DG approximation ranging from $N = 2$ to $N = 5$. Left of $x = 0$ the domain is computed with the DG method, right FV sub-cell elements are enforced. The upper half in z-direction is refined in x- and y-direction to have twice the resolution, which requires mortar interfaces. The number of baseline cells denotes the number of unrefined elements in each space direction of the periodic cube.

reach the full theoretical order of two. Nevertheless, this is not a big issue since the FV sub-cells shock capturing should only be applied in the regions of the shock, where a high order of convergence cannot be expected.

6.3. Parallel efficiency

The application of the presented method to large scale problems requires the usage of high performance computing (HPC) systems. The numerical effort is distributed to multiple processors with the aim of reducing the wall-clock time of the simulation. A perfect strong scaling is achieved when doubling the number of processors leads to a halving of the wall-clock time. The implementation of the hybrid DG/FV sub-cells method in the *FLEXI* code in general has no limit in the number of processors it can run on. But for a specific problem with a fixed number of elements, there are several bounds that restrict the number of processors. The grid cells are spread to the different cores such that each processor treats a subdomain of the mesh. Therefore, the upper bound in the possible number of processor is the number of elements when each core handles a single element. The lower bound is related to the amount of memory that is available on each core. A low number of processors leads to a large number of elements per core and hence the memory increases.

Remark 6.1. *The terms processor and core are used synonymously throughout this section. Both denote a physical computing unit inside a CPU, but not virtual cores in a simultaneous multithreading environment.*

The implementation of the pure Discontinuous Galerkin method in the *FLEXI* code has proven its ability to efficiently scale on several 10,000 cores [2, 38]. In this section, the focus is on the parallel scaling of the shock capturing with Finite Volume sub-cells. Following the best practice guideline for reporting performance results by Hoefler and Belli [40], the numerical setup, test environment and work-flow of this investigation are given in detail. All simulations are performed on the supercomputer "Hazel Hen", a Cray XC40-system of the High-Performance Computing Center (HLRS) in Stuttgart. This supercomputer consists of 7712 nodes, each equipped with 128 GB of memory and 2 sockets. Each socket holds an Intel Xeon CPU (E5-2680 v3) with 12 cores running at 2.50 GHz. The *FLEXI* code is built with the GNU Fortran (GCC) 6.3.0 compiler and the Cray MPI library 7.5.2 with the default compiler options of the *FLEXI* code, except for the switch on of the FV sub-cells. The full set of all compiler flags is given in the appendix A.3.

To test the parallel efficiency of the scheme for different problem sizes, twelve cases are investigated. All cases use the same numerical setup except the number of elements in the mesh. The polynomial degree of the Discontinuous Galerkin method is set to $N = 6$ and the solution is initialized with a freestream as in section 6.1.3. Since this case does not require any shock capturing, the use of Finite Volume sub-cells must be enforced. The mesh for the smallest case is a cuboid, which is discretized with 6^3 elements. This baseline mesh is refined for the other cases by always doubling the number of elements in one direction compared to the previous case, see table 6.4. For all cases the baseline configuration against which the parallel efficiency is measured is given in table 6.4 by the minimum number of nodes, where each nodes consists of 24 cores. The last case is so large that it requires more memory than available on a single node. Hence, the baseline configuration of this case is run on two nodes. Each case is run for all powers of

case	1	2	3	4	5	6	7	8	9	10	11	12
#elementsX	6	12	12	12	24	24	24	48	48	48	96	96
#elementsY	6	6	12	12	12	24	24	24	48	48	48	96
#elementsZ	6	6	6	12	12	12	24	24	24	48	48	48
min #nodes	1	1	1	1	1	1	1	1	1	1	1	2
max #nodes	1	2	4	8	16	32	64	128	256	512	1024	2048

Table 6.4.: The parallel efficiency is investigated for different problem sizes. Each case has twice the number of elements than the previous one. The maximal number of nodes, a case is run on, also doubles for each case, which corresponds to only 9 elements per core for all cases.

two between the minimal and maximal number of nodes which always leads to only 9 elements per core at the highest number of nodes.

The simulation is executed for exactly 100 time steps and the performance index (PID) is measured for the pure computational time without any file input/output or initialization. The performance index is the time that is required to update one DOF on one core and can be calculated as

$$\text{PID} = \frac{\text{wall-clock-time} \cdot \#\text{cores}}{\#\text{DOF} \cdot \#\text{time steps} \cdot \#\text{RK-stages}}.$$

To investigate the performance of the Finite Volume sub-cells shock capturing, four different synthetic indicator functions are used to mark elements for the FV sub-cells method. A pure Discontinuous Galerkin computation and a pure Finite Volume computation, where all elements are updated with the respective operator, are performed to compare the parallel performance of these operators. The influence of mixed DG/FV interfaces on the parallel performance is investigated with the following two indicators. The checkerboard indicator marks every second element along the space filling curve for the FV method, which leads to mixed DG/FV interfaces in the whole domain. This is the worst case in terms of operation counts, since the mixed interfaces require the interpolation of the DG side solution to FV points and the projection back of the numerical fluxes. This introduces additional operations compared to pure DG/DG or FV/FV interfaces. Nevertheless the load is distributed evenly among the processors since this extra work is required at all interfaces in the domain. Of course, this cannot be expected in a real world example where a shock front leads only locally to mixed DG/FV interfaces. This scenario is considered with the last synthetic indicator where the domain is halved in x-direction into a Discontinuous Galerkin part and a Finite Volume sub-cells part, which causes mixed interfaces only at $x = 0$. To obtain results that are comparable for the different indicators, the runs for all four indicators are performed in a single job.

Furthermore to gather some statistics, each single run is repeated five times. The median of those runs is used to calculate the parallel efficiency, which is defined by

$$\frac{\widetilde{\mathrm{PID}_1}}{\widetilde{\mathrm{PID}_P}} \cdot 100\%,$$

where $\widetilde{\mathrm{PID}_1}$ is the median of the performance indices on a single node and $\widetilde{\mathrm{PID}_P}$ the respective value for the computations on P nodes. For the largest cases, the baseline value is evaluated for two nodes. The parallel efficiency over a varying number of cores is plotted for all cases in figure 6.7. Each case is represented by a colored line, which starts at 100% with the baseline run on the lowest number of nodes. The symbols mark the median of all repetitions of the specific configuration and the error bars show the best and worst run. For all problem sizes, the number of cores is increased until only 9 element reside on a single core, which are the right end points of the curves. The pure DG computation in the top left plot of figure 6.7 shows a superlinear scaling, where the parallel efficiency becomes greater than 100% for nearly all problem sizes. This superlinear scaling can be either explained with caching effects of the processors or a not so good baseline run on the lowest number of cores. In contrast to the superlinear scaling there are also runs where the efficiency is substantially below 100% and the variability heavily increases. This behavior is especially visible for the largest (or second largest) number of cores in a case, where only nine (or eighteen) elements reside on a single core. Due to this small number of elements per core, the amount of communication compared to the local work is quite large. In this case the interaction with other jobs running at the same time on the supercomputer affects the parallel performance. Wright et al. [93] traced this loss of performance mainly to the variation of the MPI communication over the network when different jobs share the same network resources. The results for the other indicators show a similar behavior, but the superlinear scaling effects are less pronounced. Furthermore the load imbalance of the half/half indicator, shown in the bottom right plot of figure 6.7, leads to a slightly diminished parallel efficiency for some cases. Nevertheless it is for most of the computations over 90%, which is still remarkably good. However these plots of the parallel efficiency present relative data only and are not comparable among the different indicators or problem sizes.

Therefore in figure 6.8 another point of view on the same data is presented. In this figure, the performance index is plotted over the number of DOFs per core, which shows the scaling results more or less the other way round than before. In contrast to figure 6.7 going to the right on the x-axis, the number

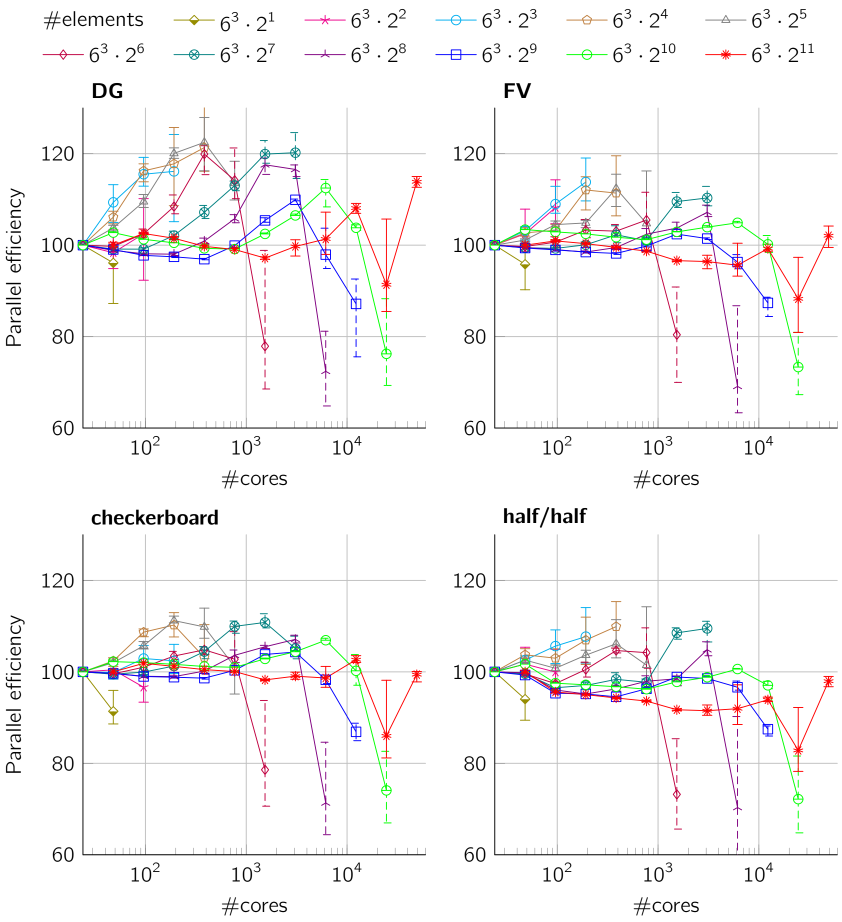


Figure 6.7.: Parallel efficiency of a strong scaling for different problem sizes. Each color/line corresponds to a specific mesh, for which the number of elements is given in the legend. The baseline simulation is performed on a single node (24 cores) except for the largest cases, which does not fit on a single node due to memory restrictions. For all meshes, the number of cores is doubled until each core has only 9 elements. To reduce statistical effects every single simulation is repeated five times and the variability is visualized with the error bars.

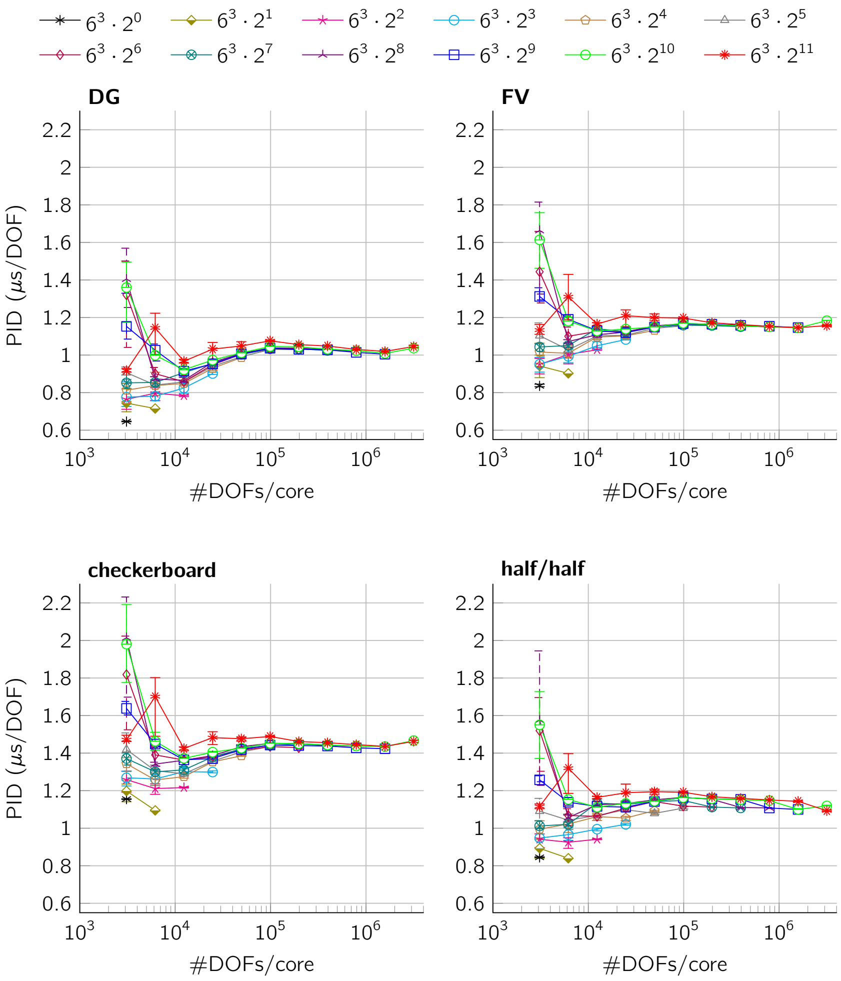


Figure 6.8.: Performance index of a strong scaling for different problem sizes and distributions of DG and FV elements. A good performance can be achieved for about 10,000 DOFs and more per core. Lower numbers of DOFs per core can lead to better results but the variability, caused by the sharing of network resources with other jobs, may reduce the parallel performance.

of cores is decreasing. The 100% baseline point of each line now corresponds to the most right point of each case and the runs with the highest number of processors (9 elements per core) are on the left side. This time the performance is not shown as a ratio to the baseline run, but looking at the absolute numbers of the PID allows to compare different cases which represent a big bandwidth of problem sizes as well as different distributions of DG and FV elements enforced by the synthetic indicators. The pure Discontinuous Galerkin computations in the top left plot of figure 6.8 show that the performance is absolutely independent of the problem size. Even for the largest case with $6^3 \cdot 2^{12} = 884,736$ elements the performance index is about $1\mu s$/DOF. For low numbers of DOFs per core the PID can, on the one hand, fall to about $0.8\mu s$/DOF which is explained by caching effects where most of the data a processor works on fits into its cache. This reduces reading and writing of data from and to the memory and thereby speeds up the access of data. On the other hand the performance index can increase for low number of DOFs per core. This is paired with a greater variability in the results which indicates an influence by jobs running at the same time on the supercomputer, which use the same network resources.

For a pure FV computation the results in the top right plot of figure 6.8 are the same, except that they show an about 15% larger PID. One of the main reasons for this is the memory access pattern of the Finite Volume sub-cells method which is not as consecutive as for the DG operator. The evaluation of the numerical flux at all inner FV interfaces is performed slice-by-slice in the ξ-, η- and ζ-direction individually and requires the states from both sides of the interface. This slice-wise data is obtained from the array of the volume solution. For the ζ-direction the storage order of the data in the slices corresponds to the storage order of the volume data, which results in sequential memory accesses. In ξ- and η-direction the memory access for the slice data in the volume array is strided. Hence the improvement, due to caching effects, for lower number of DOFs per core is not as distinct as for the pure Discontinuous Galerkin case. Nevertheless the performance indices of the DG operator and the FV sub-cells method are in the same range.

In the bottom plots of figure 6.8 the influence of the element interfaces between those methods is investigated. The left bottom plot shows the results for the synthetic checkerboard indicator where all element interfaces in the whole domain are mixed DG/FV interfaces. These mixed interfaces require additional operations and hence the checkerboard like distribution of DG and FV elements is the worst imaginable case. This can be seen in a larger PID, but the overall behavior is still the same which is not surprising since the extra work at the mixed interfaces has to be done by all processors. Hence the load is still in balance between all cores. In a real world example this cannot be expected, but it is more likely that only locally, at shock fronts, mixed DG/FV interfaces are necessary. This scenario is emulated by the half/half indicator. In this case not all processors have to perform the extra work at mixed interfaces which leads to load imbalances. However this does not influence the parallel performance. The results in the bottom right plot of figure 6.8 show values between the values of the pure DG and the pure FV computations in the top row plots.

6.4. Complex examples

The one-dimensional examples showed the abilities of the shock capturing for the Discontinuous Galerkin method using Finite Volume sub-cells. The fundamental properties and the scaling results for massive parallel setups enable the application of the scheme to more complex examples.

Remark 6.2 (Visualization with ParaView)**.** *The pseudo color or contour plots of simulation results in this work are generated with the application ParaView[3]. The state files of the FLEXI code are written in a HDF 5 format, which is not directly readable by ParaView and therefore the state files must be converted to an appropriated format first. Another possibility that does not require a conversion of the files, is the use of custom file readers, which ParaView offers via a plugin infrastructure. The FLEXI code is delivered with such a ParaView plugin that allows direct visualization of the HDF 5 state files. This has multiple advantages over a conversion approach. First of all, no additional disk space is required for the converted files, which actually can be very large for accurate visualizations of multiple derived quantities. Second, it is built in a way that the original routines of the FLEXI code are available. This is extremely useful for the calculation of the reconstruction of the Finite Volume sub-cells or for the gradients of the Discontinuous Galerkin elements via the lifting procedure. Another benefit is the possibility to choose the number of super sampling points of the rendering freely. Therewith, one can first adjust the layout with a low order visualization and only for the generation of nice pictures switch to a high order visualization. This improves the overall workflow drastically. Furthermore, the ParaView plugin directly inherits all improvements in the FLEXI code, since they share a common source code basis. From a technical point of view the ParaView plugin is linked against the FLEXI-library, which consists of the whole flow solver except for the main routine.*

[3]www.paraview.org

6.4.1. Double Mach reflection

This test problem is a famous test case for shock capturing methods in general and especially for high order methods. Even so Ernst Mach investigated reflections of shock waves already in the 19th century, it has been Woodward and Colella [92] who suggested this problem to test the ability of numerical schemes to represent shock and contact discontinuities. A Mach 10 oblique shock wave is sent into a reflecting wall, which is equivalent to a shock wave encountering a 30° wedge. The initial conditions are given by the Rankine-Hugoniot conditions

$$(\rho, v_1, v_2, p) = \begin{cases} (8.0, 8.25 \cdot cos(30^\circ), -8.25 \cdot sin(30^\circ), 116.5) & x < x_0 + \sqrt{\frac{1}{3}}y \\ (1.4, 0.0, 0.0, 1.0) & x \geq x_0 + \sqrt{\frac{1}{3}}y \end{cases},$$

where $x_0 = \frac{1}{6}$ is the start of the wall and the computational domain is $\Omega = [0, 4] \times [0, 1]$, which is discretized by an equidistant cartesian mesh.

The numerical setup for all presented computations is nearly the same. Only the mesh resolution and the variable the indicator acts on are varied, which will be mentioned each time they deviate from the following settings. The JST indicator with a lower and upper threshold of 0.005 and 0.01 is used to detect the elements containing shocks. An additional Persson indicator with a threshold of -5.5 avoids switching FV sub-cell elements to DG as this would introduce high mode oscillations. The limiter of the second order reconstruction is the MinMod limiter. A Riemann solver that does not show the carbuncle phenomenon is required, here the HLLE solver is chosen. The polynomial degree of the DG elements is $N = 5$. All results are shown at the end time $t = 0.2$, where the time step is calculated with $CFL = 0.9$.

This test case is used to present the advantages of the high order Discontinuous Galerkin scheme for the non-shock parts of the solution. Therefore, the hybrid DG/FV sub-cell method is compared to a pure second order Finite Volume scheme. The Double Mach reflection is known to produce small vortices along the slip line for numerical schemes with low numerical dissipation. For a mesh with 120×30 elements, the hybrid DG/FV sub-cell schemes is capable of resolving the first small vortices, which can be seen in the top plot of figure 6.9. The idea is now to enforce the use of FV sub-cells throughout the whole domain and refine the grid until a comparable result is obtained. As it turns out, the mesh must be refined 8 times in each space direction leading to 960×240 elements. It should be noted that this means $(6 \cdot 960) \times (6 \cdot 240) = 5760 \times 1440$ FV sub-cells, since a polynomial degree of $N = 5$, with 6 sub-cells in each space dimension, is

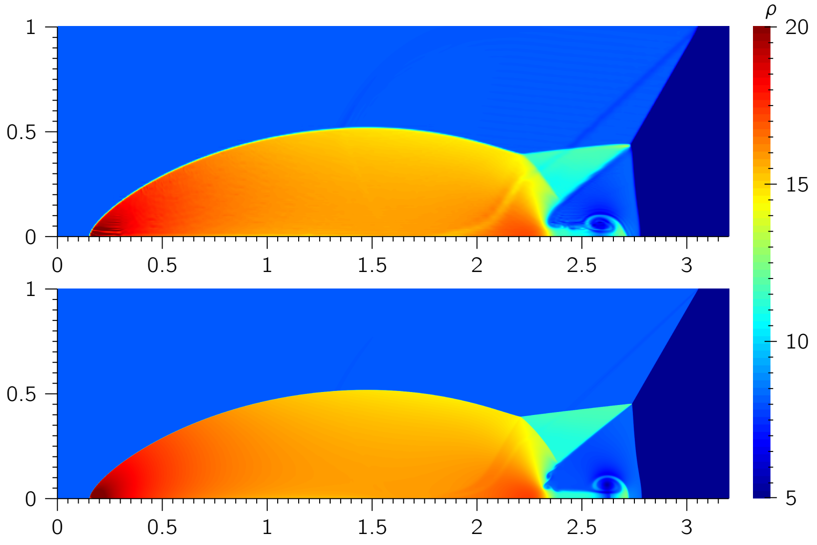


Figure 6.9.: Comparison of the density of a hybrid DG/FV sub-cells computation with a polynomial degree of $N = 5$ on a coarse grid (120×30) in the top row with a pure FV calculation on a finer grid (960×240 underlying DG elements) in the bottom row. The results are comparable even so the pure FV solution required 64 times more DOFs.

used. The bottom row of figure 6.9 shows the result of the pure Finite Volume computation. It is quite clear that the higher order of the DG scheme introduces considerably less numerical dissipation than the pure FV method, but nevertheless it is remarkable that in this case the Finite Volume requires 64 times more DOFs to catch up.

Another way of showing the influences of the numerical dissipation on the quality of the solution is to investigate the amount of FV sub-cell elements that are used during the computation. Since this depends on the thresholds of the indicator, it is possible to increase the number of FV sub-cell elements by reducing the indicator thresholds. Therefore, the setup of the above hybrid DG/FV sub-cell computation is repeated on a finer mesh with 480*x*120 DG elements first. Then, the thresholds are diminished while all other solver settings stay the same to see the influences of an increasing amount of FV sub-cells. A closeup view of the density in the interaction zone at final time $t = 0.2$ is visualized in figure 6.10.

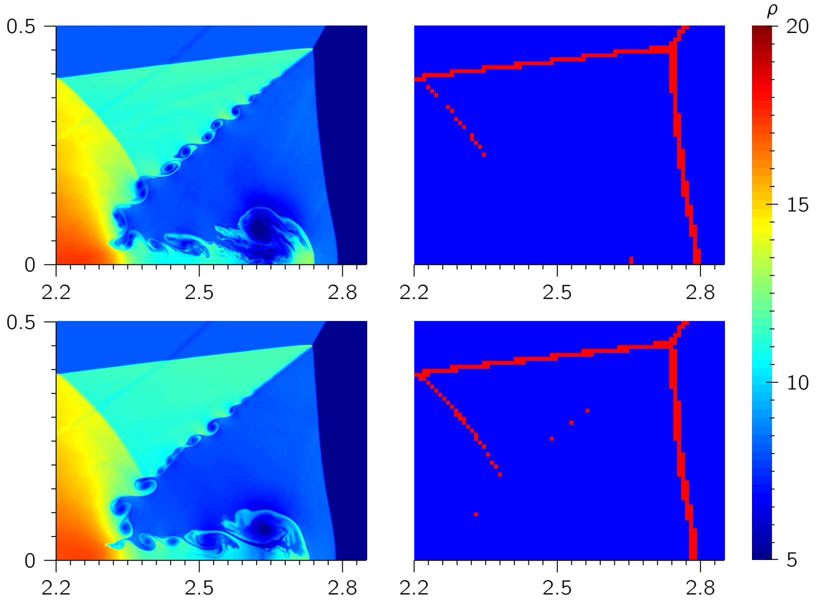


Figure 6.10.: Closeup view of the density of two computations, where only the thresholds of the indicator are different. This shows the influence of the numerical dissipation. The bottom row shows the results with lower thresholds, leading to more FV sub-cell elements, which are visualized on the right in red. More FV sub-cells lead to more numerical dissipation and hence not so fine resolved structures.

The first row shows the results for the original thresholds and the second row for smaller thresholds and a look on the right column shows immediately the greater amount of FV sub-cells.

Due to the much finer grid than before, the Kelvin-Helmholtz instabilities along the primary slip line produce vortex-like structures. In the top right of figure 6.10 it can be seen that the JST indicator on the pressure with upper and lower thresholds of 0.01 and 0.005 is only active along the shock fronts. It completely ignores the fine structures along the slip line since here the pressure does not jump. The thresholds are now lowered to 0.004 and 0.003, respectively, to enforce more FV sub-cell elements. In the bottom row of figure 6.10 one can see that, even so the thresholds were lowered a lot, the density contours are still comparable.

Along the secondary reflected shock and along the slip line more elements are detected for the shock capturing. This directly effects the resolution capabilities and leads to a loss of the very fine structures, even so the total amount of FV sub-cell elements accumulated over the whole computational time in both cases is very low. For the original thresholds it is 0.748% and 0.823% for the smaller thresholds. This means that over 99% of the computation can benefit from good properties, like e.g. low numerical dissipation or high order of the Discontinuous Galerkin method, and only the shock fronts that are not stably resolvable with the DG polynomials are handled by the Finite Volume sub-cell elements. Overall, the results are in good accordance to other high order results, like from Zanotti et al. [95].

6.4.2. Forward facing step

Another example to test the shock capturing capabilities of a numerical scheme was also presented by Woodward and Colella in [92] and is here used to investigate the influence of the variable, either density or pressure, the indicator acts on. Here, air at Mach$=3$ hits a step in a two dimensional wind tunnel, which has a length of 3 and a height of 1. The step is located at $x = 0.6$ and is 0.2 high. Therefore, the computational domain is given by $[0, 3] \times [0, 1] \setminus [0.6, 3] \times [0, 0.2]$. Reflective wall boundaries are applied to the top, bottom and step boundaries. In the x-direction inflow and outflow boundary conditions are used. The initial conditions are a freestream with density $\rho = 1.4$, velocity $v = (3, 0)$, pressure $p = 1$ and the heat capacity ratio of air $\gamma = 1.4$. The simulation is performed until the final time $t = 4.0$ on an equidistant mesh with an element size of $h = 1/100$, leading to $300 \times 100 - 20 \times 240$ DG elements. The polynomial degree of the DG approximation is $N = 5$ and the HLLE Riemann solver is used as numerical flux. To detect shocks, the Persson indicator with lower and upper thresholds of -6.5 and -5.5 is used. Two computations, one where the Persson indicator is computed on the density and one on the pressure, are compared. Additionally, the very first time steps until $t = 0.001$ are performed with FV sub-cell elements in the whole domain to relax the initial impingement of the freestream with the step of the wind tunnel. The second order reconstruction is limited with the MinMod limiter.

In figure 6.11, the numerical schlieren for both computations where the numerical setups are identical, except for the variable of the Persson indicator, are shown. At a first glance, the results do not show any differences, the shock fronts are identical. A closer look at the Kelvin-Helmholtz instability that develops along the top shear wave reveals sharper and more turbulent structures for the case where the Persson indicator is evaluated on the pressure. Figure 6.12 shows that

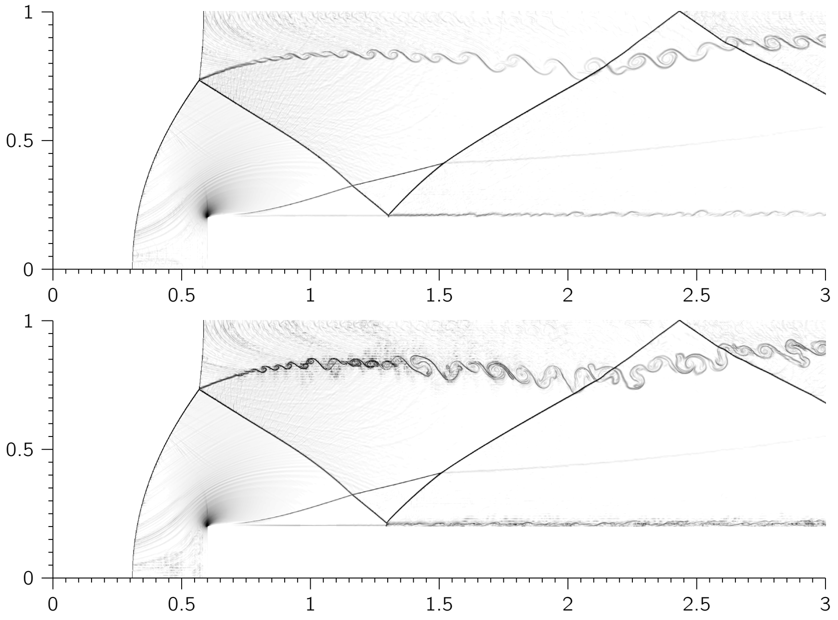


Figure 6.11.: Numerical schlieren of the forward facing step at $t = 4.0$ for nearly identical setups, where only the variable the Persson indicator acts on is different. The top image shows the results for the Persson indicator on the density, while the bottom image is computed with the Persson indicator on the pressure.

a Persson indicator acting on the density detects more FV sub-cell elements along this shear wave. This prevents the generation of small turbulent structures. In contrast to that, the pressure across the shear wave is constant. Hence, evaluating the Persson indicator on the pressure variable instead on the density prevents the detection of turbulent structures in the density. Not a single DG element is switched to FV along the slip line in this case.

Another feature that is only visible for high order computations with a very low numerical dissipation are the acoustic waves that are emitted from the shear wave. Due to their small amplitudes, they would be damped very fast for a pure Finite Volume method.

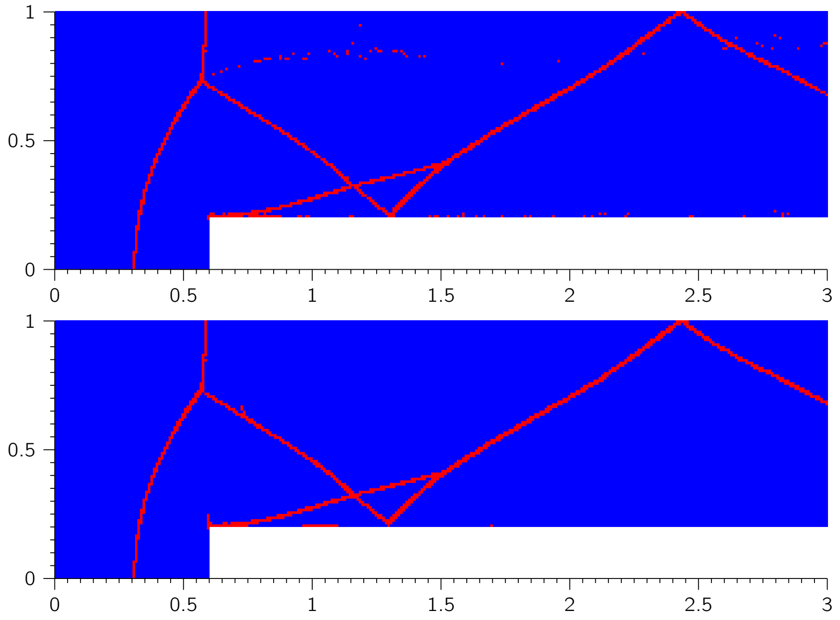


Figure 6.12.: Distribution of DG and FV sub-cell elements of the forward facing step at $t = 4.0$. The top image shows the results for the Persson indicator on the density, while the bottom image is computed with the Persson indicator on the pressure.

6.4.3. Two dimensional Riemann problem

The one-dimensional Riemann problem consists of piecewise constant initial states with a single discontinuity. The Sod shock tube from section 6.1.1 is the most famous example for a Riemann problem of the Euler equations. Its solution consist of the three elementary waves: A rarefaction wave, a contact discontinuity and a shock wave. When transferring the basic idea of the Riemann problem of piecewise constant initial data to two space dimension, the situation becomes quite complicated. In contrast to the one-dimensional case, waves now have an extension orthogonal to their direction of motion which might not be parallel to the fronts of the other waves. This leads to interactions of waves under a non-zero angle, which increases the complexity tremendously. Therefore, the basic setup

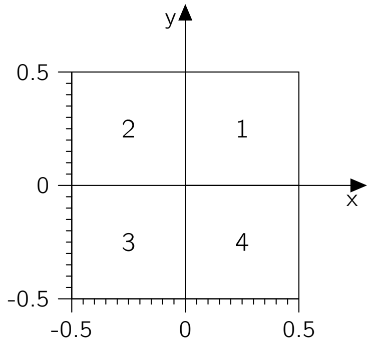


Figure 6.13.: The domain of the two dimensional Riemann problems is divided into four quadrants, numbered 1–4. In each quadrant a constant state is initialized such that only single elementary waves connect adjacent quadrants.

for the two dimensional Riemann problems is restricted as follows. The domain is quadratic and halved in both directions, see figure 6.13.

In each of the resulting quadrants, a constant state is initialized in a special manner. The four states are chosen in such a way that the quadrants are separated by a single elementary wave. Schulz-Rinne [77, 78] classified in total sixteen different combinations of initial data. Among these configurations, nine include at least one shock wave which are of major interest for the presented shock capturing. These configurations are summarized in table 6.5. There are two configurations with shock waves only, one configuration with two shock and two rarefaction waves and six configurations with two contact discontinuities. Besides the types of the four waves (shock (S), contact (J) or rarefaction (R)), a minimal set of required initial conditions is given. The remaining initial states of all four quadrants can then be calculated from the minimal set, using the conditions holding for the elementary waves. For the shock wave this is the Rankine-Hugoniot condition. At the rarefaction wave one uses the Riemann invariant and the isentropic relation

$$v + \frac{2c}{\gamma - 1} \quad \text{and} \quad \frac{p_l}{p_r} = \left(\frac{\rho_l}{\rho_r}\right)^{\gamma},$$

where c is the speed of sound and γ the heat capacity ratio. Over the contact discontinuity, pressure and normal velocity are the same, i.e. $p_l = p_r$ and $v_l = v_r$.

In the following, the general numerical setup is summarized. These parameters are the same for all configurations if they are not given explicitly for a specific

Cfg	wave types	initial conditions
3	$\begin{array}{ccc} & \overleftarrow{S}_{21} & \\ \downarrow S_{32} & & \downarrow S_{41} \\ & \overleftarrow{S}_{34} & \end{array}$	$\begin{array}{rcl} p = 0.3 & \vert & p = 1.5, \rho = 1.5, v_1 = 0, v_2 = 0 \\ \hline & \vert & \end{array}$
4	$\begin{array}{ccc} & \overleftarrow{S}_{21} & \\ \uparrow S_{32} & & \downarrow S_{41} \\ & \overrightarrow{S}_{34} & \end{array}$	$\begin{array}{rcl} p = 0.35 & \vert & p = 1.1, \rho = 1.1, v_1 = 0, v_2 = 0 \\ \hline & \vert & \end{array}$
6	$\begin{array}{ccc} & \overrightarrow{R}_{21} & \\ \uparrow S_{32} & & \downarrow S_{41} \\ & \overleftarrow{R}_{34} & \end{array}$	$\begin{array}{rcl} \rho = 0.5 & \vert & p = 1, \rho = 1, v_1 = 0, v_2 = 0 \\ \hline & \vert & \end{array}$
E	$\begin{array}{ccc} & \overleftarrow{S}_{21} & \\ J_{32} & & \downarrow S_{41} \\ & J_{34} & \end{array}$	$\begin{array}{rcl} p = 0.4 & \vert & p = 1, \rho = 1, v_1 = 0.1, v_2 = 0 \\ \hline \rho = 0.8 & \vert & \end{array}$
F	$\begin{array}{ccc} & \overrightarrow{S}_{21} & \\ J_{32} & & \uparrow S_{41} \\ & J_{34} & \end{array}$	$\begin{array}{rcl} p = 1, \rho = 1 & \vert & p = 0.4, v_1 = 0, v_2 = 0 \\ \hline \rho = 0.8 & \vert & \end{array}$
J	$\begin{array}{ccc} & J_{21} & \\ \downarrow S_{32} & & \downarrow S_{41} \\ & J_{34} & \end{array}$	$\begin{array}{rcl} \rho = 2, v_2 = 0.3 & \vert & p = 1, \rho = 1, v_2 = -0.3 \\ \hline p = 0.4 & \vert & v_1 = 0 \end{array}$
G	$\begin{array}{ccc} & \overrightarrow{R}_{21} & \\ J_{32} & & \downarrow S_{41} \\ & J_{34} & \end{array}$	$\begin{array}{rcl} p = 0.4 & \vert & p = 1, \rho = 1, v_1 = 0.1, v_2 = -0.3 \\ \hline \rho = 0.8 & \vert & \end{array}$
H	$\begin{array}{ccc} & \overleftarrow{R}_{21} & \\ J_{32} & & \uparrow S_{41} \\ & J_{34} & \end{array}$	$\begin{array}{rcl} p = 1 & \vert & p = 0.4, v_1 = 0.1, v_2 = 0.1 \\ \hline \rho = 0.8 & \vert & \rho = 1 \end{array}$
K	$\begin{array}{ccc} & J_{21} & \\ \downarrow S_{32} & & \uparrow R_{41} \\ & J_{34} & \end{array}$	$\begin{array}{rcl} p = 0.35 & \vert & p = 1.1, \rho = 1.1, v_1 = 0, v_2 = 0 \\ \hline & \vert & \end{array}$

Table 6.5.: Elementary waves and initial conditions of all two dimensional Riemann problems involving shock waves. The configurations are labeled as in [77]. Only the necessary initial conditions are given. All remaining initial states are defined by the conditions of the waves, e.g. the Rankine-Hugoniot condition for the shock wave.

configuration. The polynomial degree is set to $N = 5$ for the DG elements and the JST indicator on the density is used to detect the elements that need a FV sub-cells treatment. The slopes of the second order reconstruction are limited with the MinMod limiter and the numerical fluxes are resolved with the Roe Riemann solver. Furthermore, the basic mesh consists of 100×100 uniform elements. The boundary conditions for all configurations are Dirichlet boundary conditions where the outer state is given by the exact solution of the respective elementary wave. Since the discontinuity of the outer state of the boundary condition is sharp but the numerical calculation smears the wave fronts, this might produce spurious artifacts that travel into the computational domain. In some configurations, the domain is therefore extended at the problematic boundaries to keep the artifacts outside of the original domain. If this extension of the domain is needed, it is explicitly mentioned in the detailed description of the individual configurations. Furthermore, the Riemann solver for the boundary condition fluxes is changed in some cases from the Roe Riemann solver to the HLLE.

6.4.3.1. Configurations without contact discontinuities

There are three configurations without a contact discontinuity. The configurations 3 and 4 are initialized with shock waves only and configuration 6 consists of two shock and two rarefaction waves.

Configuration 3

Four shock waves are initialized at the quadrant interfaces travelling downwards and left in this example. The basic mesh of 100×100 DG elements is refined in the region $[-0.3, 0] \times [-0.3, 0]$ to half the grid size. In total, the mesh consists of $100^2 - 30^2 + 60^2 = 12700$ elements. The local refinement requires the mortar technique. In contrast to all other configurations, the Roe Riemann solver is replaced with the HLLE Riemann solver out of stability reasons. Also, the Riemann solver for the boundary conditions is the HLLE. The lower and upper thresholds of the JST indicator are chosen as 0.004 and 0.009 and additionally to this indicator the Persson indicator with a threshold of -6.0 is used to avoid switching of FV sub-cell elements to DG elements, if they contain high mode oscillations. The numerical solution at final time $t = 0.3$ is shown in the first row of figure 6.14. Regarding the main flow structures, the result shows a good agreement to the results in [53]. Nevertheless, due to the high order of this simulation, many more small scale features are resolved, which were also detected by Dumbser et al. [24].

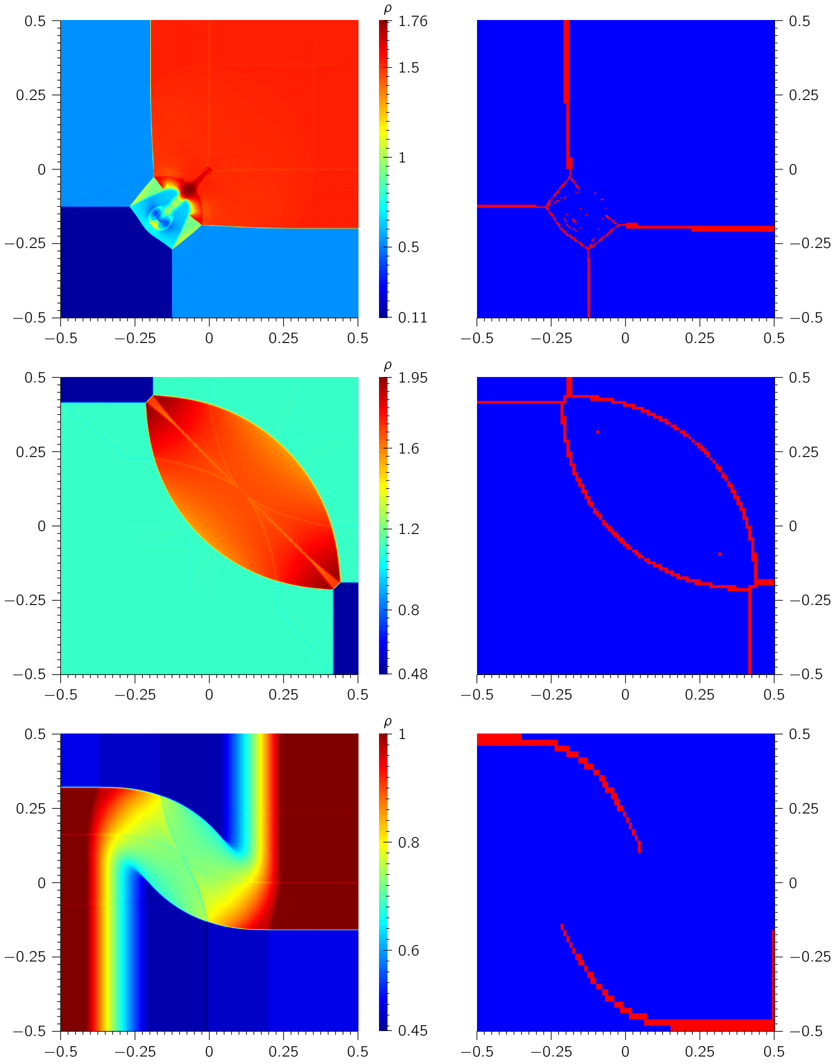


Figure 6.14.: Two dimensional Riemann problem, configurations 3, 4 and 6. Left: Density at final time. Right: FV (red) and DG (blue) elements.

Configuration 4

This configuration is initialized with four shock waves, but in contrast to configuration 3 the left and bottom waves travel in opposite direction. The lower and upper thresholds of the JST indicator are chosen as 0.005 and 0.002, but no additional Persson indicator is needed. The boundary condition Riemann solver is the HLLE. In the second row of figure 6.14, the numerical results at $t_{end} = 0.25$ are shown. Most shock waves are resolved with a single FV sub-cells element or at most in two elements if the shock front is near a DG element boundary. As one can see, this example does not produce small structures. Therefore, the mesh resolution can be reduced. Results with a mesh sizes of 67×67 elements can be found in [84].

Configuration 6

Two shock and two rarefaction waves which move in clockwise direction are initialized in this case. To avoid spurious artifacts at the boundary conditions of the rarefaction waves, the domain has been extended in y-direction to $[-1, 1]$, with the same mesh size of $h = \frac{1}{100}$ and additionally the HLLE Riemann solver is used for this boundaries. Lower and upper thresholds for the JST indicator are 0.004 and 0.01 and as for configuration 3, the Persson indicator is used with a threshold of -6.0 to avoid switching of FV sub-cell elements to DG elements, if they contain high mode oscillations. The last row of figure 6.14 contains the results at $t_{end} = 0.2$. At most, two FV sub-cell elements are required to resolve the shock front. Only at the very beginning of the computation, the rarefaction waves require the FV shock capturing technique. After a short time these waves gain a certain width such that they can be easily resolved by the DG polynomials. A comparison to the numerical investigations of Schulz-Rinne [78] shows a good agreement.

6.4.3.2. Configurations with two shock waves and two contact discontinuities

The three configurations consist of two contact discontinuities and two shock waves and are used to investigate the mortar interfaces for locally refined meshes in more detail. For each configuration a baseline simulation on the original 100×100 mesh is performed. Afterwards, the mesh is refined in the regions of most interest and the simulations are repeated with the same numerical setup again. The only difference between the setups of these configurations is at which quadrant interface the specific waves are initialize and the direction of travel of the shock wave fronts. Due to the similarity of these examples, the JST indicator with the

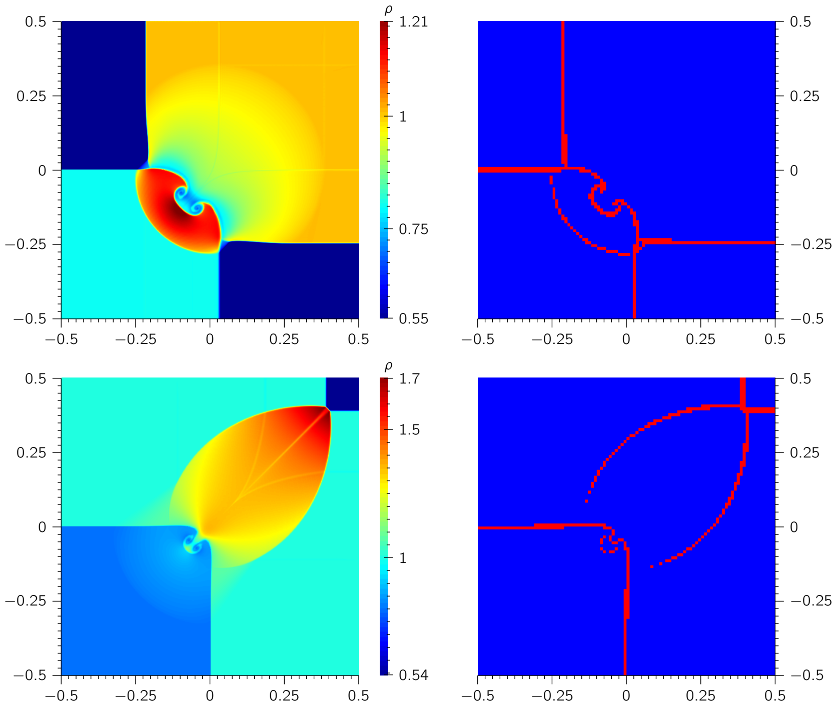


Figure 6.15.: Two dimensional Riemann problem, configurations E and F. Left: Density at final time. Right: FV (red) and DG (blue) elements.

same lower and upper threshold of 0.002 and 0.005 was used. Only configuration F changes the Riemann solver for the boundary conditions from Roe to HLLE.

Configuration E and F

The two shock waves are initialized at the top and right quadrant interface and travel left/down for configuration E and right/up for configuration F. Referring to Schulz-Rinne [78], both examples are symmetric to $y - x = (v_2 - v_1) * t$. Since both initial velocities are zero for configuration F, the contact discontinuities do not move at all in this case. This is also true for the left contact discontinuity of configuration E, but not for bottom wave which moves with $v_2 = 0.1$ to the right and therefore is numerically more challenging for the boundary condition. To circumvent this, the domain of configuration E is extended at the bottom boundary to $y = -1$ with the same mesh size. The first row (E) and second row (F) of figure 6.15 show the numerical results at $t = 0.3$ and $t = 0.25$, respectively. Due to the movement of the bottom contact discontinuity in configuration E, this wave gets a numerical extension, but can be resolved within a single FV sub-cells element. In comparison to this, the left contact discontinuity and both discontinuities of configuration F are located perfectly at an element interface and thus stay perfectly sharp. The shock fronts are resolved within at most two elements in both examples and the numbers of FV sub-cell elements are quite minimal. Only for the non-moving contact discontinuities, which are directly located at the element interfaces, it might not be absolutely necessary to have the adjacent elements marked. This is only possible for this special case where the wave fronts coincide with the mesh, but the JST indicator detects the jump in density and therefore marks this elements in a more general way.

Now the meshes of both cases are refined in the regions where small flow structures occur from $h = \frac{1}{100}$ to $h = \frac{1}{200}$. For both configurations the mesh resolution is doubled in the central subdomain $[-0.2, 0] \times [-0.2, 0]$, which is visualized in figure 6.16 by the dashed rectangles. For the element interfaces lying at the connection between the outer region and the refined part, mortar interfaces are required. For configuration E the choice of the refined rectangle leads to an intersection of the wave fronts with the mortar interfaces under a non-zero and non-orthogonal angle. Comparing the results, it is quite remarkable that the mortar interfaces do not influence the wave fronts at all. Of course, inside the refined region the improved resolution leads to smaller structures in the two developing vortices. First signs of a Kelvin-Helmholtz instability are visible and the minimal density inside the vortices is lower for the refined case.

For configuration F the situation is different. Here, the contact discontinuities and the mortar interfaces are parallel and even lie on one and another. Again, the mortar interfaces do not affect the wave fronts and the doubled resolution mostly changes the small vortices. As for configuration E, the absolute density level inside is lower and a Kelvin-Helmholtz instability starts to develop. It is quite clear that small structures along the shear wave can only be visible if the resolution of the simulation is high enough to resolve them. Hence, for the last configuration J with two shock waves and contact discontinuities each the mesh resolution is increased further to show this effect.

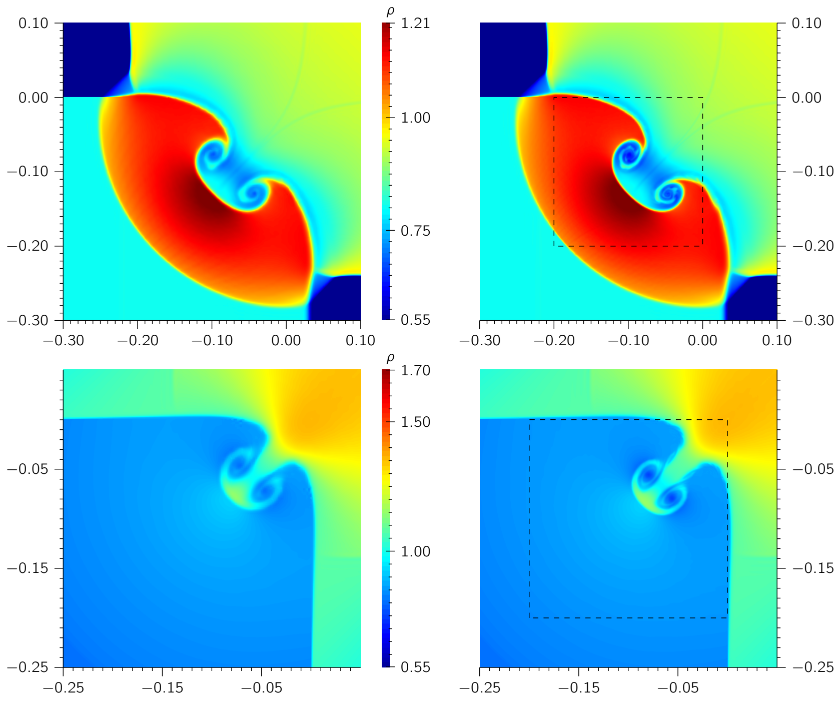


Figure 6.16.: Closeup view of the two dimensional Riemann problem, configurations E and F. Left: Density at final time for the original mesh. Right: Inside the dashed rectangle the mesh resolution is doubled.

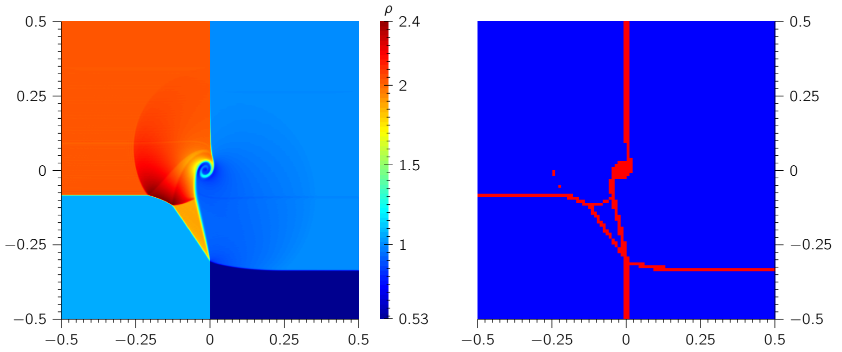


Figure 6.17.: Two dimensional Riemann problem, configuration J. Left: Density at final time. Right: FV (red) and DG (blue) elements.

Configuration J

This configuration is separated into a left and right part by a contact discontinuity. The two shock waves at the left and right quadrant interfaces travel with different speeds downwards, which produces a vortex in the center of the domain. The numerical results at $t = 0.3$ are plotted in the last row of figure 6.17 and are comparable to the results of Kurganov and Tadmor [53]. In this case an additionally Persson indicator with a threshold of -6.0 is used to avoid switching of FV sub-cell elements to DG elements, if they contain high mode oscillations. Increasing the overall mesh resolution from $h = \frac{1}{100}$ to $h = \frac{1}{200}$ and with the mortar interfaces locally in the region of highest interest $[-0.3, -0.1] \times [-0.4, 0.2]$ to even $h = \frac{1}{400}$, the solution shows more small-scale features. In figure 6.18 a close-up view of the center region for both mesh resolutions is plotted. Due to the reduced numerical dissipation, the solution for the fine mesh shows a Kelvin-Helmholtz instability along the shear wave. The same high order behavior was detected by Dumbser et al. [24] at other two dimensional Riemann problems involving shear waves.

6.4.3.3. Configurations with one shock wave, one rarefaction wave and two contact discontinuities

The remaining two dimensional Riemann problems involving shocks are presented only briefly, since the fundamental properties of the hybrid DG/FV sub-cells scheme are here the same as for the previous two dimensional Riemann problems.

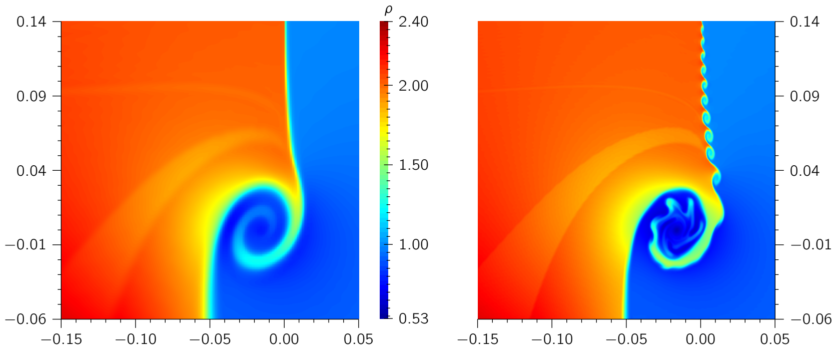


Figure 6.18.: Two dimensional Riemann problem, configuration J. Left: Close-up view of density for $h = \frac{1}{100}$. Right: Close-up view of density for $h = \frac{1}{400}$.

Referring to Schulz-Rinne [77], there are three different configurations consisting of two contact discontinuities and one shock and one rarefaction wave. In configuration G and H the two contact discontinuities are adjacent to each other while in configuration K they are opposite.

Configurations G and H

For both examples the top quadrant interface is initialized with a rarefaction wave and the right one with a shock wave. The other two quadrant interfaces are contact discontinuities. The main difference between the configurations is the direction of travel of the rarefaction and the shock wave. In configuration G these waves move clockwise, while in configuration H they move counter-clockwise. To avoid problems with boundary conditions, the domain has been extended with the same mesh size of $h = \frac{1}{100}$ at the left, bottom and top to $x = -1$ and $y = \pm 1$ respectively. The lower and upper thresholds of the JST indicator are chosen as 0.002 and 0.001. In addition to this indicator the Persson indicator with a threshold of -5.5 is used to avoid switching of FV sub-cell elements to DG elements, if they contain high mode oscillations. In the first (G) and second (H) row of figure 6.19 the numerical solution at $t = 0.2$ is plotted. A comparison to Kurganov and Tadmor [53] or Schulz-Rinne [78] shows a very nice match. Both examples need at most two FV sub-cells elements to resolve the waves.

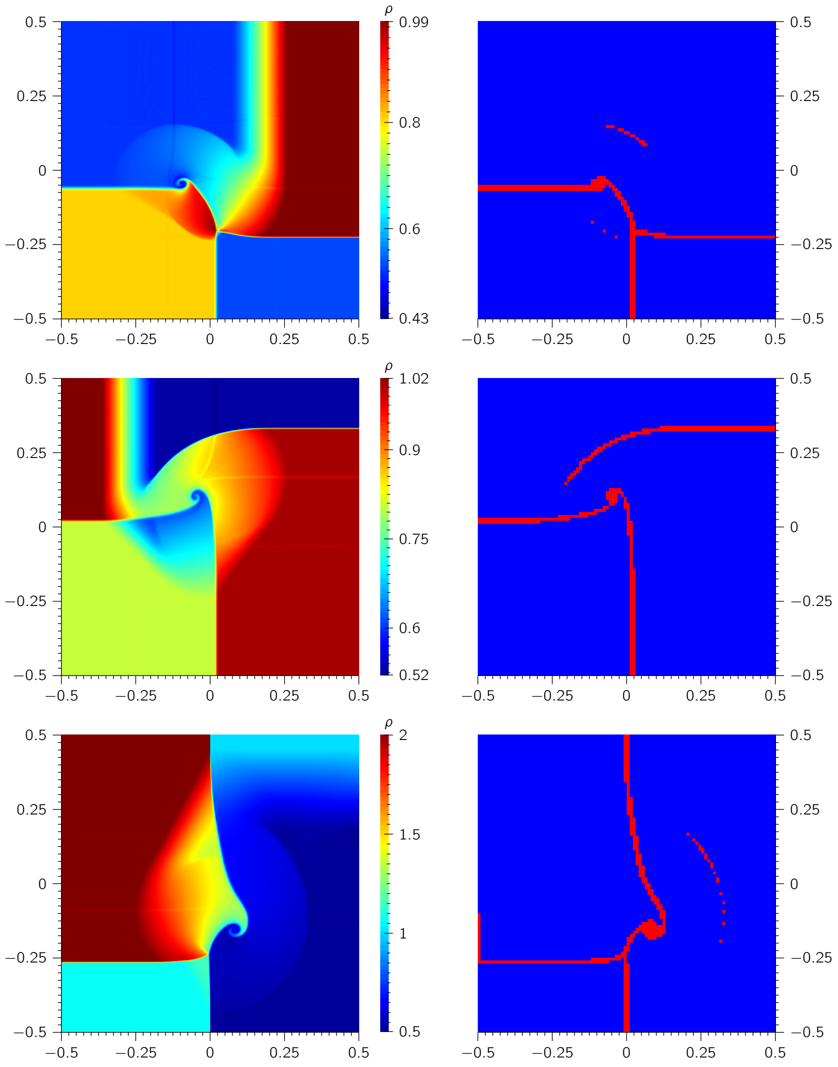


Figure 6.19.: Two dimensional Riemann problem, configurations G, H and K. Left: Density at final time. Right: FV (red) and DG (blue) elements.

Configuration K

This configuration is comparable to configuration J in the sense that the domain is divided by the two slip lines into a left and a right section. At the junction of the two slip lines a vortex forms. To compute this example the JST indicator thresholds are set to 0.001 and 0.004. The numerical solution at final time $t = 0.3$ is given in the last row of figure 6.19. As before, the wave fronts are resolved in at most two FV sub-cell elements. Interesting is the part below the rarefaction wave on the right where a light shock wave occurs. Only a few marked cells are sufficient to resolve this shock.

6.4.4. Shock boundary layer interaction

The last example in this chapter is intended to demonstrate the capabilities of the shock capturing in a real world massively parallel setting. In the research field of aircraft propulsion systems that are capable of generating thrust in the hypersonic flight regime, the supersonic combustion ramjet, short scramjet, is a promising design. A ramjet is an air-breathing jet engine where the compression of the inflowing air is achieved only by the converging inlet at supersonic speed and not due to rotating compressors as in turbo jet engines. The main difference of scramjets compared to ramjets is that the air throughout the entire engine is supersonic, whereas the ramjet slows the incoming flow down to subsonic speeds before the combustion. Therefore, the operation at high Mach numbers requires a profound knowledge of the appearing physical phenomena. One of these phenomena is the interaction of a shock wave with a flat plate turbulent boundary layer, which is investigated in this example with a direct numerical simulation (DNS).

The numerical simulation and the visualizations were performed by Muhammed Atak with a previous revision of the *FLEXI* code with the Gaussian distribution of the FV sub-cells, not published as open source. A flat boundary layer at Mach$=$ 2.67 is impinged by an oblique shock, which has a pressure relation of $p_2/p_1 =$ 1.5. In general, this leads to a non travelling shock which is only moving due to the interaction with the turbulent boundary layer. The computational domain is discretized with a structured hexahedral mesh consisting of 1,125,000 grid cells. The polynomial degree of the Discontinuous Galerkin elements is set to $N = 5$, which results in 6^3 FV sub-cells per DG element or 243 million DOFs in total. Elements containing a shock are detected by the Jameson indicator in combination with the Ducros sensor, which avoids the marking of the turbulent boundary layer by the Jameson indicator. The computation was performed on up to $93,750$ processors.

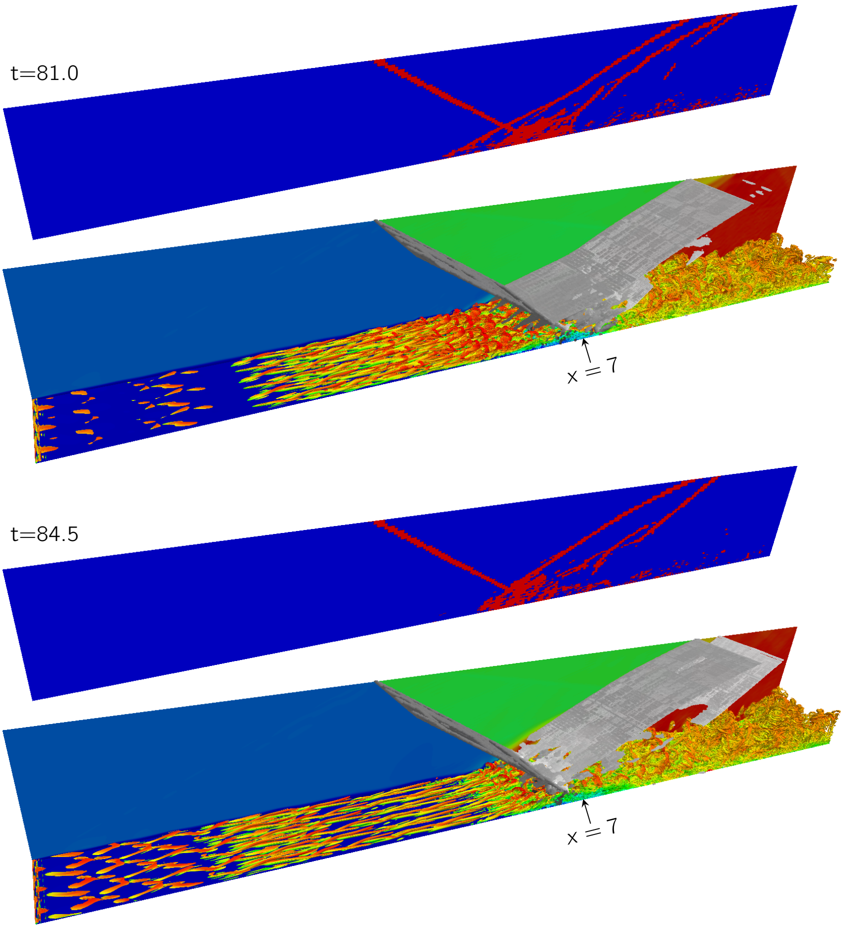


Figure 6.20.: Two snapshots at $t = 81.0s$ and $t = 84.5s$ of the shock boundary layer interaction. Visualized are isocontours of the Lambda-2 vortex criterion, which are colored by the streamwise velocity. The backside plane shows the density and the upper additional backside plane shows the distribution of the DG (blue) and the FV sub-cell (red) elements. In gray, the shock position is visualized with isocontours of zero dilatation.

In figure 6.20, two snapshots at $t = 81.0s$ and $t = 84.5s$ of the computation are shown. The turbulent structures of the boundary layer are visualized with the Lambda-2 vortex criterion, which are additionally colored by the streamwise velocity. With the isocontour of zero dilatation, the shock positions are plotted in gray. The primary shock impinges on the boundary layer at about $x = 7$. Due to the interaction with the boundary layer, two new shocks emerge from the interaction zone. These secondary shocks are not steady like the primary shock front, but dynamically develop from the turbulent structures. This can be seen especially when comparing the distributions of DG (blue) and FV sub-cell (red) elements, which are visualized in the two additional backside planes of figure 6.20. Altogether, the shock capturing using Finite Volume sub-cells works really well for this complex flow situation.

7. Conclusion and prospects

The optimization of lift and drag of an airfoil with respect to its shape is one of the key issues to improve the performance of an airplane by reducing the fuel consumption. For the numerical realization of such a shape optimization, the shape of an object has to be discretized by a finite number of parameters. The optimization procedure then modifies all these parameters until an extremum of the resulting aerodynamic performance values has been found. This requires the evaluation of many flow solutions. Since this can become quite expensive, the number of required computations is the major factor for the costs of an optimization. It even gets worse if the number of solver runs additionally depends on parameters of the discretization. In this case, the number of parameters which define the shape directly multiplies the number of necessary flow solutions. Hence, one limits the set of parameters to a few to keep the numerical cost in acceptable bounds. However, this restricts the freedom of the shape's morphing, which is in contrast to the goal of finding not only a good, but the best design.

To overcome expensive computations for a large set of design parameters this work discusses shape derivatives in Hadamard form. Since their whole derivation takes place in the continuous setting, they are completely independent from the numerical scheme with regards to the number of parameters used to approximate the shape. Nevertheless, the type of the numerical scheme is important in another aspect. As it turns out, the shape derivative depends on the formulation of the governing equation, either in variational or pointwise form. For methods based on the variational formulation, the shape derivative of drag and lift coefficients includes extra terms, which only vanish if the solution fulfills the strong equations. Numerical investigations showed that this leads to a discrepancy of a shape derivative based on the strong form when using the Discontinuous Galerkin method, which solves the Navier–Stokes equations weakly. Especially for flow phenomena where a strong form solution is not clear, a significant difference between both forms is to be expected. This is for example the case for the shock which forms at transonic conditions above an airfoil and leads to a rapid increase of the drag. The shape derivative of the drag could be used to optimize the shape such that this loss in performance is reduced or totally eliminated. However, limitations in the implementation of the Discontinuous Galerkin scheme used for this research prevented such cases. With improvements in the code, the application of the shape derivative should be extended in the future to flow conditions including shocks.

A milestone on the path to this aim is a solid shock capturing, which is provided in the second part of this work. High order methods in general, but also the Discontinuous Galerkin method used here, have the problem that polynomials used to represent the solution are not suitable to approximate discontinuities, since they suffer from the Gibbs phenomenon. Hence, these methods have limitations when it comes to shocks or other flow discontinuities which require a shock capturing. Among several techniques to treat this problem, hybrid methods, where the high order method is chosen for the smooth parts of the flow and the low order scheme handles the shocks, are well suited. Therefore, in this work the Discontinuous Galerkin spectral elements method is coupled to a Finite Volume scheme. Elements containing shocks or other problematic flow solutions are detected by indicator functions. In these marked elements the DG operator is switched to the Finite Volume scheme. With a reconstruction, this method is of second order and due to the usage of slope limiters a perfect candidate for the shock capturing. However, it has substantially higher demands on the mesh resolution than the high order Discontinuous Galerkin method, which uses rather large elements. Therefore, a logical mesh refinement of sub-cells inside the elements of the original grid is introduced in a special manner. For every degree of freedom of a DG element, exactly one Finite Volume sub-cell is generated, which enables to use the same data structures for both methods. This is crucial for an efficient implementation for high performance computations.

The implementation of the shock capturing in the open source code *FLEXI* has been investigated with several numerical examples. One-dimensional test cases provided a good understanding of the basic properties of the scheme and with scaling tests the high parallel efficiency has been verified. Therewith, the method has shown its potential to be applied to more complex and larger problems, like the shock boundary layer interaction which was performed on tens of thousands processors. Furthermore, it has been shown that the shock capturing is freestream preserving on curved grids even with mortar interfaces at local mesh refinements, which is essential for complex geometries. The future of this method should therefore be an application to "real world" cases involving shocks or other flow discontinuities. To come full circle, this is for example the case for airfoils at transonic speed. The shape derivatives derived in the first part of this work can provide the sensitivities needed for a shape optimization. However, they require the implementation of an adjoint solver into the flow solver *FLEXI*, which remains an open task for future research.

A. Parameter files

A.1. Validation examples

A.1.1. Sod shock tube

HOPR parameter file

```
! Each polynomial degree requires a different number of grid cells
! to end up with a comparable load. The number of elements in
! x-direction is given by:
! #elements: XX = 10,12,13,16,18,22,25,33,44

! OUTPUT
ProjectName = SOD_XX               ! Name of output files
Debugvisu = F                      ! no debug visualization

! MESH
Mode      = 1                      ! Mode for Cartesian boxes
nZones    = 1                      ! number of blocks

! BLOCK 1
Corner    = (/ 0.,-1.,-1. ,, 1.,-1.,-1. ,, 1.,1.,-1. ,, 0.,1.,-1.,,
            ↪ 0.,-1., 1. ,, 1.,-1., 1. ,, 1.,1., 1. ,, 0.,1., 1.  /)
nElems    = (/ XX,1,1 /)           ! number of elements in each direction
BCIndex   = (/ 5,3,2,4,1,6 /)      ! Indices of Boundary Conditions
                                   ! (z-,y-,x+,y+,x-,z+)
elemtype  = 108                    ! element type (108: Hexahedral)

vv        = (/ 0.,2.,0. /)         ! vectors connection opposite
vv        = (/ 0.,0.,2. /)         ! periodic BCs

! BOUNDARY CONDITIONS
BoundaryName = BC_x-
BoundaryType = (/ 2,0,0, 0 /)      ! Dirchilet
BoundaryName = BC_x+
BoundaryType = (/ 2,0,0, 0 /)      ! Dirchilet
BoundaryName = BC_y-
BoundaryType = (/ 1,0,0, 1 /)      ! periodic
BoundaryName = BC_y+
BoundaryType = (/ 1,0,0,-1 /)      ! periodic
BoundaryName = BC_z-
BoundaryType = (/ 1,0,0, 2 /)      ! periodic
BoundaryName = BC_z+
BoundaryType = (/ 1,0,0,-2 /)      ! periodic
```

FLEXI parameter file

```
! OUTPUT
ProjectName           = sod

! INTERPOLATION
N                     = 3        ! 4,5,...,11
! the polynomial degree is varied between 3 and 11
! for each polynomial degree the number of elements in the mesh
! file is adjusted accoring to the following table
! N         =   3,  4,  5,  6,  7,  8,  9, 10, 11
! #elements =  44, 33, 26, 22, 18, 16, 13, 12, 10

! MESH
MeshFile              = SOD_XX_mesh.h5 ! XX is #elements
useCurveds            = F

! FV
IndicatorType         = Jameson ! JST indicator
IndVar                = 6       ! pressure
IndStartTime          = 0.00001 ! one time step pure FV
FV_LimiterType        = MinMod
FV_IndUpperThreshold  = 0.015   ! if IndValue above, switch to FV
FV_IndLowerThreshold  = 0.014   ! if IndValue below, switch to DG
FV_toDG_indicator     = T       ! additional Persson for FV -> DG
FV_toDG_limit         = -6.7    ! threshold for additional Persson

! EQUATION
IniExactFunc          = 11      ! shock tube
RefState              = (/ 1.0,   0.,0.,0., 1.0 /)
RefState              = (/ 0.125, 0.,0.,0., 0.1 /)

! RIEMANN
Riemann               = hllc

! TIMEDISC
tend                  = 0.2     ! End time
CFLscale              = 0.8     ! Scaling of theoretical CFL number

! ANALYZE
Analyze_dt            = 0.2     ! Timestep of analyze outputs
CalcErrorNorms        = F       ! Calculate error norms
```

A.1.2. Shu-Osher density fluctuations shock wave interaction problem

HOPR parameter file

```
! OUTPUT
ProjectName = SHUOSHER_100        ! Name of output files
Debugvisu = F                     ! no debug visualization

! MESH
Mode       = 1                    ! Mode for Curved boxes
nZones     = 1                    ! number of blocks

! BLOCK 1
Corner     = (/ -5.,0.,0. ,, 5.,0.,0. ,, 5.,1.,0. ,, -5.,1.,0. ,,
             ↪ -5.,0.,1. ,, 5.,0.,1. ,, 5.,1.,1. ,, -5.,1.,1.   /)
nElems     = (/ 100,1,1 /)        ! number of elements in each direction
BCIndex    = (/ 5,3,2,4,1,6 /)    ! Indices of Boundary Conditions
                                  ! (z-,y-,x+,y+,x-,z+)
elemtype   = 108                  ! element type (108: Hexahedral)

vv         = (/  0.,1.,0. /)      ! vectors connection opposite
vv         = (/  0.,0.,1. /)      ! periodic BCs

! BOUNDARY CONDITIONS
BoundaryName = BC_x-
BoundaryType = (/ 2,0,0, 0 /)     ! Dirichlet
BoundaryName = BC_x+
BoundaryType = (/ 2,0,0, 0 /)     ! Dirichlet
BoundaryName = BC_periodicy-
BoundaryType = (/ 1,0,0, 1 /)     ! periodic
BoundaryName = BC_periodicy+
BoundaryType = (/ 1,0,0,-1 /)     ! periodic
BoundaryName = BC_periodicz-
BoundaryType = (/ 1,0,0, 2 /)     ! periodic
BoundaryName = BC_periodicz+
BoundaryType = (/ 1,0,0,-2 /)     ! periodic
```

FLEXI parameter file

```
! OUTPUT
ProjectName          = shuosher

! INTERPOLATION
N                    = 3        ! Polynomial degree

! MESH
MeshFile             = SHUOSHER_100_mesh.h5
useCurveds           = F

! FV
IndicatorType        = Jameson  ! JST indicator
IndVar               = 6        ! pressure
FV_LimiterType       = MinMod

! For test of varying upper threshold (lower threshold fix)
! performe a simulation for each of the following values:
! XXX = 0.007,0.008,0.009,0.010,0.011,0.012,0.020,0.040,0.080,0.120
FV_IndUpperThreshold = XXX      ! if IndValue above, switch to FV
FV_IndLowerThreshold = 0.005    ! if IndValue below, switch to DG

! For test of varying lower threshold (upper threshold fix)
! performe a simulation for each of the following values:
! XXX = 0.001,0.002,0.003,0.004,0.005,0.006,0.007,0.008
FV_IndUpperThreshold = 0.011    ! if IndValue above, switch to FV
FV_IndLowerThreshold = XXX      ! if IndValue below, switch to DG

! EQUATION
IniExactFunc         = 12       ! shu osher example

! RIEMANN
Riemann              = Roe

! TIMEDISC
tend                 = 1.8      ! End time
CFLscale             = 0.9      ! Scaling of theoretical CFL number

! ANALYZE
Analyze_dt           = 1.8      ! Timestep of analyze outputs
CalcErrorNorms       = F        ! Calculate error norms
```

A.1.3. Freestream preservation

HOPR parameter file

```
! OUTPUT
ProjectName = FREESTREAM          ! Name of output files
Debugvisu = F

! MESH
Mode        = 1                   ! Mode for Cartesian boxes
nZones      = 2                   ! number of blocks

! BLOCK 1
Corner      = (/ -1.,-1.,-1.,, 3.,-1.,-1.,, 3.,3.,-1.,, -1.,3.,-1.,,
            ↪ -1.,-1., 1., ,3.,-1., 1.,, 3.,3., 1.,, -1.,3., 1. /)
nElems      = (/ 4,4,2 /)         ! number of elements in each direction
BCIndex     = (/ 5,3,2,4,1,0 /)   ! Indices of Boundary Conditions
                                  ! (z-,y-,x+,y+,x-,z+)
elemtype    =108                  ! element type (108: Hexahedral)

! BLOCK 2
Corner      = (/ -1.,-1.,1.,, 3.,-1.,1.,, 3.,3.,1.,, -1.,3.,1.,,
            ↪ -1.,-1.,3.,, 3.,-1.,3.,, 3.,3.,3.,, -1.,3.,3. /)
NElems      = (/ 8,8,2 /)         ! number of elements in each direction
BCIndex     = (/ 0,3,2,4,1,6 /)   ! Indices of Boundary Conditions
                                  ! (z-,y-,x+,y+,x-,z+)
elemtype    = 108                 ! element type (108: Hexahedral)

vv          = (/ 2.,0.,0. /)      ! vectors connection opposite
vv          = (/ 0.,2.,0. /)      ! periodic BCs
vv          = (/ 0.,0.,2. /)      !

MeshScale        = 0.5            ! scale mesh by this factor
MeshPostDeform   = 31             ! x=x+0.1*sin(PI*x)*sin(PI*y)*sin(PI*z)
useCurveds       = T
BoundaryOrder    = 4              ! polynomial degree of geometry

! BOUNDARY CONDITIONS
BoundaryName = BC_x-
BoundaryType = (/ 1,0,0, 1 /)     ! periodic
BoundaryName = BC_x+
BoundaryType = (/ 1,0,0,-1 /)     ! periodic
BoundaryName = BC_y-
BoundaryType = (/ 1,0,0, 2 /)     ! periodic
BoundaryName = BC_y+
BoundaryType = (/ 1,0,0,-2 /)     ! periodic
BoundaryName = BC_z-
BoundaryType = (/ 1,0,0, 3 /)     ! periodic
BoundaryName = BC_z+
BoundaryType = (/ 1,0,0,-3 /)     ! periodic
```

FLEXI parameter file

```
! OUTPUT
ProjectName     = freestream

! INTERPOLATION
N               = 6                 ! polynomial degree (2*NGeo)
NAnalyze        = 10                ! Number of analyze points

! MESH
MeshFile        = FREESTREAM_mesh.h5
useCurveds      = T

! FV
IndicatorType   = checkerboard
FV_LimiterType  = minmod

! EQUATION
IniExactFunc    = 1                 ! constant RefState
IniRefState     = 1
RefState        = (/ 1.0,1.0,1.0,1.0,1.0 /)
Riemann         = Roe

! TIMEDISC
tend            = 0.5               ! End time
CFLscale        = 0.9               ! Scaling of theoretical CFL number

! ANALYZE
CalcErrorNorms  = T                 ! Calculate error norms
CalcBodyForces  = F                 ! Calculate body forces
```

A.2. Order of convergence

HOPR parameter file

```
! Grid convergence is investigated for a periodic cube, where the upper
! half in z-direction has twice the resolution in the x and y-direction
! The number of baseline number of elements per direction is varied:
! #elements: XX = 8,12,16,20,24

! OUTPUT
ProjectName = CONV_XX              ! Name of output files
Debugvisu = F                      ! no debug visualization

! MESH
Mode        = 1                    ! Mode for Cartesian boxes
nZones      = 2                    ! number of blocks

! BLOCK 1
Corner      = (/ -1.,-1.,-1.  ,, 1.,-1.,-1.  ,, 1.,1.,-1.  ,, -1.,1.,-1.,,
            ↪ -1.,-1., 0.  ,, 1.,-1., 0.  ,, 1.,1., 0.  ,, -1.,1., 0.  /)
nElems      = (/ XX,XX,XX/2 /)     ! number of elements in each direction
BCIndex     = (/ 5,3,2,4,1,0 /)    ! Indices of Boundary Conditions
                                   ! (z-,y-,x+,y+,x-,z+)
elemtype    = 108                  ! element type (108: Hexahedral)

! BLOCK 2 (refined in x- and y-direction)
Corner      = (/ -1.,-1., 0.  ,, 1.,-1., 0.  ,, 1.,1., 0.  ,, -1.,1., 0.,,
            ↪ -1.,-1., 1.  ,, 1.,-1., 1.  ,, 1.,1., 1.  ,, -1.,1., 1.  /)
NElems      = (/ XX*2,XX*2,XX/2 /)! number of elements in each direction
BCIndex     = (/ 0,3,2,4,1,6 /)    ! Indices of Boundary Conditions
                                   ! (z-,y-,x+,y+,x-,z+)
elemtype    = 108                  ! element type (108: Hexahedral)

vv          = (/ 2.,0.,0. /)       ! vectors connection opposite
vv          = (/ 0.,2.,0. /)       ! periodic BCs
vv          = (/ 0.,0.,2. /)       !

! BOUNDARY CONDITIONS
BoundaryName = BC_x-
BoundaryType = (/ 1,0,0, 1 /)      ! periodic
BoundaryName = BC_x+
BoundaryType = (/ 1,0,0,-1 /)      ! periodic
BoundaryName = BC_y-
BoundaryType = (/ 1,0,0, 2 /)      ! periodic
BoundaryName = BC_y+
BoundaryType = (/ 1,0,0,-2 /)      ! periodic
BoundaryName = BC_z-
BoundaryType = (/ 1,0,0, 3 /)      ! periodic
BoundaryName = BC_z+
BoundaryType = (/ 1,0,0,-3 /)      ! periodic
```

FLEXI parameter file

```
! OUTPUT
ProjectName     = convtest

! INTERPOLATION
N               = 2  ! 2,...,5
! the polynomial degree is varied between 2 and 5
! for each polynomial degree the computation is performed
! on different grids, where the baseline number of elements
! per direction is:
! #elements = 8,12,16,20,24
NAnalyze        = 10                ! Number of analyze points

! MESH
MeshFile        = CONV_XX_mesh.h5 ! XX = #elements
useCurveds      = F

! FV
! the convergence is investigated for pure DG and FV with
! different limiters:
! LimiterFunction: YY = none, minmod, central
IndicatorType   = DG                ! DG,FV
FV_LimiterType  = YY                ! YY = LimiterFunction
IndVar          = 1                 ! density

! EQUATION
IniExactFunc    = 2                 ! diagonal density wave
IniRefState     = 1
RefState        = (/ 1.0,0.3,0.0,0.0,0.71428571 /)
AdvVel          = (/ 1.0,1.0,1.0 /)
Riemann         = Roe

! TIMEDISC
tend            = 0.5               ! End time
CFLscale        = 0.8               ! Scaling of theoretical CFL number

! ANALYZE
CalcErrorNorms  = T                 ! Calculate error norms
CalcBodyForces  = F                 ! Calculate body forces
```

A.3. Parallel efficiency

Compile flags

```
-DEQNSYSNR=2 -DFV_ENABLED=1 -DFV_RECONSTRUCT=1
-DH5DIFF=\"/opt/cray/hdf5/1.10.0.1/bin/h5diff\" -DLUSTRE -DPARABOLIC=1
-DPP_Lifting=1 -DPP_N=N -DPP_NodeType=1 -DPP_VISC=0 -DPP_nVar=5
-DPP_nVarPrim=6 -DUSE_MPI=1 -fdefault-real-8 -fdefault-double-8
-fbackslash -ffree-line-length-0 -DGNU -O3 -march=core-avx2
-finline-functions -fstack-arrays -Jinclude -xf95-cpp-input -fPIC
```

HOPR parameter file

```
! Parallel efficiency is investigated for different problems sizes.
! Therefore in total 12 different meshes are generated.
! The smallest configuration consists of 6x6x6 = 216 elements.
! All other cases are build by doubling the number of elements of
! the previous case.
! This results in the following configurations, where XXX,YYY,ZZZ
! are the number of elements in the respective directions.
! Case   1  2  3  4  5  6  7  8  9 10 11 12
!------------------------------------------
! XXX    6 12 12 12 24 24 24 48 48 48 96 96
! YYY    6  6 12 12 12 24 24 24 48 48 48 96
! ZZZ    6  6  6 12 12 12 24 24 24 48 48 48

! OUTPUT
ProjectName = XXXxYYYxZZZ        ! Name of output files
Debugvisu = F                    ! no debug visualization

! MESH
Mode      = 1                    ! Mode for Cartesian boxes
nZones    = 1                    ! number of blocks

! BLOCK 1
Corner    = (/ -6.,0.,0.,, 6.,0.,0.,, 6.,8.,0.,, -6,,8.,0.,,
            ↪ -6.,0.,8.,, 6.,0.,8.,, 6.,8.,8.,, -6.,8.,8.  /)
nElems    = (/ XXX,YYY,ZZZ /)    ! number of elements in each direction
BCIndex   = (/ 1,2,3,4,5,6 /)    ! Indices of Boundary Conditions
                                 ! (z-,y-,x+,y+,x-,z+)
elemtype  = 108                  ! element type (108: Hexahedral)

! BOUNDARY CONDITIONS
BoundaryName = BC_z-
BoundaryType = (/ 2,0,0,0 /)     ! Dirichlet
BoundaryName = BC_y-
BoundaryType = (/ 2,0,0,0 /)     ! Dirichlet
BoundaryName = BC_x+
BoundaryType = (/ 2,0,0,0 /)     ! Dirichlet
BoundaryName = BC_y+
BoundaryType = (/ 2,0,0,0 /)     ! Dirichlet
BoundaryName = BC_x-
BoundaryType = (/ 2,0,0,0 /)     ! Dirichlet
BoundaryName = BC_z+
BoundaryType = (/ 2,0,0,0 /)     ! Dirichlet
```

FLEXI parameter file

```
! OUTPUT
ProjectName      = cartbox
WriteStateFiles = F

! INTERPOLATION
N                = 6
doWeakLifting    = T

! MESH
MeshFile         = ../XXXxYYYxZZZ_mesh.h5

! FV
! To compare the DG and the FV sub-cells method and furthermore
! mixed DG/FV interfaces the indicator is set to one of the
! following four synthetic indicator functions
! III = DG, FV, checkerboard, halfhalf
IndicatorType      = III
FV_LimiterType     = MinMod
FV_IniSupersample = F             ! do not supersample initial solution

! Riemann
Riemann          = LF

! EQUATION
IniExactFunc    = 1               ! Freestream
IniRefState     = 1
RefState        = (/1.,1.,1.,1.,1./)
mu0             = 0.000018547     ! physical constants
R               = 1.0             !
Pr              = 0.72            !
kappa           = 1.4             !

! TIMEDISC
TimeDiscMethod = carpenterrk4-5 ! Runge-Kutta scheme
tend            = 500.0           ! big number, the simulation
Analyze_dt      = 500.0           ! is terminated by:
maxIter         = 100             ! maximal number of timesteps
CFLscale        = 0.99            ! Scaling of theoretical CFL
DFLscale        = 0.4             ! Scaling of theoretical DFL
```

A.4. Complex examples

A.4.1. Double Mach reflection

HOPR parameter file

```
! OUTPUT
! The Double Mach Reflection example is performed on grids,
! with different resolutions :
! #elements for physical length of 1.0 : XX = 30, 120, 240,
ProjectName = DMR_XX               ! Name of output files
Debugvisu = F                      ! no debug visualization

! MESH
Mode       = 1                     ! Mode for Cartesian boxes
nZones     = 2                     ! number of blocks

! BLOCK 1
Corner     = (/ 0.,0.,-6. ,, 1.,0.,-6. ,, 1.,6.,-6. ,, 0.,6.,-6. ,,
           ↪ 0.,0., 6. ,, 1.,0., 6. ,, 1.,6., 6. ,, 0.,6., 6.   /)
nElems     = (/ XX/6,XX,1 /)       ! number of elements in each direction
BCIndex    = (/ 1,2,0,4,5,6 /)     ! Indices of Boundary Conditions
                                   ! (z-,y-,x+,y+,x-,z+)
elemtype   = 108                   ! element type (108: Hexahedral)

! BLOCK 2
Corner     = (/ 1.,0.,-6. ,, 24.,0.,-6. ,, 24.,6.,-6. ,, 1.,6.,-6. ,,
           ↪ 1.,0., 6. ,, 24.,0., 6. ,, 24.,6., 6. ,, 1.,6., 6.   /)
NElems     = (/ XX*23/6,XX,1 /)    ! number of elements in each direction
BCIndex    = (/ 1,7,3,4,0,6 /)     ! Indices of Boundary Conditions
                                   ! (z-,y-,x+,y+,x-,z+)
elemtype   = 108                   ! element type (108: Hexahedral)

vv         = (/ 0.,0.,12. /)       ! vector connection opposite
                                   ! periodic BCs
postScaleMesh  = T                 ! rescale mesh after building of mesh
meshScale      = 0.1666666666666   ! to 1/6

! BOUNDARY CONDITIONS
BoundaryName = BC_z-
BoundaryType = (/ 1,0,0, 1 /)      ! periodic
BoundaryName = BC_y-
BoundaryType = (/  2,0,0, 0 /)     ! periodic
BoundaryName = BC_x+
BoundaryType = (/24,0,2, 0 /)      ! pressure ouflow
BoundaryName = BC_y+
BoundaryType = (/ 2,0,0, 0 /)      ! Dirichlet
BoundaryName = BC_x-
BoundaryType = (/ 2,0,0, 0 /)      ! Dirichlet
BoundaryName = BC_z+
BoundaryType = (/ 1,0,0,-1 /)      ! periodic
BoundaryName = BC_wall
BoundaryType = (/ 9,0,0, 0 /)      ! Euler slip wall
```

A.4.1.1. Pure Finite Volume on a fine grid

FLEXI parameter file

```
! OUTPUT
ProjectName             = dmr_purefv

! INTERPOLATION
N                       = 5         ! Polynomial degree

! MESH
MeshFile                = DMR_240_mesh.h5
useCurveds              = F

! FV
IndicatorType           = FV
IndVar                  = 6         ! pressure
FV_LimiterType          = MinMod
FV_IndUpperThreshold    = 0.010     ! if IndValue above, switch to FV
FV_IndLowerThreshold    = 0.005     ! if IndValue below, switch to DG

! EQUATION
IniExactFunc            = 13
RefState                = (/ 8.0,7.14471,-4.125,0.,116.5 /) ! pre shock
RefState                = (/ 1.4,0.     ,0.     ,0.,1.0   /) ! post shock
Riemann                 = HLLE

! TIMEDISC
tend                    = 0.2       ! End time
CFLscale                = 0.9       ! Scaling of theoretical CFL number

! ANALYZE
Analyze_dt              = 0.1       ! Timestep of analyze outputs
CalcErrorNorms          = F         ! Calculate error norms
```

A.4.1.2. Hybrid DG/FV sub-cell simulation on a coarse grid comparative to pure FV on a fine grid

FLEXI parameter file

```
! OUTPUT
ProjectName             = dmr_coarse

! INTERPOLATION
N                       = 5         ! Polynomial degree

! MESH
MeshFile                = DMR_30_mesh.h5
useCurveds              = F

! FV
IndicatorType           = Jameson ! JST indicator
IndVar                  = 6         ! pressure
```

```
FV_LimiterType         = MinMod
FV_IndUpperThreshold = 0.010    ! if IndValue above, switch to FV
FV_IndLowerThreshold = 0.005    ! if IndValue below, switch to DG
FV_toDG_indicator      = T        ! additional Persson for FV -> DG
FV_toDG_limit          = -5.5     ! threshold for additional Persson

! EQUATION
IniExactFunc           = 13
RefState               = (/ 8.0,7.14471,-4.125,0.,116.5 /) ! pre shock
RefState               = (/ 1.4,0.      ,0.     ,0.,1.0   /) ! post shock
Riemann                = HLLE

! TIMEDISC
tend                   = 0.2      ! End time
CFLscale               = 0.9      ! Scaling of theoretical CFL number

! ANALYZE
Analyze_dt             = 0.2      ! Timestep of analyze outputs
CalcErrorNorms         = F        ! Calculate error norms
```

A.4.1.3. Investigation of the numerical dissipation introduced by the FV sub-cells

FLEXI parameter file

```
! OUTPUT
! To show the influence of the numerical dissipation introduced
! by the FV sub-cells the amount of FV sub-cells is increased
! by adjusting the upper and lower thresholds.
! Two configurations are investigated:
! upper,lower threshold: XXX,YYY = 0.010, 0.005
! upper,lower threshold: XXX,YYY = 0.004, 0.003
ProjectName            = dmr_thresholds_XXX_YYY

! INTERPOLATION
N                      = 5        ! Polynomial degree

! MESH
MeshFile               = DMR_120_mesh.h5
useCurveds             = F

! FV
IndicatorType          = Jameson ! JST indicator
IndVar                 = 6        ! pressure
FV_LimiterType         = MinMod
FV_IndUpperThreshold = XXX      ! if IndValue above, switch to FV
FV_IndLowerThreshold = YYY      ! if IndValue below, switch to DG
FV_toDG_indicator      = T        ! additional Persson for FV -> DG
FV_toDG_limit          = -5.5     ! threshold for additional Persson

! EQUATION
IniExactFunc           = 13
RefState               = (/ 8.0,7.14471,-4.125,0.,116.5 /) ! pre shock
RefState           = (/ 1.4,0.     ,0.    ,0.,1.0    /) ! post shock
Riemann            = HLLE

! TIMEDISC
tend               = 0.2     ! End time
CFLscale           = 0.9     ! Scaling of theoretical CFL number

! ANALYZE
Analyze_dt         = 0.2     ! Timestep of analyze outputs
CalcErrorNorms     = F       ! Calculate error norms
```

A.4.2. Forward facing step

HOPR parameter file

```
! OUTPUT
ProjectName = FFS               ! Name of output files
Debugvisu = F                   ! no debug visualization

! MESH
Mode       = 1                  ! Mode for Cartesian boxes
nZones     = 3                  ! number of blocks

! BLOCK 1
Corner     = (/ 0.,0.,0. ,, 0.6,0.,0. ,, 0.6,0.2,0. ,, 0.,0.2,0. ,,
             ↪ 0.,0.,1. ,, 0.6,0.,1. ,, 0.6,0.2,1. ,, 0.,0.2,1.   /)
nElems     = (/ 60,20,1 /)      ! number of elements in each direction
BCIndex    = (/ 1,2,2,0,5,6 /)  ! Indices of Boundary Conditions
                                ! (z-,y-,x+,y+,x-,z+)
elemtype   = 108                ! element type (108: Hexahedral)

! BLOCK 2
Corner     = (/ 0.,0.2,0. ,, 0.6,0.2,0. ,, 0.6,1.,0. ,, 0.,1.,0. ,,
             ↪ 0.,0.2,1. ,, 0.6,0.2,1. ,, 0.6,1.,1. ,, 0.,1.,1.   /)
nElems     = (/ 60,80,1 /)      ! number of elements in each direction
BCIndex    = (/ 1,0,0,4,5,6 /)  ! Indices of Boundary Conditions
                                ! (z-,y-,x+,y+,x-,z+)
elemtype   = 108                ! element type (108: Hexahedral)

! BLOCK 3
Corner     = (/ 0.6,0.2,0. ,, 3.,0.2,0. ,, 3.,1.,0. ,, 0.6,1.,0. ,,
             ↪ 0.6,0.2,1. ,, 3.,0.2,1. ,, 3.,1.,1. ,, 0.6,1.,1.   /)
nElems     = (/ 240,80,1 /)     ! number of elements in each direction
BCIndex    = (/ 1,2,3,4,0,6 /)  ! Indices of Boundary Conditions
                                ! (z-,y-,x+,y+,x-,z+)
elemtype   = 108                ! element type (108: Hexahedral)

vv         = (/ 0.,0.,1. /)     ! vector connection opposite
                                ! periodic BCs
! BOUNDARY CONDITIONS
BoundaryName = BC_z-
BoundaryType = (/ 1,0,0, 1 /)   ! periodic
BoundaryName = BC_wall
```

```
BoundaryType = (/ 9,0,1, 0 /)    ! Euler slip wall
BoundaryName = BC_outflow
BoundaryType = (/24,0,1, 0 /)    ! pressure outflow
BoundaryName = BC_symmetry
BoundaryType = (/ 9,0,1, 0 /)    ! Euler slip wall
BoundaryName = BC_inflow
BoundaryType = (/ 2,0,1, 0 /)    ! Dirichlet
BoundaryName = BC_z+
BoundaryType = (/ 1,0,0,-1 /)    ! periodic
```

FLEXI parameter file

```
! OUTPUT
! With the Forward Facing Step example the influence of the
! variable the indicator is evaluated for is investigated.
! The Persson indicator is evaluated for:
! indicator variable: XXX = 1, 6  ! 1: density, 6: pressure
ProjectName          = ffs_persson_XXX

! INTERPOLATION
N                    = 5         ! Polynomial degree

! MESH
MeshFile             = FFS_mesh.h5
useCurveds           = F

! FV
IndicatorType        = Persson
IndVar               = XXX       ! 1: density, 6: pressure
FV_LimiterType       = MinMod
IndStartTime         = 0.001
FV_IndLowerThreshold = -6.5      ! if IndValue above, switch to FV
FV_IndUpperThreshold = -5.5      ! if IndValue below, switch to DG

! EQUATION
IniExactFunc         = 1
IniRefState          = 1
RefState             = (/ 1.4,3.,0.,0.,1.0 /)
Riemann              = HLLE

! TIMEDISC
tend                 = 4.0       ! End time
CFLscale             = 0.9       ! Scaling of theoretical CFL number

! ANALYZE
Analyze_dt           = 1.0       ! Timestep of analyze outputs
CalcErrorNorms       = F         ! Calculate error norms
```

A.4.3. Two dimensional Riemann problem

HOPR parameter file

```
! For the different two dimensional Riemann problems
! indiviual meshes are build:
!   XXX = Configuration
! The HOPR parameter file for all configurations is
! the same, except the Block definitions.
! They will be listed for the configurations indiviually.
! OUTPUT
ProjectName = riemann2d_cfgXXX  ! Name of output files
Debugvisu = F                   ! no debug visualization

! MESH
Mode      = 1                   ! Mode for Cartesian boxes

! BLOCK 1
! ... insert definition of block 1 here
! BLOCK 2
! ... insert definition of block 2 here
! ...
! ...

vv        = (/ 0.,0.,1. /)      ! vector connection opposite
                                ! periodic BCs
BoundaryName = BC_2
BoundaryType = (/-101,0,2, 0 /) ! special Riemann2D BC
BoundaryName = BC_4
BoundaryType = (/-101,0,4, 0 /) ! special Riemann2D BC
BoundaryName = BC_3
BoundaryType = (/-101,0,3, 0 /) ! special Riemann2D BC
BoundaryName = BC_1
BoundaryType = (/-101,0,1, 0 /) ! special Riemann2D BC
BoundaryName = BC_zminus
BoundaryType = (/   1,0,0, 1 /) ! periodic
BoundaryName = BC_zplus
BoundaryType = (/   1,0,0,-1 /) ! periodic
```

FLEXI parameter file

```
! For the different two dimensional Riemann problems
! this basis parameter file is used:
!   XXX = Configuration
! OUTPUT
ProjectName             = riemann2d_cfgXXX

! INTERPOLATION
N                       = 5

! MESH
MeshFile                = riemann2d_cfgXXX_mesh.h5
useCurveds              = F
```

```
! FV
IndicatorType         = Jameson ! JST indicator
IndVar                = 1       ! density
FV_LimiterType        = MinMod
! The upper and lower thresholds vary for the different
! configurations and are given in the following table:
! Configuration          |   3   |   4   |   6   | E,F,J | G,H   |   K
! Configuration          |   3   |   4   |   6   | 11-13 | 15-16 |   19
! upper threshold: YYY | 0.009 | 0.005 | 0.01  | 0.005 | 0.002 | 0.004
! lower threshold: ZZZ | 0.004 | 0.002 | 0.004 | 0.002 | 0.001 | 0.001
FV_IndUpperThreshold = YYY
FV_IndLowerThreshold = ZZZ
FV_toDG_indicator     = T       ! for Cfg 3,6,13,15,16
                                ! all other Cfgs set it to 'F'
FV_toDG_limit         =         ! -5.5 for Cfg 15,16.
                                ! -6.0 fog Cfg 3,6,13

! EQUATION
IniExactFunc          = -XXX

! RIEMANN
Riemann               = Roe     ! except CFG 3, which uses HLLE
RiemannBC             =         ! HLLE for Cfg 3,4,6,12
                                ! Roe  for all other Cfgs
! TIMEDISC
! The end time varies for the different configurations:
! Configuration |  3  |  4   |  6  | 11  |  12  | 13  | 15-16 | 19
! End time      | 0.3 | 0.25 | 0.2 | 0.3 | 0.25 | 0.3 |  0.2  | 0.3
tend                  =
CFLscale              = 0.9     ! Scaling of theoretical CFL number

! ANALYZE
Analyze_dt            = 0.3     ! Timestep of analyze outputs
CalcErrorNorms        = F       ! Calculate error norms
```

A.4.3.1. Configurations without contact discontinuities

Configuration 3

HOPR parameter file (Double resolution in $[-0.3, 0] \times [-0.3, 0]$)

```
nZones     = 5                    ! number of blocks

! BLOCK 1
Corner     = (/ -0.3,-0.3,0,, 0.0,-0.3,0,, 0.0,0.0,0,,-0.3,0.0,0,,
             ↪ -0.3,-0.3,1,, 0.0,-0.3,1,, 0.0,0.0,1,,-0.3,0.0,1  /)
nElems     = (/ 60,60,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,0,0,0,6 /)

! BLOCK 2
Corner     = (/ -0.5,-0.5,0,,-0.3,-0.5,0,,-0.3,0.0,0,,-0.5,0.0,0,,
             ↪ -0.5,-0.5,1,,-0.3,-0.5,1,,-0.3,0.0,1,,-0.5,0.0,1  /)
```

```
nElems     = (/ 20,50,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,0,0,1,6 /)

! BLOCK 3
Corner     = (/ -0.3,-0.5,0,,0.5,-0.5,0,,0.5,-0.3,0,,-0.3,-0.3,0,,
             ↪ -0.3,-0.5,1,,0.5,-0.5,1,,0.5,-0.3,1,,-0.3,-0.3,1  /)
nElems     = (/ 80,20,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,2,0,0,6 /)

! BLOCK 4
Corner     = (/ 0.0,-0.3,0,, 0.5,-0.3,0,, 0.5,0.5,0,, 0.0,0.5,0,,
             ↪ 0.0,-0.3,1,, 0.5,-0.3,1,, 0.5,0.5,1,, 0.0,0.5,1  /)
nElems     = (/ 50,80,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,2,4,0,6 /)

! BLOCK 5
Corner     = (/ -0.5,0.0,0,, 0.0,0.0,0,, 0.0,0.5,0,, -0.5,0.5,0,,
             ↪ -0.5,0.0,1,, 0.0,0.0,1,, 0.0,0.5,1,, -0.5,0.5,1  /)
nElems     = (/ 50,50,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,0,4,1,6 /)
```

Configuration 4

HOPR parameter file

```
nZones     = 1                    ! number of blocks

! BLOCK 1
Corner     = (/ -0.5,-0.5,0,, 0.5,-0.5,0,, 0.5,0.5,0,, -0.5,0.5,0,,
             ↪ -0.5,-0.5,1,, 0.5,-0.5,1,, 0.5,0.5,1,, -0.5,0.5,1  /)
nElems     = (/ 100,100,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,2,4,1,6 /)
```

Configuration 6

HOPR parameter file

```
nZones     = 1                    ! number of blocks

! BLOCK 1
Corner     = (/ -0.5,-1.,0 ,, 0.5,-1.,0 ,, 0.5, 1.,0 ,, -0.5, 1.,0 ,,
             ↪ -0.5,-1.,1 ,, 0.5,-1.,1 ,, 0.5, 1.,1 ,, -0.5, 1.,1  /)
nElems     = (/ 100,200,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,2,4,1,6 /)
```

A.4.3.2. Configurations with two shock waves and two contact discontinuities

Configuration E

HOPR parameter file

```
nZones     = 1                      ! number of blocks

! BLOCK 1
Corner     = (/ -0.5,-1.,0  ,, 0.5,-1.,0  ,, 0.5,0.5,0  ,, -0.5,0.5,0  ,,
             ↪ -0.5,-1.,1  ,, 0.5,-1.,1  ,, 0.5,0.5,1  ,, -0.5,0.5,1  /)
nElems     = (/ 100,150,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,2,4,1,6 /)
```

HOPR parameter file (Double resolution in $[-0.2, 0] \times [-0.2, 0]$)

```
nZones     = 6                      ! number of blocks

! BLOCK 1
Corner     = (/ -0.5,-0.5,0,, -0.2,-0.5,0,, -0.2,0.0,0,, -0.5,0.0,0,,
             ↪ -0.5,-0.5,1,, -0.2,-0.5,1,, -0.2,0.0,1,, -0.5,0.0,1  /)
nElems     = (/ 30,50,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,0,0,1,6 /)

! BLOCK 2
Corner     = (/ -0.2,-0.5,0,, 0.5,-0.5,0,, 0.5,-0.2,0,, -0.2,-0.2,0,,
             ↪ -0.2,-0.5,1,, 0.5,-0.5,1,, 0.5,-0.2,1,, -0.2,-0.2,1  /)
nElems     = (/ 70,30,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,2,0,0,6 /)

! BLOCK 3
Corner     = (/ 0.0,-0.2,0,, 0.5,-0.2,0,, 0.5,0.5,0,, 0.0,0.5,0,,
             ↪ 0.0,-0.2,1,, 0.5,-0.2,1,, 0.5,0.5,1,, 0.0,0.5,1  /)
nElems     = (/ 50,70,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,2,4,0,6 /)

! BLOCK 4
Corner     = (/ -0.5,0.0,0,, 0.0,0.0,0,, 0.0,0.5,0,, -0.5,0.5,0,,
             ↪ -0.5,0.0,1,, 0.0,0.0,1,, 0.0,0.5,1,, -0.5,0.5,1  /)
nElems     = (/ 50,50,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,0,4,1,6 /)

! BLOCK 5
Corner     = (/ -0.2,-0.2,0,, 0.0,-0.2,0,, 0.0,0.0,0,, -0.2,0.0,0,,
             ↪ -0.2,-0.2,1,, 0.0,-0.2,1,, 0.0,0.0,1,, -0.2,0.0,1  /)
nElems     = (/ 40,40,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,0,0,0,6 /)
```

```
! BLOCK 6
Corner    = (/ -0.5,-1.0,0,, 0.5,-1.0,0,, 0.5,-0.5,0,, -0.5,-0.5,0,,
            ↪ -0.5,-1.0,1,, 0.5,-1.0,1,, 0.5,-0.5,1,, -0.5,-0.5,1  /)
nElems    = (/ 100,50,1 /)
elemtype  = 108
BCIndex   = (/ 5,3,2,0,1,6 /)
```

Configuration F

HOPR parameter file

Same as for configuration 4 in appendix A.4.3.1.

HOPR parameter file (Double resolution in $[-0.2, 0] \times [-0.2, 0]$)

```
nZones    = 5                     ! number of blocks

! BLOCK 1
Corner    = (/ -0.5,-0.5,0,, -0.2,-0.5,0,, -0.2,0.0,0,, -0.5,0.0,0,,
            ↪ -0.5,-0.5,1,, -0.2,-0.5,1,, -0.2,0.0,1,, -0.5,0.0,1  /)
nElems    = (/ 30,50,1 /)
elemtype  = 108
BCIndex   = (/ 5,3,0,0,1,6 /)

! BLOCK 2
Corner    = (/ -0.2,-0.5,0,, 0.5,-0.5,0,, 0.5,-0.2,0,, -0.2,-0.2,0,,
            ↪ -0.2,-0.5,1,, 0.5,-0.5,1,, 0.5,-0.2,1,, -0.2,-0.2,1  /)
nElems    = (/ 70,30,1 /)
elemtype  = 108
BCIndex   = (/ 5,3,2,0,0,6 /)

! BLOCK 3
Corner    = (/ 0.0,-0.2,0,, 0.5,-0.2,0,, 0.5,0.5,0,, 0.0,0.5,0,,
            ↪ 0.0,-0.2,1,, 0.5,-0.2,1,, 0.5,0.5,1,, 0.0,0.5,1  /)
nElems    = (/ 50,70,1 /)
elemtype  = 108
BCIndex   = (/ 5,0,2,4,0,6 /)

! BLOCK 4
Corner    = (/ -0.5,0.0,0,, 0.0,0.0,0,, 0.0,0.5,0,, -0.5,0.5,0,,
            ↪ -0.5,0.0,1,, 0.0,0.0,1,, 0.0,0.5,1,, -0.5,0.5,1  /)
nElems    = (/ 50,50,1 /)
elemtype  = 108
BCIndex   = (/ 5,0,0,4,1,6 /)

! BLOCK 5
Corner    = (/ -0.2,-0.2,0,, 0.0,-0.2,0,, 0.0,0.0,0,, -0.2,0.0,0,,
            ↪ -0.2,-0.2,1,, 0.0,-0.2,1,, 0.0,0.0,1,, -0.2,0.0,1  /)
nElems    = (/ 40,40,1 /)
elemtype  = 108
BCIndex   = (/ 5,0,0,0,0,6 /)
```

Configuration J

HOPR parameter file

Same as for configuration 4 in appendix A.4.3.1.

HOPR parameter file (Double resolution in $[-0.3, 0.1] \times [-0.4, 0.2]$)

```
nZones     = 5                    ! number of blocks

! BLOCK 1
Corner     = (/ -0.5,-0.5,0,, -0.3,-0.5,0,, -0.3,0.2,0,, -0.5,0.2,0,,
             ↪ -0.5,-0.5,1,, -0.3,-0.5,1,, -0.3,0.2,1,, -0.5,0.2,1  /)
nElems     = (/ 40,140,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,0,0,1,6 /)

! BLOCK 2
Corner     = (/ -0.3,-0.5,0,, 0.5,-0.5,0,, 0.5,-0.4,0,, -0.3,-0.4,0,,
             ↪ -0.3,-0.5,1,, 0.5,-0.5,1,, 0.5,-0.4,1,, -0.3,-0.4,1  /)
nElems     = (/ 160,20,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,2,0,0,6 /)

! BLOCK 3
Corner     = (/ 0.1,-0.4,0,, 0.5,-0.4,0,, 0.5,0.5,0,, 0.1,0.5,0,,
             ↪ 0.1,-0.4,1,, 0.5,-0.4,1,, 0.5,0.5,1,, 0.1,0.5,1  /)
nElems     = (/ 80,180,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,2,4,0,6 /)

! BLOCK 4
Corner     = (/ -0.5,0.2,0,, 0.1,0.2,0,, 0.1,0.5,0,, -0.5,0.5,0,,
             ↪ -0.5,0.2,1,, 0.1,0.2,1,, 0.1,0.5,1,, -0.5,0.5,1  /)
nElems     = (/ 120,60,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,0,4,1,6 /)

! BLOCK 5
Corner     = (/ -0.3,-0.4,0,, 0.1,-0.4,0,, 0.1,0.2,0,, -0.3,0.2,0,,
             ↪ -0.3,-0.4,1,, 0.1,-0.4,1,, 0.1,0.2,1,, -0.3,0.2,1  /)
nElems     = (/ 160,240,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,0,0,0,6 /)
```

A.4.3.3. Configurations with one shock wave, one rarefaction wave and two contact discontinuities

Configuration G

HOPR parameter file

```
nZones     = 4                    ! number of blocks
```

```
! BLOCK 1
Corner     = (/ -0.5,-0.5,0,, 0.5,-0.5,0,, 0.5,0.5,0,, -0.5,0.5,0,,
           ↪ -0.5,-0.5,1,, 0.5,-0.5,1,, 0.5,0.5,1,, -0.5,0.5,1  /)
nElems     = (/ 100,100,1 /)
elemtype   = 108
BCIndex    = (/ 5,0,2,0,0,6 /)

! BLOCK 2
Corner     = (/ -0.5,-1.0,0,, 0.5,-1.0,0,, 0.5,-0.5,0,, -0.5,-0.5,0,,
           ↪ -0.5,-1.0,1,, 0.5,-1.0,1,, 0.5,-0.5,1,, -0.5,-0.5,1  /)
nElems     = (/ 100,50,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,2,0,1,6 /)

! BLOCK 3
Corner     = (/ -1.0,-0.5,0,, -0.5,-0.5,0,, -0.5,0.5,0,, -1.0,0.5,0,,
           ↪ -1.0,-0.5,1,, -0.5,-0.5,1,, -0.5,0.5,1,, -1.0,0.5,1  /)
nElems     = (/ 50,100,1 /)
elemtype   = 108
BCIndex    = (/ 5,3,0,4,1,6 /)

! BLOCK 4
Corner     = (/ -0.5,0.5,0,, 0.5,0.5,0,, 0.5,1.0,0,, -0.5,1.0,0,,
           ↪ -0.5,0.5,1,, 0.5,0.5,1,, 0.5,1.0,1,, -0.5,1.0,1  /)
nElems     = (/ 100,50,1 /)
elemtype   = 108
BCIndex   =(/ 5,0,2,4,1,6 /)
```

Configuration H

HOPR parameter file

Same as for configuration G in appendix A.4.3.3.

Configuration K

HOPR parameter file

Same as for configuration 4 in appendix A.4.3.1.

A.4.4. Shock boundary layer interaction

The simulation of the shock boundary layer interaction was performed with a previous version of *FLEXI*, not published as open source. Nevertheless a parameter file for this simulation is listed for the sake of completeness. Since several options changed their name and available values, this parameter file can only be seen as a reference.

FLEXI parameter file

```
! OUTPUT
ProjectName       = SWBLIM267_p2p1_15_x_7
GatheredWrite     = T
GroupSize         = 16

! INTERPOLATION
N                 = 5     ! Polynomial degree
GeometricNGeo     = 1     ! Degree of mesh representation
NAnalyze          = 10    ! Number of analyze points
Filter_relax      = 0.0

doOverintegration = F     ! no overintegration

! MESH
MeshFile          = TBLM267_25010045_mesh.h5
useCurveds        = F

! FV
IndType           = 78     ! Jameson * Ducros
IndVariable       = 1      ! Density
FV_LimiterType    = MinMod
FV_Type           = 1      ! Gaussian
FV_IndMin         = 0.000002
FV_IndMax         = 0.000001

! EQUATION
IniExactFunc           = 167
RefState               = (/ 0.456097987,0.0,0.,0.,0.100195783 /)
UseNonDimensionalEqn   = T
BulkMach               = 2.67
BulkReynolds           = 100000.
Tref                   = 564. ! ref. temperature for Sutherland's law

! SPONGE
SpongeLayer       = T
SpongeExactFunc   = T
rampNo            = 2                  ! Number of sponge zones
damping           = 0.6                ! 1st sponge zone
xStart            = (/ 9.0,0.,0. /)    ! in x-direction
SpongeDir         = (/ 1.,0.,0. /)
SpongeDistance    = 1.4
damping           = 0.01               ! 2nd sponge zone
xStart            = (/ 0.,1.25,0. /)   ! in y-direction
SpongeDir         = (/ 0.,1.,0. /)
```

```
SpongeDistance   = 0.265

! BLASIUS
Ma_1             = 2.67          ! Mach Number of the BL
T_1              = 564.          ! Temperature of the BL
Re_1             = 100000.       ! Reynolds Number of the  BL
Wallparam        = 1             ! wall temperature: ad=0, iso=1
Twall_1          = 1236.57638454 ! if Wallparam=1: wall temperature
Pr               = 0.71          ! Prandtl-Number
kappa            = 1.4

! DISTURBANCES
nInterpolationData   = 3     ! Number of data sets to be interpolated
InterpolationMethod  = 0     ! spline (0) or linear interpolation (1)
nEigenfuncs          = 5     ! Number of Eigenfunctions used to disturb
                             ! initial solution
xScale               = 1.156 ! Scaling of the y-velocity by the
                             ! x-coordinate
AmpliMax             = (/ 0.04,0.04,0.04,0.04,0.08 /)
Frequenz             = (/ 3.0 ,4.0 ,6.0 ,8.0 ,9.0  /)
DisturbPhase         = (/ 0.01,0.2 ,0.8 ,0.0 ,0.7  /)
ObliqueFactor        = 21.0

! TIMEDISC
tend                 = 120.5 ! End time
CFLscale             = 0.9   ! Scaling of theoretical CFL number
DFLscale             = 0.6   ! Scaling of theoretical DFL number

! ANALYZE
Analyze_dt           = 0.5   ! Timestep of analyze outputs
CalcErrorNorms       = F     ! Calculate error norms
```

Bibliography

[1] C. Altmann, A. Taube, G. Gassner, F. Lörcher, and C.-D. Munz. "Shock detection and limiting strategies for high order discontinuous Galerkin schemes". In: *Shock Waves: 26th International Symposium on Shock Waves, Volume 2*. Ed. by K. Hannemann and F. Seiler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1053–1058. DOI: `10.1007/978-3-540-85181-3_42`.

[2] M. Atak, A. Beck, T. Bolemann, D. Flad, H. Frank, and C.-D. Munz. "High Fidelity Scale-Resolving Computational Fluid Dynamics Using the High Order Discontinuous Galerkin Spectral Element Method". In: *High Performance Computing in Science and Engineering 15: Transactions of the High Performance Computing Center, Stuttgart (HLRS) 2015*. Ed. by E. W. Nagel, H. D. Kröner, and M. M. Resch. Cham: Springer International Publishing, 2016, pp. 511–530. DOI: `10.1007/978-3-319-24633-8_33`.

[3] I. Babuska and M. Suri. "The p and h-p Versions of the Finite Element Method, Basic Principles and Properties". In: *SIAM Review* 36.4 (1994), pp. 578–632. DOI: `10.1137/1036141`.

[4] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press, 1996.

[5] D. S. Balsara, C. Altmann, C.-D. Munz, and M. Dumbser. "A sub-cell based indicator for troubled zones in RKDG schemes and a novel class of hybrid RKDG+HWENO schemes ". In: *Journal of Computational Physics* 226.1 (2007), pp. 586–620. DOI: `10.1016/j.jcp.2007.04.032`.

[6] G. E. Barter and D. L. Darmofal. "Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation". In: *Journal of Computational Physics* 229.5 (2010), pp. 1810–1827. DOI: `10.1016/j.jcp.2009.11.010`.

[7] F. Bassi and S. Rebay. "A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations". In: *Journal of computational physics* 131.2 (1997), pp. 267–279. DOI: `10.1006/jcph.1996.5572`.

[8] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, and M. Savini. “A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows”. In: *Proceedings of the 2nd European Conference on Turbomachinery, Fluid Dynamics and Thermodynamics*. Ed. by R. Decuypere and G. Dibelius. Antwerpen, Belgium, 1997, pp. 99–108.

[9] C. E. Baumann and J. T. Oden. “A discontinuous hp finite element method for the Euler and Navier–Stokes equations”. In: *International Journal for Numerical Methods in Fluids* 31.1 (1999), pp. 79–95. DOI: 10.1002/(SICI)1097-0363(19990915)31:1<79::AID-FLD956>3.0.CO;2-C.

[10] A. D. Beck, G. J. Gassner, T. Bolemann, H. Frank, F. Hindenlang, and C.-D. Munz. “Underresolved Turbulence Simulations with Stabilized High Order Discontinuous Galerkin Methods”. In: *Direct and Large-Eddy Simulation IX*. Ed. by J. Fröhlich, H. Kuerten, B. J. Geurts, and V. Armenio. Vol. 20. ERCOFTAC Series 1. Springer International Publishing, 2015, pp. 103–108. DOI: 10.1007/978-3-319-14448-1_14.

[11] A. Beck, T. Bolemann, D. Flad, H. Frank, G. Gassner, F. Hindenlang, and C.-D. Munz. “High Order Discontinuous Galerkin Spectral Element Methods for Transitional and Turbulent Flow Simulations”. In: *International Journal of Numerical Methods in Fluids* 76.8 (2014), pp. 522–548. DOI: 10.1002/fld.3943.

[12] K. Bock and J. Stiller. “Energy-Minimizing Curve Fitting for High-Order Surface Mesh Generation”. In: *Applied Mathematics* 5 (2014), pp. 3318–3327. DOI: 10.4236/am.2014.521309.

[13] C. Castro, C. Lozano, F. Palacios, and E. Zuazua. “Systematic Continuous Adjoint Approach to Viscous Aerodynamic Design on Unstructured Grids”. In: *AIAA Journal* 45.9 (Sept. 2007), pp. 2125–2139. DOI: 10.2514/1.24859.

[14] S. Clain, S. Diot, and R. Loubère. “A high-order finite volume method for systems of conservation laws – Multi-dimensional Optimal Order Detection (MOOD)”. In: *Journal of Computational Physics* 230.10 (2011), pp. 4028–4050. DOI: 10.1016/j.jcp.2011.02.026.

[15] B. Cockburn, S. Hou, and C.-W. Shu. “The Runge–Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws. IV: The Multidimensional Case”. In: *Mathematics of Computation* 54.190 (1990), pp. 545–581. DOI: 10.2307/2008501.

[16] B. Cockburn, S.-Y. Lin, and C.-W. Shu. "TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems". In: *Journal of Computational Physics* 84.1 (1989), pp. 90–113. DOI: 10.1016/0021-9991(89)90183-6.

[17] B. Cockburn and C.-W. Shu. "Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems". In: *Journal of Scientific Computing* 16.3 (2001), pp. 173–261. DOI: 10.1023/A:1012873910884.

[18] B. Cockburn and C.-W. Shu. "The Runge–Kutta discontinuous Galerkin Method for conservation Laws V: Multidimensional systems". In: *Journal of Computational Physics* 141.2 (1998), pp. 199–224. DOI: 10.1006/jcph.1998.5892.

[19] B. Cockburn and C.-W. Shu. "The Runge–Kutta local projection $P1$-discontinuous-Galerkin finite element method for scalar conservation laws". In: *1st National Fluid Dynamics Conference*. Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, 1987. DOI: 10.2514/6.1988-3797.

[20] B. Cockburn and C.-W. Shu. "TVB Runge–Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws II: General Framework". In: *Mathematics of Computation* 52.186 (1989), pp. 411–435. DOI: 10.2307/2008474.

[21] R. Courant, K. Friedrichs, and H. Lewy. "Über die partiellen Differenzengleichungen der mathematischen Physik". In: *Mathematische Annalen* 100.1 (1928), pp. 32–74. DOI: 10.1007/BF01448839.

[22] M. C. Delfour and J.-P. Zolésio. *Shapes and Geometries. Analysis, Differential Calculus and Optimization*. Philadelphia: Society for Industrial and Applied Mathematics, 2001. DOI: 10.1137/1.9780898719826.

[23] F. Ducros, V. Ferrand, F. Nicoud, C. Weber, D. Darracq, C. Gacherieu, and T. Poinsot. "Large-Eddy Simulation of the Shock/Turbulence Interaction". In: *Journal of Computational Physics* 152.2 (1999), pp. 517–549. DOI: 10.1006/jcph.1999.6238.

[24] M. Dumbser, O. Zanotti, R. Loubère, and S. Diot. "A posteriori subcell limiting of the discontinuous Galerkin finite element method for hyperbolic conservation laws". In: *Journal of Computational Physics* 278 (2014), pp. 47–75. DOI: 10.1016/j.jcp.2014.08.009.

[25] L. Euler. "Principes généraux de l'état d'équilibre des fluides". In: *Mémoires de l'académie des sciences de Berlin* 11 (1755), pp. 217–273.

[26] H. Frank and C.-D. Munz. "Direct aeroacoustic simulation of acoustic feedback phenomena on a side-view mirror". In: *Journal of Sound and Vibration* 371 (2016), pp. 132–149. DOI: 10.1016/j.jsv.2016.02.014.

[27] G. Gassner. "Discontinuous Galerkin Methods for the Unsteady Compressible Navier-Stokes Equations". PhD thesis. Universität Stuttgart, 2009. DOI: 10.18419/opus-3788.

[28] G. Gassner and D. A. Kopriva. "A Comparison of the Dispersion and Dissipation Errors of Gauss and Gauss–Lobatto Discontinuous Galerkin Spectral Element Methods". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2560–2579. DOI: 10.1137/100807211.

[29] N. R. Gauger. "Das Adjungiertenverfahren in der aerodynamischen Formoptimierung". PhD thesis. TU Braunschweig, 2003.

[30] J. W. Gibbs. "Fourier's Series". In: *Nature* 59 (Apr. 1899), pp. 200, 606. DOI: 10.1038/059606a0.

[31] M. Giles and N. Pierce. "Adjoint equations in CFD – Duality, boundary conditions and solution behaviour". In: *13th Computational Fluid Dynamics Conference*. Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, June 1997. DOI: 10.2514/6.1997-1850.

[32] M. B. Giles and N. A. Pierce. "An Introduction to the Adjoint Approach to Design". In: *Flow, Turbulence and Combustion* 65.3 (2000), pp. 393–415. DOI: 10.1023/A:1011430410075.

[33] A. Griewank. "Projected Hessians for Preconditioning in One-Step One-Shot Design Optimization". In: *Large-Scale Nonlinear Optimization*. Ed. by G. Di Pillo and M. Roma. Boston, MA: Springer US, 2006, pp. 151–171. DOI: 10.1007/0-387-30065-1_10.

[34] A. Harten. "High resolution schemes for hyperbolic conservation laws". In: *Journal of Computational Physics* 49.3 (1983), pp. 357–393. DOI: 10.1016/0021-9991(83)90136-5.

[35] R. Hartmann. "Numerical Analysis of Higher Order Discontinuous Galerkin Finite Element Methods". In: *VKI LS 2008-08: CFD – ADIGMA course on very high order discretization methods, Oct. 13-17, 2008*. Ed. by H. Deconinck. Von Karman Institute for Fluid Dynamics, Rhode Saint Genèse, Belgium, 2008.

[36] R. M. Hicks and P. A. Henne. "Wing Design by Numerical Optimization". In: *Journal of Aircraft* 15.7 (July 1978), pp. 407–412. DOI: 10.2514/3.58379.

[37] F. Hindenlang, T. Bolemann, and C.-D. Munz. “Mesh Curving Techniques for High Order Discontinuous Galerkin Simulations”. In: *IDIHOM: Industrialization of High-Order Methods – A Top-Down Approach: Results of a Collaborative Research Project Funded by the European Union, 2010–2014*. Ed. by N. Kroll, C. Hirsch, F. Bassi, C. Johnston, and K. Hillewaert. Cham: Springer International Publishing, 2015, pp. 133–152. DOI: `10.1007/978-3-319-12886-3_8`.

[38] F. Hindenlang. “Mesh Curving Techniques for High Order Parallel Simulations on Unstructured Meshes”. PhD thesis. Universität Stuttgart, 2014. DOI: `10.18419/opus-3957`.

[39] F. Hindenlang, G. J. Gassner, C. Altmann, A. Beck, M. Staudenmaier, and C.-D. Munz. “Explicit discontinuous Galerkin methods for unsteady problems ”. In: *Computers & Fluids* 61 (2012), pp. 86–93. DOI: `10.1016/j.compfluid.2012.03.006`.

[40] T. Hoefler and R. Belli. “Scientific Benchmarking of Parallel Computing Systems: Twelve Ways to Tell the Masses when Reporting Performance Results”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC ’15. Austin, Texas: ACM, 2015, 73:1–73:12. DOI: `10.1145/2807591.2807644`.

[41] A. Huerta, E. Casoni, and J. Peraire. “A simple shock-capturing technique for high-order discontinuous Galerkin methods”. In: *International Journal for Numerical Methods in Fluids* 69.10 (2012), pp. 1614–1632. DOI: `10.1002/fld.2654`.

[42] K. Ito, K. Kunisch, and G. H. Peichl. “Variational approach to shape derivatives”. In: *ESAIM: Control, Optimisation and Calculus of Variations* 14.3 (2008), pp. 517–539. DOI: `10.1051/cocv:2008002`.

[43] K. Ito, K. Kunisch, and G. H. Peichl. “Variational approach to shape derivatives for a class of Bernoulli problems”. In: *Journal of Mathematical Analysis and Applications* 314.1 (2006), pp. 126–149. DOI: `10.1016/j.jmaa.2005.03.100`.

[44] A. Jameson, L. Martinelli, and N. Pierce. “Optimum Aerodynamic Design Using the Navier–Stokes Equations”. In: *Theoretical and Computational Fluid Dynamics* 10.1 (1998), pp. 213–237. DOI: `10.1007/s001620050060`.

[45] A. Jameson, W. Schmidt, and E. Turkel. "Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes". In: *14th Fluid and Plasma Dynamics Conference*. Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, 1981. DOI: `10.2514/6.1981-1259`.

[46] A. Jameson. "Aerodynamic design via control theory". In: *Journal of Scientific Computing* 3.3 (1988), pp. 233–260. DOI: `10.1007/BF01061285`.

[47] A. Jameson. "Optimum aerodynamic design using CFD and control theory". In: *12th Computational Fluid Dynamics Conference*. Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, 1995. DOI: `10.2514/6.1995-1729`.

[48] L. Kaland, M. Sonntag, and N. R. Gauger. "Adaptive Aerodynamic Design Optimization for Navier-Stokes Using Shape Derivatives with Discontinuous Galerkin Methods". In: *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*. Ed. by D. Greiner, B. Galván, J. Périaux, N. Gauger, K. Giannakoglou, and G. Winter. Cham: Springer International Publishing, 2015, pp. 143–158. DOI: `10.1007/978-3-319-11541-2_9`.

[49] C. A. Kennedy, M. H. Carpenter, and R. Lewis. "Low-storage, explicit Runge–Kutta schemes for the compressible Navier–Stokes equations". In: *Applied Numerical Mathematics* 35.3 (2000), pp. 177–219. DOI: `10.1016/S0168-9274(99)00141-5`.

[50] D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations*. Springer, 2009. DOI: `10.1007/978-90-481-2261-5_8`.

[51] D. A. Kopriva. "Metric Identities and the Discontinuous Spectral Element Method on Curvilinear Meshes". In: *Journal of Scientific Computing* 26.3 (2006), pp. 301–327. DOI: `10.1007/s10915-005-9070-8`.

[52] D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. "Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method". In: *International Journal for Numerical Methods in Engineering* 53.1 (2002), pp. 105–122. DOI: `10.1002/nme.394`.

[53] A. Kurganov and E. Tadmor. "Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers". In: *Numerical Methods for Partial Differential Equations* 18.5 (2002), pp. 584–608. DOI: `10.1002/num.10025`.

[54] B. v. Leer. "Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme". In: *Journal of Computational Physics* 14.4 (1974), pp. 361–370. DOI: 10.1016/0021-9991(74)90019-9.

[55] B. v. Leer. "Towards the ultimate conservative difference scheme. III. Upstream-centered finite-difference schemes for ideal compressible flow". In: *Journal of Computational Physics* 23.3 (1977), pp. 263–275. DOI: 10.1016/0021-9991(77)90094-8.

[56] B. v. Leer. "Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection". In: *Journal of Computational Physics* 23.3 (1977), pp. 276–299. DOI: 10.1016/0021-9991(77)90095-X.

[57] B. v. Leer. "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method". In: *Journal of Computational Physics* 32.1 (1979), pp. 101–136. DOI: 10.1016/0021-9991(79)90145-1.

[58] B. van Leer. "Towards the ultimate conservative difference scheme. I. The quest of monotonicity". In: *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics: Vol. I General Lectures. Fundamental Numerical Techniques July 3–7, 1972 Universities of Paris VI and XI*. Ed. by H. Cabannes and R. Temam. Berlin, Heidelberg: Springer Berlin Heidelberg, 1973, pp. 163–168. DOI: 10.1007/BFb0118673.

[59] R. Loubère, M. Dumbser, and S. Diot. "A New Family of High Order Unstructured MOOD and ADER Finite Volume Schemes for Multidimensional Systems of Hyperbolic Conservation Laws". In: *Communications in Computational Physics* 16.3 (Sept. 2014), pp. 718–763. DOI: 10.4208/cicp.181113.140314a.

[60] C.-L. Navier. "Mémoire sur les lois du mouvement des fluids". In: *Mémoires de l'Académie Royale des Sciences de l'Institut de France* 6 (1822), pp. 389–416.

[61] J. Nitsche. "Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind". In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 36.1 (1971), pp. 9–15. DOI: 10.1007/BF02995904.

[62] K. T. Panourgias and J. A. Ekaterinaris. "A nonlinear filter for high order discontinuous Galerkin discretizations with discontinuity resolution within the cell ". In: *Journal of Computational Physics* 326 (2016), pp. 234–257. DOI: 10.1016/j.jcp.2016.08.049.

[63] D. Papadimitriou and K. Giannakoglou. "A continuous adjoint method with objective function derivatives based on boundary integrals, for inviscid and viscous flows ". In: *Computers & Fluids* 36.2 (2007), pp. 325–341. DOI: 10.1016/j.compfluid.2005.11.006.

[64] P.-O. Persson and J. Peraire. "Sub-Cell Shock Capturing for Discontinuous Galerkin Methods". In: *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, 2006. DOI: 10.2514/6.2006-112.

[65] O. Pironneau. "On optimum profiles in Stokes flow". In: *Journal of Fluid Mechanics* 59.1 (1973), pp. 117–128. DOI: 10.1017/S002211207300145X.

[66] S. Premasuthan, C. Liang, and A. Jameson. "Computation of flows with shocks using the Spectral Difference method with artificial viscosity, I: Basic formulation and application". In: *Computers & Fluids* 98 (2014), pp. 111–121. DOI: 10.1016/j.compfluid.2013.12.013.

[67] J. Qiu and C.-W. Shu. "Hermite WENO schemes and their application as limiters for Runge–Kutta discontinuous Galerkin method: one-dimensional case". In: *Journal of Computational Physics* 193.1 (2004), pp. 115–135. DOI: 10.1016/j.jcp.2003.07.026.

[68] W. H. Reed and T. R. Hill. *Triangular mesh methods for the neutron transport equation*. Technical Report LA-UR-73-479. Los Alamos Scientific Laboratory, 1973.

[69] P. L. Roe. "Characteristic-Based Schemes for the Euler Equations". In: *Annual Review of Fluid Mechanics* 18.1 (1986), pp. 337–365. DOI: 10.1146/annurev.fl.18.010186.002005.

[70] P. L. Roe. "Discrete models for the numerical analysis of time-dependent multidimensional gas dynamics". In: *Journal of Computational Physics* 63.2 (1986), pp. 458–476. DOI: 10.1016/0021-9991(86)90204-4.

[71] M. Sabat, A. Larat, A. Vié, and M. Massot. "Comparison of Realizable Schemes for the Eulerian Simulation of Disperse Phase Flows". In: *Finite Volumes for Complex Applications VII-Elliptic, Parabolic and Hyperbolic Problems*. Ed. by J. Fuhrmann, M. Ohlberger, and C. Rohde. Vol. 78. Springer Proceedings in Mathematics & Statistics. Springer International Publishing, 2014, pp. 935–943. DOI: 10.1007/978-3-319-05591-6_95.

[72] S. Schmidt. "Efficient Large Scale Aerodynamic Design Based on Shape Calculus". PhD thesis. University of Trier, Germany, 2010.

[73] S. Schmidt, C. Ilic, V. Schulz, and N. R. Gauger. "Airfoil design for compressible inviscid flow based on shape calculus". In: *Optimization and Engineering* 12.3 (2011), pp. 349–369. DOI: 10.1007/s11081-011-9145-3.

[74] S. Schmidt, C. Ilic, V. Schulz, and N. R. Gauger. "Three-Dimensional Large-Scale Aerodynamic Shape Optimization Based on Shape Calculus". In: *AIAA Journal* 51.11 (Sept. 2013), pp. 2615–2627. DOI: 10.2514/1.J052245.

[75] S. Schmidt and V. Schulz. "Shape derivatives for general objective functions and the incompressible Navier-Stokes equations". In: *Control and Cybernetics* 39.3 (2010), pp. 677–713.

[76] V. Schulz and I. Gherman. "One-Shot Methods for Aerodynamic Shape Optimization". In: *MEGADESIGN and MegaOpt — German Initiatives for Aerodynamic Simulation and Optimization in Aircraft Design: Results of the closing symposium of the MEGADESIGN and MegaOpt projects, Braunschweig, Germany, 23–24 May, 2007*. Ed. by N. Kroll, D. Schwamborn, K. Becker, H. Rieger, and F. Thiele. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 207–220. DOI: 10.1007/978-3-642-04093-1_15.

[77] C. W. Schulz-Rinne. "Classification of the Riemann Problem for Two-Dimensional Gas Dynamics". In: *SIAM Journal on Mathematical Analysis* 24.1 (1993), pp. 76–88. DOI: 10.1137/0524006.

[78] C. W. Schulz-Rinne, J. P. Collins, and H. M. Glaz. "Numerical Solution of the Riemann Problem for Two-Dimensional Gas Dynamics". In: *SIAM Journal on Scientific Computing* 14.6 (1993), pp. 1394–1414. DOI: 10.1137/0914082.

[79] C.-W. Shu and S. Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes". In: *Journal of Computational Physics* 77.2 (1988), pp. 439–471. DOI: 10.1016/0021-9991(88)90177-5.

[80] C.-W. Shu and S. Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes, Part II". In: *Journal of Computational Physics* 83.1 (1989), pp. 32–78. DOI: 10.1016/0021-9991(89)90222-2.

[81] G. A. Sod. "A survey of several finite difference methods for systems of non-linear hyperbolic conservation laws". In: *Journal of Computational Physics* 27.1 (1978), pp. 1–31. DOI: 10.1016/0021-9991(78)90023-2.

[82] B. Soemarwoto. *The Variational Method for Aerodynamic Optimization Using the Navier-Stokes Equations*. Tech. rep. 97-71. Institute for Computer Applications in Science and Engineering, 1997.

[83] J. Sokolowski and J.-P. Zolésio. *Introduction to Shape Optimization. Shape Sensitivity Analysis*. Berlin: Springer-Verlag, 1992. DOI: 10.1007/978-3-642-58106-9.

[84] M. Sonntag and C.-D. Munz. "Efficient Parallelization of a Shock Capturing for Discontinuous Galerkin Methods using Finite Volume Sub-cells". In: *Journal of Scientific Computing* 70.3 (2017), pp. 1262–1289. DOI: 10.1007/s10915-016-0287-5.

[85] M. Sonntag, S. Schmidt, and N. R. Gauger. "Shape derivatives for the compressible Navier-Stokes equations in variational form ". In: *Journal of Computational and Applied Mathematics* 296 (2016), pp. 334–351. DOI: 10.1016/j.cam.2015.09.010.

[86] K. A. Sørensen and H. Bieler. "Verification and Assessment". In: *ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*. Springer, Berlin, Heidelberg, 2010, pp. 465–482. DOI: 10.1007/978-3-642-03707-8_33.

[87] G. G. Stokes. "On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids". In: *Transactions of the Cambridge Philosophical Society* 8 (1845), pp. 287–305. DOI: 10.1017/CBO9780511702242.005.

[88] P. K. Sweby. "High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws". In: *SIAM Journal on Numerical Analysis* 21.5 (1984), pp. 995–1011. DOI: 10.1137/0721062.

[89] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. "Robust untangling of curvilinear meshes". In: *Journal of Computational Physics* 254 (2013), pp. 8–26. DOI: 10.1016/j.jcp.2013.07.022.

[90] J. VonNeumann and R. D. Richtmyer. "A Method for the Numerical Calculation of Hydrodynamic Shocks". In: *Journal of Applied Physics* 21.3 (1950), pp. 232–237. DOI: 10.1063/1.1699639.

[91] J. H. Williamson. "Low-storage Runge–Kutta schemes". In: *Journal of Computational Physics* 35.1 (1980), pp. 48–56. DOI: 10.1016/0021-9991(80)90033-9.

[92] P. Woodward and P. Colella. "The numerical simulation of two-dimensional fluid flow with strong shocks". In: *Journal of Computational Physics* 54.1 (1984), pp. 115–173. DOI: 10.1016/0021-9991(84)90142-6.

[93] N. J. Wright, S. Smallen, C. M. Olschanowsky, J. Hayes, and A. Snavely. "Measuring and Understanding Variation in Benchmark Performance". In: *Proceedings of the 2009 DoD High Performance Computing Modernization Program Users Group Conference*. HPCMP-UGC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 438–443. DOI: `10.1109/HPCMP-UGC.2009.72`.

[94] H. Yee, N. Sandham, and M. Djomehri. "Low-Dissipative High-Order Shock-Capturing Methods Using Characteristic-Based Filters". In: *Journal of Computational Physics* 150.1 (1999), pp. 199–238. DOI: `10.1006/jcph.1998.6177`.

[95] O. Zanotti, F. Fambri, M. Dumbser, and A. Hidalgo. "Space–time adaptive ADER discontinuous Galerkin finite element schemes with a posteriori sub-cell finite volume limiting ". In: *Computers & Fluids* 118 (2015), pp. 204–224. DOI: `10.1016/j.compfluid.2015.06.020`.

[96] A. Zymaris, D. Papadimitriou, K. Giannakoglou, and C. Othmer. "Continuous adjoint approach to the Spalart–Allmaras turbulence model for incompressible flows ". In: *Computers & Fluids* 38.8 (2009), pp. 1528–1538. DOI: `10.1016/j.compfluid.2008.12.006`.

List of tables

List of figures