A parallel discontinuous Galerkin code for the Navier-Stokes and Reynolds-averaged Navier-Stokes equations

Von der Fakultät für Luft- und Raumfahrttechnik und Geodäsie

der Universität Stuttgart zur Erlangung der Würde

eines Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Abhandlung

Vorgelegt von

Björn Landmann

geboren in Kandel

Hauptberichter: Prof. Dr.-Ing. Siegfried Wagner

Mitberichter: Prof. Dr. John Ekaterinaris

Tag der mündlichen Prüfung: 14.12.2007

Institut für Aerodynamik und Gasdynamik

Universität Stuttgart

2008

Contents

Symbols

Greek

α	angle of attack
α_j, α_r	local lifting operators
γ	specific heat ratio
δ_{ij}	Kronecker delta function
$\mathcal{O}\,()$	of the order of magnitude
λ	heat conduction coefficient
λ_t	turbulent heat conduction coefficient
ν	kinematic viscosity μ/ρ
μ	molecular dynamic viscosity
μ_t	eddy viscosity
ξ, η, ζ	reference coordinates
ρ	fluid density
τ, τ_{ij}	viscous stress tensor
Θ	gradient of the state vector
σ, σ_{ij}	turbulent stress tensor
$\tilde{\nu}$	working variable of SA turbulence model
ω	specific dissipation rate
$\tilde{\omega}$	logarithmised specific dissipation rate
$\lvert\Omega\rvert$	magnitude of vorticity vector

Latin

b_k	basis function
B	bilinear form
c	chord length
c_p	specific heat capacity at constant pressure
c_d	global drag coefficient
c_l	global lift coefficient
D	cylinder diameter
d	distance to the nearest wall
e	volume-specific inner energy
E	volume-specific total energy
F_i^x, F_i^y, F_i^z	cartesian components of the inviscid flux tensor
F_v^x, F_v^y, F_v^z	cartesian components of the viscid flux tensor
$\mathcal{F}_i$, $\mathcal{F}_v$	inviscid and viscid flux tensor
H	volume-specific total enthalpy
$\mathcal{H}_i$	numerical inviscid flux
$\mathcal{H}_v$	numerical viscid flux
$\mathcal{H}_{aux}$	numerical auxiliary flux
k	turbulent kinetic energy
M, M_{ij}	element mass matrix
Ma	Mach number
$\vec{n}$	normal vector
p	mean static pressure
Pr	Prandtl number
q	heat flux vector
R	correction gradient
Re	Reynolds number
S	mean strain-rate tensor
T	static temperature
TU_∞	freestream tubulence intensity
t	time
u, v, w	cartesian components of velocity vector
u_i	cartesian component of v in index notation
u_h	approximate solution
U	conservative state vector
U_k	solution degrees of freedom
u^+	dimensionless, sublayer-scaled velocity u/u_τ
u_τ	friction velocity $\sqrt{\tau_w/\rho}$
v	mean velocity vector
v	test function
v_h	approximate test function
V_k	test function degrees of freedom
y^+	dimensionless, sublayer-scaled wall distance yu_τ/ν

Superscripts

$'$	fluctuating part of a flow variable, Reynolds-averaged
$\bar{}$	time-averaged value, Reynolds-averaged
$''$	fluctuating part of a flow variable, Favre-averaged
$\tilde{}$	time-averaged value, Favre-averaged
$^+$	dimensionless, sublayer-scaled value
$^+$	interface value taken from the exterior
$^-$	interface value taken from the interior
^{b}	boundary value

Abbreviations

BICGSTAB	Biconjugate gradient stabilised
BR	Bassi-Rebay
BVI	Blade vortex interaction
CAA	Computational aeroacoustic
CFD	Computational fluid dynamics
CSD	Computational structural dynamics
DES	Detached eddy simulation
DG	Discontinuous Galerkin
DIAG	Diagonal
DNS	Direct numerical simulation
DS	Dynamic stall
FD	Finite differences
FE	Finite elements
FV	Finite volumes
GMRES	Generalised minimum residual
HLL	Harten-Lax-van Leer
HSI	High speed impulsive
ILU	Incomplete lower upper
IMEX	Implicit-explicit
IP	Interior penalty
ITL	Iterative template library
LDG	Local discontinuous Galerkin
LES	Large eddy simulation
LHS	Left hand side
MTL	Matrix template library
NIPG	Nonsymmetric interior penalty Galerkin
ODE	Ordinary differential equation
RANS	Reynolds-averaged Navier-Stokes
RHS	Right hand side
RK	Runge-Kutta
RKDG	Runge-Kutta Discontinuous Galerkin

SA	Spalart-Allmaras
SGS	Subgrid scale
SSOR	Symmetric successive overrelaxation
TNT	turbulent not turbulent
TVD	Total Variation Diminishing
URANS	Unsteady Reynolds-averaged Navier-Stokes

Abstract

The numerical simulation of flow problems has gained further importance during the recent years. This is—obviously next to the increase of computing power—due to the steady improvements of the numerical discretisation methods and the improvement of the efficiency of the associated solution algorithms. Even wider acceptance could be obtained, if the flexibility, the automatism or the efficiency of the flow simulation could be further improved. One promising and relatively new discretisation approach, which recently attracted attention, is the discontinuous Galerkin (DG) method. The DG approach seems to have the potential to solve some problems, which mainly have their origin in the presently used discretisation methods.

In the present work, an attempt has been made to examine the qualities of the present state of the art discretisation methods based on the DG approach in space. Therefore, different DG methods, including some recently developed methods, are employed for the discretisation of the compressible Euler- and Navier-Stokes equations as well as for the Reynolds-averaged Navier-Stokes equations. The turbulence modeling is applied with a one-equation or a two equation model, namely the Spalart-Allmaras or the $k-\omega$ model. The temporal discretisation of the partial differential equations is either performed explicitely with the aid of classical Runge-Kutta methods or with an implicit discretisation approach. In order to retain the formal order of accuracy in the interior of the flow field, the boundary is approximated with corresponding geometrical accuracy.

Computations are performed for steady and unsteady one- and two-dimensional flow problems, including standard test cases such as the flow over a flat plate or around a circular cylinder. The computed results are compared with experimental, computational, exact, and empirical data. Thereby, also practical differences of the DG solver compared to traditionally used discretisation methods, like post-processing, are discussed.

Convergence analysis demonstrates, that the implemented solution method delivers the user defined formal order of discretisation accuracy on unstructured grids. In spite of the relatively coarse grids, consulted for all computations, all obtained results are in very good agreement to the reference data, due to the high-order discretisation. Furthermore, the need of the high-order boundary implementation is clearly demonstrated by several flow test cases.

For the computation of high Reynolds number (turbulent) flows, two main conclusions can be drawn. First, the temporal discretisation has do be performed implicitly, in order to overcome the severe time step restriction of explicit schemes. Second, a curved hybrid grid approach is absolutely mandatory, if high-order DG based simulations should have the potential to outperform standard second order methods, which are based on the

classical FV approach. This is due to the fact, that high-order calculations need less fine, but curved grids. Such grids can easily degenerate in the boundary layer, if curved triangular or tetrahedral elements are used.

The developed flow solver behaves very robust for the Euler- and Navier-Stokes test cases. For the RANS test cases, the clean implementation of the turbulence model delivers some problems concerning positivity of turbulence quantities and associated stability of the solver. However, these can be reduced, either by modifying the equations or/and introducing positivity constraints in the solution algorithm. The solver is characterised by excellent parallel efficiency, which is mainly achieved by the clever parallelisation strategy, which utilises the locality of the DG discretisation.

Zusammenfassung

Die numerische Simulation von Strömungsvorgängen hat in den letzten Jahren enorm an Bedeutung gewonnen. Diese Entwicklung ist - selbstverständlich neben dem Zuwachs an Rechenleistung - der ständigen Verbesserung der numerischen Diskretisierungsmethoden und der Steigerung der Effizienz der zugehörigen Lösungsalgortihmen zu verdanken. Eine noch größere Akzeptanz könnte erreicht werden, wenn die Flexibilität, der Automatismus oder auch die Effizienz der Strömungssimulation noch weiter verbessert werden könnten. Eine vielversprechende, relativ neue Diskretisierungsmethode, welche in den letzten Jahren enorm an Aufmerksamkeit gewonnen hat, ist die unstetige Galerkin Methode (discontinuous Galerkin, DG). Der DG Ansatz könnte das Potential besitzen einige Probleme, die ihren Ursprung in den momentan verwendeten Diskretisierungsverfahren haben, zu lösen.

In der vorliegenden Arbeit wird ein Versuch unternommen, die Eigenschaften DG basierter Verfahren, welche dem aktuellen Stand der Forschung entsprechen, zu untersuchen. Hierzu werden einige DG Verfahren, in denen auch erst kürzlich entwickelte Verfahren enthalten sind, für die räumliche Diskretisierung der kompressiblen Euler- und Navier-Stokes Gleichungen sowie auch jener der Reynolds-gemittelten Navier-Stokes (RANS) Gleichungen herangezogen. Die Modellierung der Turbulenz, wird entweder mit einem Ein- oder Zweigleichungsmodell, dem Spalart-Allmaras oder dem $k - \omega$ Model vorgenommen. Die zeitliche Diskretisierung der partiellen Differentialgleichungen erfolgt mit klassischem Runge-Kutta Verfahren auf explizite Weise oder aber mit einem impliziten Ansatz in der Zeit. Um die formale Genauigkeitsordnung des DG Verfahrens im Inneren des Rechengebietes auch am Rand beizubehalten, werden auch die Ränder mit entprechender (geometrischer) Genauigkeit approximiert.

Es werden Berechnungen für stationäre und instationäre ein- und zweidimensionale Strömungsprobleme durchgeführt, welche Testfälle wie die Strömung über eine ebene Platte oder um einen Kreiszylinder beinhalten. Die erzielten Resultate werden mit experimentellen, berechneten, exakten und empirischen Daten verglichen. Hierbei werden auch praktische Aspekte des DG Lösers, wie zum Beispiel das Post-Processing, im Vergleich zu traditionellen Diskretisierungsverfahren, diskutiert.

Mit Hilfe von Konvergenzanalysen wird gezeigt, daß das implementierte Lösungsverfahren, die vom Benutzer spezifizierte Rechengenauigkeitsordnung auf unstrukturierten Gittern liefert. Desweiteren zeigt sich, daß infolge des Diskretisierungsansatzes hoher Ordnung, trotz der Verwendung äußerst grober Rechengitter, für alle Testfälle eine ausgezeichnete Übereinstimmung mit den jeweiligen Referenzergebnissen erzielt werden kann. Außerdem wird anhand einiger Testfälle deutlich gemacht, daß die Verwendung

von (geometrischen) Randbedingungen hoher Ordnung zwingend notwendig ist.

Für die Berechnung von (turbulenten) Strömungen hoher Reynoldszahl können zwei wesentliche Schlußfolgerungen gezogen werden. Erstens muß die zeiliche Diskretisierung auf implizite Weise erfolgen, um die harte Zeitschrittbeschränkung der expliziten DG Verfahren zu umgehen. Zweitens, ist eine krummlinige hybride Gitterstrategie zwingend notwendig, damit DG basierte Simulationen das Potential haben können, traditionelle Verfahren zweiter Ordnung, welche auf der Methode der finiten Volumen basieren, in Sachen Effizienz zu übertrumpfen. Das liegt an der Tatsache, daß Berechnungen mit hoher Genauigkeitsordnung weniger feine gekrümmte Rechengitter erfordern, und deshalb solche Gitter leicht im Bereich der Grenzschicht degenerieren können, falls Dreiecke oder Tetraeder als einzige Diskretisierungselemente genutzt würden.

Der entwickelte Strömungslöser verhält sich bei der Berechnung von Euler- und Navier-Stokes Problemen sehr robust. Bei der Simulation von RANS Testfällen ergeben sich bei standardgemäßer Implementierung der Turbulenzmodelle Probleme mit der Positivität von Turbulenzgrößen und damit verbunden leider auch Stabiltätsprobleme des Lösers. Diese Probleme können jedoch entweder durch Modifikation der Gleichungen und/oder Einführung von Positivitätsbeschränkungen im Lösungsprozess, reduziert werden. Der Löser besticht durch exzellente parallele Effizienz, welche vor allem durch eine geschickte Parallelisierungsstrategie, die die Lokalität des DG Ansatzes ausnutzt, erzielt werden kann.

1. Introduction

Fluid dynamics plays an important role in the design process of a large variety of industrial products, and in particular in the aeronautical industry. Historically, fluid dynamics was divided into an experimental and a theoretical branch. The former performs experiments and develops experimental techniques, which lead to a better understanding of the physical phenomena involved in the flow field. However, wind tunnel tests are expensive and time-consuming and for some cases still cannot be accomplished for technical reasons. The theoretical branch attempts to solve the governing equations or simplified forms of them by means of mathematical methods. Unfortunately, analytical solutions exist only for a few, very simple flow configurations.

Due to the enormous growth in computer power over the last 40 years, both in speed and memory, the numerical solution of the governing equations—generally known under the name of computational fluid dynamics (CFD)—has become an important third approach in the study of fluid mechanics, in addition to pure experiment and pure theory. CFD can be applied to flow problems that cannot be investigated experimentally or analytically. Therefore, the subject of this thesis is CFD.

Unfortunately, there exists a very complex physical phenomenon in fluid mechanics and consequently in CFD—namely turbulence. In nature, laminar (turbulence free) flows are rather an exception, and most interesting flows in engineering applications are turbulent. The proper numerical simulation of high Reynolds number turbulent flows, by direct numerical simulation (DNS), with the aid of CFD still by far exceeds the computing power and memory space of current supercomputers. The only resort is to use some kind of (turbulence) modeling in order to reduce the problem size and the associated computational requirements. As suspected, the resulting side-effect due to the modeling is that the reliability of the simulated results decreases. Thus, CFD will certainly not fully replace experiments in the foreseeable future, but it could further strongly reduce costly wind tunnel tests in the design process [103].

The outline of the introduction chapter is as follows. In section 1.1, the demands on CFD methods for complex applications of industrial interest, like rotary or fixed-wing airplanes, are summarised. In section 1.2, we subscribe to what extent classical high-order methods can meet these demands. In section 1.3, the peculiarities of the DG method and the advantages or disadvantages, compared to the classical high-order methods, are described. In the last section, a motivation for and an overview of this thesis, is presented.

1.1. Demands on CFD methods

A very simple answer to the question: "What are the demands on a CFD method" is: "A good CFD code/method should deliver a prescribed, desired solution accuracy, with the lowest possible costs in terms of CPU time". However, the answer is not as simple, because the desired solution accuracy strongly depends on the problem to be solved, and the overall time or turnaround time, that we need to obtain a result, is not just the CPU time of the simulation. In particular, we have to include the strong problem dependent grid generation time of a basic (first shot) grid and several manual or (semi)automatic grid adaptations in the turnaround time. Thus, we can easily conclude, that our simple appearing question can only be answered in a problem dependent way.

Firstly, we have to define, which kind of flow problem, we want to solve with our method. Then a question, at least of the same importance is, on which kind of (super)computer we want to, or can run our simulation on. Since our long-term objective is the simulation of the aeroelastic and aeroacoustic behavior of a complete helicopter in hover and forward flight, the demands on the CFD method should be derived here from the flow around helicopter rotors. These demands mainly coincide with those requested from the calculation of unsteady flow fields around complex (fixed-wing) geometries.

In order to accurately predict the aeroelastic and aeroacoustic behavior of a helicopter, the most challenging part definitely is the aerodynamic simulation—the CFD part. The demands are very extensive, since the flow field is complicated by the presence of so many fundamental fluid dynamical research problems.

In forward flight, at the advancing blade, the blade relative velocity is composed of the flight speed and the rotational speed. Therefore, the relative velocity can approach the speed of sound and locally supersonic flow may be present. The position and strength of the resulting shock strongly influences possible shock-induced separation or shock-induced high speed impulsive (HSI) noise. Therefore, our numerical method of choice has to accurately handle flow fields, where shock waves are/can be present.

However, at the retreating blade, we observe low relative velocities or even partially reverse flow. This can result in the dynamic stall (DS) of the flow, which is the abrupt separation of the flow and complete break down of the local aerodynamic lift, leading to strong vibrations of the blade and in particular at the helicopter fuselage. Since the local relative flow speed is extremely low, the dynamic stall is mainly dominated by viscous effects. Therefore, our scheme should model these viscous effects as accurately as possible, in order to predict the DS phenomena.

In contrast to most fixed-wing configurations, we are also interested in the wake behind the rotor blades, because there are possible flight scenarios (vertical descent or landing flight) where a succeeding blade can interfere/collide with the vortices shed from other blades. This phenomenon, which is called blade vortex interaction (BVI), is responsible for structural BVI vibrations and BVI noise. Therefore, capturing of the wake is essential, for accurately predicting the rotor flow field [36]. It is well known that our scheme of choice has to be low-dissipative in order to conserve the blade vortices at least to the point of BVI. Otherwise tremendous grid resolution is needed [68]. Furthermore, application of h-type grid adaptivity with standard volume techniques [93, 42] that do

not readily accept hanging nodes, is difficult to implement.

Since accurate rotor flow simulations require correct blade motions, aero-elastic simulations need to be performed [78]. Consequently, this can only be achieved, if the CFD method allows moving and/or deforming meshes.

The grid generation process for a complete helicopter (main and tail rotor, fuselage, etc.) is very time-consuming, if block-structured or Chimera [19] approaches are used [79, 77, 5]. Especially the Chimera approach requires significant pre-processing in order to identify interpolation stencils. This effort can be reduced, if instead unstructured grid generation is practiced. Therefore, a suitable CFD method should smoothly work (high accurate) on unstructured grids.

Typical simulations only for an isolated main rotor in forward flight require several million cells, in order to obtain acceptable accuracy [37, 63, 68]. Problems of that size can only be processed in a reasonable time (several days), if supercomputers with a (sustained) performance of several hundred gigaflops are considered. In high performance computing there is a trend to machines with distributed memory, as the increase of computing power is more and more achieved by interconnecting many processors and not by increased performance of the individual processors. This trend can be clearly identified by having a look at the recent lists of top 500 supercomputers [3]. Most of the listed machines achieve their performance by agglomeration of several hundred or thousand scalar processors and even the vector machines mainly achieve their computing power by distributed memory parallelisation of many vector nodes. Therefore, a promising CFD method—working on such massively parallel computers—should possess excellent parallelisability properties, in order to take full advantage from agglomeration of several hundred or thousands scalar processors.

As described above the inevitable use of some kind of turbulence modeling places uncertainty in the simulated results. Consequently, the amount of modeling should be kept as small as possible. This amount depends on the respectively current computing power. The modeling approach, which can also be used in industrial practice, was and still is the RANS approach, which is based on the numerical simulation of the Reynolds-averaged Navier-Stokes (RANS) equations. One approach with less modeling influence is the large eddy simulation (LES) [87] or in between RANS and LES the so-called zonal hybrid RANS-LES approach [90, 18]. Due to their high costs, compared to simple RANS, both (LES and RANS-LES) are mainly in academical use up today. However, expecting that computing power increases steadily according to Moore's law, they will replace RANS in the near future. LES as well as the RANS-LES approaches merge into DNS, if the grid resolution is sufficient for proper DNS. The influence of numerical errors on LES is analysed by several researchers in the last decades [61, 25]. One important awareness is, that the effect of numerical errors can mask the subgrid scale (SGS) model viscosity, which should model the effect of the SGS motions on the resolved velocities [61]. SGS models are developed independently from their numerical solution, under the assumption that it is error-free. Hence, a clean/proper approach is to use a scheme, which itself produces sufficiently low discretisation errors. In our opinion, this is a key argument for the use of high-order schemes. It is well known, that high-order schemes become efficient, in particular, if the error tolerance is low. To conclude, when

Physical phenomena or practical aspect	Demand on CFD method
Transonic flow (shock waves)	Compressible, robust at shocks
Dynamic stall	Viscosity (NS, RANS, RANS-LES)—no Euler
BVI noise, BVI vibration	Low dissipation
Aeroelasticity	Moving and/or deforming grids
Complex geometry	Unstructured grid approach
Massively parallel supercomputer (cluster)	Excellent parallelisability
High Reynolds number turbulence	Turb. modeling (RANS, LES, RANS-LES)
Low tolerable error levels (LES, RANS-LES)	High-order

Table 1.1.: Physical phenomena and/or practical aspects and their related demands on CFD methods

approaches, such as LES or RANS-LES, will replace RANS in the future, a high-order discretisation scheme is desired.

The demands on a (helicopter) CFD code, described in the previous paragraphs, is summarised in Table 1.1.

1.2. Traditional higher-order discretisations in space in CFD for aerodynamic flows

CFD has matured significantly over the past few decades. Because, numerous research efforts have been aimed at developing algorithms for solving partial differential equations. These efforts have lead to many discretisation methods and numerical schemes. Nowadays, the most popular discretisation methods used in CFD are the Finite Difference (FD), Finite Volume (FV) and the Finite Element (FE) techniques. In applied aerodynamics, most used CFD methods are at best second-order accurate in space [103, 101], meaning that the solution error decreases as $\mathcal{O}\left(h^2\right)$, where h is a measure of the grid spacing[1]. Typically, a method is called a high-order, if the discretisation error reduces at least with third or higher order, $\mathcal{O}\left(h^{\geq 3}\right)$.

It is very important to state, that the achievement of high-order results is practically useless by just proofing that an algorithm is high-order for several simple test problems in one-dimension. At least of same importance is, that the scheme possesses similar properties for (application oriented) at least in two or better in three-dimensional flows. As mentioned above most flow fields of practical interest are turbulent. Hence, a high-order scheme has to work for at least the Navier-Stokes equations, but in fact it also has to work properly in combination with some kind of turbulence modeling, to be attractive for industrial usage.

Now, we wish to give an overview of the existing classical discretisation methods, in

[1] Another often used explanation is, that the order of a scheme is a measure of the rapidity, the discretisation error decreases, with the mesh size tending to zero.

which emphasis is put on how to construct a high-order implementation for at least the Navier-Stokes equations. The restrictions or benefits that a high-order solver based on these methods possesses are mentioned. Based on this review, we later want to give a motivation for the work presented in this thesis.

1.2.1. Finite Difference (FD) methods

The most simple approach to discretise a conservation law is based on its differential or divergence form of the integral equations. The idea of the FD method is to find a discrete approximation for the occurring derivatives and replace the analytical derivatives with the discrete ones, resulting in a discrete problem, which can be solved numerically. There exist a multiplicity of approaches for the discrete estimations of the derivatives. Excellent text book describing these methodologies are for example Hirsch [52, 53] or Ferziger and Peric [44].

The main strengths of FD schemes are, that they are easy to program and that they are extremely efficient in terms of computational cost. Due to their excellent efficiency and well analysed numerical properties, they are often/traditionally used for numerically sensitive and computationally costly problems, such as laminar-turbulent transition [60, 105]. High order versions of the FD method are easy to construct, since the accuracy of the method is determined by the accuracy of the estimation of the discrete derivative [67, 109].

High-order approximations of the derivatives—resulting in a high-order FD scheme—are very easy to obtain, by just adding more grid points to the point-wise discretisation stencil. A major drawback of the FD method in application is, that it is restricted to structured grids. In principle, it is possible to derive a finite difference scheme on unstructured grids [69], but for that a reconstruction of a polynomial function is needed, which is a very complex problem for unstructured grids and will be described in more detail in the next section. In addition, high-order FD methods require smooth, regular grids for stability reasons [102], meaning that for geometrically complex configurations high effort has to be put into the block-structured grid generation. Consequently, FD schemes are mainly applied for fundamental studies, where elementary geometries are the objects of research. The most common solution to ensure stability is to reduce the boundary accuracy, and special near wall grid refinement is used to compensate for the reduced accuracy.

1.2.2. Finite Volume (FV) methods

In contrast to the FD method, the starting point of FV schemes is the integral formulation of the conservation laws. Instead of estimating derivatives, like in the previously discussed FD method, fluxes through the discretisation element boundaries have to be evaluated. There are different views how these fluxes have to be chosen. A very popular approach for convection-dominated problems is the upwind method [84, 66], where the flux choice is based on characteristics of wave propagation.

The higher order versions of the FV method are generally obtained with the help of a so called reconstruction procedure [12], whereas an intermediate higher-order solution is constructed out of the piecewise constant element data of adjacent cells. The cells, which are included in the reconstruction, are depicted as the reconstruction stencil of the method. The problem with high-order FV methods working on unstructured grids is, that the reconstruction stencil (especially in 3D) becomes extremely large [35] and the resulting scheme would be extremely complex to program, and more importantly, would be expensive in terms of CPU time. In general, real high-order is only achieved on relatively smooth and regular grids. A further drawback is, that due to the increased stencil, such a scheme is not suited for efficient parallelisation, because the stencil is quite large for the reconstruction and consequently a lot of information has to be exchanged between the parallel nodes. The same holds for high order FD methods.

To conclude, in principle, FV methods are approved schemes for the simulation of flows around complex geometries, but a fundamental problem is to construct a high-order scheme working on unstructured grids.

1.2.3. Finite Element (FE) methods

The philosophy behind the finite element method is somewhat different than the previously discussed discretisation techniques. FE methods take the differential equations, multiply them by an arbitrary test function, and integrate them by parts. The discrete solution is constructed as linear combination of the so called ansatz functions, which often are nothing else than piecewise polynomials. The choice of the ansatz and test function space adjudicates upon which type of FE method is obtained. Typical versions are for example the Galerkin, Petrov-Galerkin or Least-squares FE methods, see for example [56] .

In principle FE methods can be categorised in two major classes of schemes, continuous and discontinuous methods. In contrast to the continuous finite element method, in DG there is no global continuity requirement for ansatz and test functions—leading to the frequently-used term discontinuous FE method. The approximation space is therefore not a subspace of the continuous solution space, or, in other words, the element is non-conforming. Since the discontinuous FE method is the method of choice for this thesis, it will be described in more detail in the next section.

The standard continuous FE methods for the convection-dominated Navier-Stokes equations typically produce oscillations unless artificial dissipation terms are added to the formulation. There are many choices for the stabilisation terms. In the streamline upwind Petrov-Galerkin [22] method for example, a stabilising term is added in the weak formulation, which creates an upwind effect by weighting more heavily the upwind stream nodes within each element [21]. A variety of other methodologies have been proposed to provide additional stability to the convection terms, monotone discrete systems and ease of implementation.

High order versions of the FE method are easy to construct, since the accuracy of the method mainly depends on the degree of chosen polynomial test or ansatz functions. Consequently, the discretisation stencil is compact, because no reconstruction procedure

is necessary. Therefore, these methods are well suited for *hp*-adaptation, where the grid (identified by h) as well as the polynomial degree (identified by p) can be modified in order to increase the solution accuracy.

A disadvantage of the conforming FE discretisation compared to FD and FV is, that if explicit discretisations in time are used, a coupled system of equations has to be solved for every time step. This is due to the coupling of the degrees of freedom at cell interfaces, where continuity requirements have to be fulfilled. In contrast, for the DG method this disadvantagedoes not exist. However DG dpace discretisations have very severe stability limitations for explicit schemes [9, 10].

1.3. Discontinuous Finite Element methods

A relatively new approach in the field of CFD is the so-called Discontinuous Galerkin (DG) approach. In DG, test functions and basis functions used to define the ansatz belong to the same class, where the designation "Galerkin" originates from. Due to the introduction of uncoupled, possibly discontinuous approximations within each element, no coupled system of equations (like for the continuous Galerkin method) has to be solved, if explicit discretisations in time are used. However, due to the double-valued solutions on cell interfaces now the number of degrees of freedom compared to the continuous Galerkin approach is increased. For orders of accuracy less than or equal to 4, the number of unknowns for the DG method is by a factor of more than 2 greater for triangles and nearly 5 greater for tetrahedra than a comparable continuous formulation, see [21].

The development of the DG method dates back to the first introduction in 1973 of Reed and Hill [83]. Le Saint and Raviart [65] analysed DG for linear hyperbolic problems, derived first a priori error estimates and proved rates of convergence. The development went on with handling linear and non-linear hyperbolic systems, where a major part was carried out by Cockburn et al. [29, 28, 26, 31]. They established an approved framework to solve nonlinear, time dependent hyperbolic conservation laws, such as the Euler equations, using explicit Total Variation Diminishing (TVD) Runge-Kutta (RK) time discretisations and DG discretisation in space. These schemes are termed RKDG schemes. The RKDG method is an essentially high-order FE method using ideas of the high-order FV method, such as exact or approximate Riemann solvers to evaluate numerical fluxes, in order to handle discontinuities at the cell interfaces. Recently, Dumbser et al. [41] introduced the ADER-DG approach, which couples the ADER [89] with the spatial DG approach. With the aid of ADER, they developped arbitrary high-order schemes for hyperbolic conservation laws not only in space but also in time.

For a more comprehensive historical overview of DG methods up to May 1999, we want to refer to the article of Cockburn, Karniadakis and Shu, see [27].

The wave propagation properties of the DG method, which especially are of importance for clean aeroacoustic and large eddy simulations, were analysed by Hu et. al [54] and Ainsworth [4].

During the last decade the development of DG has gradually shifted to convection-diffusion problems. The extension of the RKDG method to handle convection-diffusion systems was also developed recently by Cockburn and Shu [32]. Motivated by pioneering work of Bassi and Rebay [13] for the compressible Navier-Stokes equations, they developed the so called local DG (LDG) method [30] and proved stability and convergence with error estimates. The LDG approach can handle higher (≥ 2) derivatives, such as the viscous second order terms in the NS equations.

In the late 1970s and early 1980s, Douglas [39], Baker [11], Wheeler [106] and Arnold [6] developed the so-called interior penalty (IP) discontinuous FE methods for pure elliptic operators. These schemes also work with discontinuous spaces for the test and ansatz functions, but they were not developed as DG methods at that time. Recently, applications of penalty methods to the Navier-Stokes equations were published by Hartmann [50, 51], Baumann and Oden [74, 17], Dolejsi and Feisthauer [38]. In the meantime, Arnold et al. brought penalty methods into a unified DG framework [8].

Another approach not mentioned so far is the space-time discontinuous Galerkin method, introduced by van der Vegt and van der Ven [98, 99] for inviscid compressible flow simulations. As the name lets assume, the space-time DG method uses not only a discontinuous ansatz in space but also an ansatz in time. Due to that, the method is well suited for flow simulations with moving and deforming meshes. The extension of the space-time DG formulation to the compressible NS equations has just been accomplished [59].

For flows with discontinuities, a stabilisation technique like in the case of classical discretisation methods is required, that prevents spurious oscillations. These techniques can be split into two classes. One way consists in supplementing the numerical scheme with an artificial viscosity term, see for example [76], [49]. The other way is concerned with the elaboration of a local projection method or slope limiter to enforce the nonlinear stability, see for example [28], [23]. An alternative is to reduce the scheme to first order and refine the grid near the shock wave or discontinuity. The first order scheme is nonlinearly stable near discontinuities and the reduced accuracy is partially compensated with the aid of grid refinement. In contrast to the continuous Galerkin method, the DG method easily enables so-called *hp*-refinement, where h stands for the grid and p for the order of the scheme [88]. *hp*-refinement uses low-order accurate solution and high grid density (h-refinement) in the neighborhood of discontinuities. For the resolution of smooth but complex flow features and for wave propagation, the order of accuracy of the numerical solution increases (p-refinement) and a relatively coarse mesh can be used. Another important advantage compared to the FV method is, that DG can work on non-conforming grids. Therefore, so-called hanging nodes are allowed in the mesh, which is a desirable feature for flexible and efficient grid adaption. The capability of *hp*-refinement in the DG framework was demonstrated by several authors [49, 97].

The emphasis of the development of DG methods in the last decade was mainly concentrated on the development of a general, stable, consistent, compact discretisation. The analysis of the efficiency, in particular compared to classical methods, has taken a back seat. The development of convergence accelerators, like p-multigrid or *hp*-multigrid has recently been started [45, 72], even if mainly applied to the Euler equations. The

Demand on CFD method	FD	FV	FE	DG
Compressible, robust at shocks	−	+	−−	+
Viscosity—no Euler	++	++	++	+
Low dissipation	++	+	++	++
Moving and/or deforming grids	−	+	+	+
Unstructured grid approach	−−	+	++	++
Compactness of the scheme	−	−−	+	++
Turbulence modeling (RANS, LES, RANS-LES)	+	+	+	+
Adaptivity	−	+	+	++
Grid quality requirement	−−	−	++	++
Memory requirement	++	++	+	−
Computational cost	++	+	−	−−
Programming complexity	++	+	−	−

Table 1.2.: Demands on CFD method and fulfillment by FD, FV, FE and DG approaches

alternative approach for efficient solution, in particular for steady problems, by implicit in time approximate Newton methods was also introduced recently [14, 82].

In the field of DG based turbulent flow simulations only very little experience has been gained to date. A study of DG for the simulation of turbulent flows with the aid of direct numerical simulation (DNS) was performed by Collis [34]. Particularly, the application of DG to the RANS equations has only been reported by Bassi and Rebay so far [15]. For closure of the RANS equations they use the fully coupled $k-\omega$ turbulence model equations. Recently Bassi and Rebay published an extended version of their solution algorithm [16], where realisability conditions were added for the ω-equation in order to increase the numerical robustness of the method.

Finally, we summarised the demands on CFD methods and the fulfillment by FD, FV, FE and DG approaches in table 1.2.

1.4. Motivation and Goals

Based on the statements made in the previous sections, we are now ready to present the goals of this thesis. The long-term objective in (helicopter) CFD is to perform (unsteady) large eddy simulations for (arbitrarily) complex geometries [90]. We want to use the unstructured (hybrid) grid approach for efficient grid generation and grid adaptation, which is important/indispensable for CFD based initial design or initial optimisation work. The computational platform we aim for is a distributed memory standard (Linux) cluster. The temporal and in particular spatial discretisation method for reasonable LES should enable a high-resolution, low dissipative and dispersive method on unstructured grids.

One of currently only very few methods, which does meet the demands of our long-

term objective, see Table 1.2, is the DG approach [101].

Since only moderate practical experience is gained for DG based simulations of the Navier-Stokes equations and in particular Reynolds-averaged Navier-Stokes equations simulations, this will be the main subject of the thesis. The intention of the work is not the development of new physical flow models or new numerical schemes, but the applicability of DG on classical physical models will be analyzed. In detail, the parallelisation, grid requirements, accuracy, robustness etc. will be discussed. The assessment of the efficiency of the method will not be a fundamental part of the thesis.

In consideration of the shown circumstances, the DG discretisation method could have the potential to replace or at least to complement the traditional approaches described in the previous sections. Hopefully, this thesis can give more facts to confirm this statement. Surely, the question whether the Discontinuous Galerkin method will replace the Finite Volume method in future CFD codes cannot be answered now, but this work can contribute some facts to better assess the practical usefulness of DG in industrial day work. This thesis is only the first step towards unstructured high-order simulations of complex flow fields, like that around a helicopter rotor.

The outline of the thesis is the following. In Chapter 2, all classes of physical flow models are briefly described. In Chapter 3, the numerical discretisation of these equations in space by the DG method is discussed in more detail. The chosen approach for time integration of the resulting system of equations is presented in Chapter 4. Chapter 5 deals with the parallel implementation strategy of our DG code. Results for several flow models applied for typical test cases are presented in Chapter 6 and finally the conclusion and outlook is subject of the last Chapter 7.

2. Governing aerodynamic equations

In this chapter all classes of physical flow models, used in this study are briefly described, namely the Euler equations (section 2.1), the Navier-Stokes equations (section 2.2) as wells as the Reynolds-averaged Navier-Stokes (section 2.3) equations. In the last section, we describe the chosen non-dimensionalisation of these equations.

2.1. Euler equations

If viscous flow effects and thermal conduction are neglected, we arrive at the Euler equations of fluid mechanics. These Euler equations in differential form read

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathcal{F}_i(U) = 0 \tag{2.1}$$

where U is the conservative state vector

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \tag{2.2}$$

and $\mathcal{F}_i = (F_i^x,\, F_i^y,\, F_i^z)$ is the inviscid flux tensor with

$$F_i^x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho E + p)\, u \end{pmatrix},\; F_i^y = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ (\rho E + p)\, v \end{pmatrix},\; F_i^z = \begin{pmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ (\rho E + p)\, w \end{pmatrix} \tag{2.3}$$

The specific total energy E is composed of the specific internal energy e and specific kinetic energy

$$E = e + \frac{1}{2}\left(u^2 + v^2 + w^2\right)$$

Furthermore, the thermally and calorically perfect gas assumption is made, for which the following relations for the total enthalpy H holds

$$H = E + \frac{p}{\rho},$$

and the equation of state reads

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right) = \frac{\gamma - 1}{\gamma}\left(\rho H - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)\right)$$

where the specific heat ratio γ is assumed to be constant and equal to 1.4.

2.2. Navier-Stokes equations

If we also include the effects of viscosity and thermal conduction we have to expand the Euler equations by the diffusive flux tensor $\mathcal{F}_v$

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathcal{F}_i(U) = \nabla \cdot \mathcal{F}_v(U, \nabla U) \tag{2.4}$$

The extra term $\nabla \cdot \mathcal{F}_v(U)$ in (2.4) is the divergence of the viscous flux tensor $\mathcal{F}_v = (F_v^x, F_v^y, F_v^z)$, defined by

$$\begin{aligned} F_v^x &= \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x \end{pmatrix}, F_v^y = \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + q_y \end{pmatrix} \\ F_v^z &= \begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + q_z \end{pmatrix} \end{aligned} \tag{2.5}$$

where τ is a tensor as well, the viscous stress tensor

$$\tau = \begin{pmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{pmatrix}$$

In this work air is considered, which is assumed to be a Newtonian fluid for which the Stokes hypothesis is valid. Then τ is symmetric and a linear function of the velocity gradients

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) \tag{2.6}$$

where μ is the molecular viscosity coefficient, also referred to as the dynamic viscosity coefficient and δ_{ij} is the Kronecker delta function. All other symbols have their usual meaning. According to kinetic gas theory, μ is only a function of temperature, if a monoatomic gas is considered. Although air is a mixture of mainly diatomic gases, this result is also valid for air at moderate temperatures. In all computations, the semi-empirical formula of Sutherland is used:

$$\frac{\mu}{\mu_0} = \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + S}{T + S} \tag{2.7}$$

$$\begin{aligned} \mu_0 &= 1.716 \cdot 10^{-5} \, \frac{kg}{m\,s} \\ T_0 &= 273.15\,K \\ S &= 110.55\,K \end{aligned}$$

The last row of the viscous flux tensor $\mathcal{F}_v$, the energy flux part, contains the heat flux vector q, which is modeled according to Fourier's law:

$$q_j = -\lambda \frac{\partial T}{\partial x_j} \tag{2.8}$$

where the thermal conductivity coefficient λ is related to the molecular viscosity coefficient μ and the specific heat capacity at constant pressure, c_p by the non-dimensional Prandtl number Pr:

$$\lambda = \frac{\mu c_p}{Pr}$$

For air the Prandtl number is approximately constant for temperatures between 200 K and 600 K and equal to 0.72. This approximation holds for all computational test cases presented in this work.

2.3. Reynolds-averaged Navier-Stokes equations and closure models

Most of aeronautical flows are turbulent, which complicates the simulation of the flow field significantly. In principle, the turbulent flow is a solution of the Navier-Stokes equations (2.4), but the number of grid cells needed to resolve all turbulent length scales for high Reynolds number flows is far beyond current and foreseeable computer resources [90]. Kolmogorov showed, that the ratio between the largest turbulent scale and the Kolmogorov micro scale is $Re^{3/4}$. From this it follows that we need approximately $O(Re^{3/4})$ grid cells per space dimension to resolve all length scales properly. Because turbulence is essentially a three-dimensional phenomenon, we need approximately $O(Re^{9/4})$ grid cells for our computations. For aeronautical flows, Reynoldsnumbers of several million are

common practice. For example, consider an airliner in cruise condition, with a Reynolds number of $Re = 10^7$, we roughly need $O(10^{16})$ computing cells. This is by far beyond the performance of current and near-future supercomputers. Thus a suitable approximation of the time dependent NS equations has to be taken. One practical possibility to compute such high Reynolds number turbulent flows is to solve the equations for the mean values of the flow quantities. This approach, well known as the method of Reynolds averaging, will be discussed in the next section.

2.3.1. Reynolds and Favre averaging

In this section the Reynolds and Favre averaging of the Navier-Stokes equations is introduced very briefly. Reynolds' approach to treat turbulent flows approximately is based on a decomposition of the flow variables. A turbulent flow quantity $A(\vec{x}, t)$ is rewritten as

$$A = \overline{A} + A' \tag{2.9}$$

with an overbar denoting the mean value, while the fluctuating part is marked by a prime. The mean value represents the time-averaged quantity of A, defined by

$$\overline{A} \equiv \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} A \, dt$$

We require that Δt be large compared to the period of the random fluctuations associated with the turbulence. The Δt is sometimes indicated to approach infinity as a limit, but this should be interpreted as being relative to the characteristic fluctuation period of turbulence.

If unsteady flows need to be modelled, Δt has to be small with respect to the time constant for slow variations in the flow field, which need to be resolved with a numerical simulation. This condition has to be fulfilled, if unsteady Reynolds-averaged NS simulations, also called URANS simulations, are to be performed meaningfully.

Since compressibility effects have to be taken into account, for most aircraft applications mass averaging according to Favre is applied [43]. In this approach mass-averaged quantities are defined according to

$$\tilde{A} = \frac{\overline{\rho A}}{\overline{\rho}}$$

The decomposition, in contrast to (2.9), is written know as

$$A = \tilde{A} + A''$$

where the tilde denotes the mean value while the double prime denotes the fluctuation part of A.

The next step is substituting the mass-averaged variables plus the double primed fluctuations into the Navier-Stokes equations (2.4). Then one averages the entire equations

in time and uses appropriate averaging identities for simplification. The resulting set of equations are called mass-weighted Reynolds-averaged Navier-Stokes (RANS) equations, which govern the time-mean motion of the fluid. The RANS equations are formally nearly identical to the system of time dependent Navier-Stokes equations. The averaging process generates some extra terms (compared to the NS equations), which can be interpreted as apparent stresses and apparent heat fluxes. In detail, we have

$$(\tau_{ij})_{\text{turb}} = -\ \overline{\rho u_i'' u_j''} \tag{2.10}$$

$$(q_j)_{\text{turb}} = c_p \overline{\rho T'' u_j''}. \tag{2.11}$$

There cannot be derived any physically justified explicit equations for the correlations in (2.10) and (2.11). This is the so called closure problem of the RANS approach. The averages of fluctuating quantities in (2.10) and (2.11) must be modelled as functions of the mean flow quantities. To date, most modelling of turbulent compressible flow is based on the Boussinesq hypothesis, which states that the Reynolds stress tensor is linearly related to the mean strain rate

$$\overline{\rho u_i'' u_j''} = \mu_t \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3} \frac{\partial \tilde{u}_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \delta_{ij} \rho k \tag{2.12}$$

The proportionality factor μ_t is the eddy viscosity, also referred to as the turbulent viscosity, and k is the kinetic energy of turbulence $k = \frac{1}{2}\overline{u_i'' u_i''}$. The last term is present due to the fact that the turbulent normal stresses must sum up to $-2\rho k$.

The turbulent heat transfer is assumed to be proportional to the mean temperature gradient, so that

$$c_p \overline{\rho T'' u_j''} = -\lambda_t \frac{\partial \tilde{T}}{\partial x_j} \tag{2.13}$$

The turbulent thermal conductivity, λ_t is related to the eddy viscosity by $\lambda_t = \frac{\mu_t c_p}{Pr_t}$. Here Pr_t is the so called turbulent Prandtl number, which is taken constant and equal to 0.90.

Henceforth, in order to ease the appearance of the RANS equations we drop the $\tilde{}$ and $\bar{}$ to denote mean quantities. For compressible turbulent flow, with the assumption that the Reynolds stresses and the turbulent heat flux can be approximated with the mean flow quantities via (2.12) and (2.13), the governing equations become

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathcal{F}_i(U) = \nabla \cdot \mathcal{F}_v(U, \nabla U) \tag{2.14}$$

with $\mathcal{F}_i$ and $\mathcal{F}_v$ according to (2.3) and (2.5). But τ_{ij} and q_j are modified to the effective turbulent quantities as

$$\tau_{ij} = (\mu + \mu_t) \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \rho k \delta_{ij} \tag{2.15}$$

$$q_j = -\left(\lambda + \lambda_t\right) \frac{\partial T}{\partial x_j} \tag{2.16}$$

The introduction of eddy viscosity and turbulent thermal conductivity permits nearly the same form of the governing equations to be used as for laminar flow, if μ and λ are replaced by $\mu + \mu_t$ and $\lambda + \lambda_t$, respectively. Since λ_t is a function of μ_t, we still require a relation for the eddy viscosity in order to close the system. This is subject of the next sections, where the eddy viscosity models used in this thesis will be described briefly.

2.3.2. Wilcox $\mathrm{k} - \omega$ turbulence model

The $\mathrm{k} - \omega$ model from Wilcox [107] can be insorted into the class of two-equation turbulence models. These models provide not only for computation of the turbulent velocity scale, but also for the turbulent length scale. Hence, two-equation models are qualified as to be complete, that is, one can use them to predict properties of a given turbulent flow with no prior knowledge of the turbulence structure. Like in the majority of cases, the turbulent kinetic energy per unit mass ($\sqrt{k}$) is chosen as the turbulent velocity scale v_t. In order to account for a length scale l_t the specific dissipation rate ω is adopted, which is related to the length scale by $l_t = \sqrt{k}/\omega$. The space-time behavior of k and ω is modeled in two coupled partial differential equations, (2.17), (2.18). The general configuration of both transport equations is equal. The terms I indicate the time derivatives, II convective parts, III the diffusion terms, IV the production terms and V the destruction terms.

Turbulent kinetic energy:

$$\underbrace{\frac{\partial \rho k}{\partial t}}_{I} + \underbrace{\frac{\partial \rho k u_i}{\partial x_i}}_{II} = \underbrace{\frac{\partial}{\partial x_i}\left[\left(\mu + \sigma_{k1}\mu_t\right)\frac{\partial k}{\partial x_i}\right]}_{III} + \underbrace{\sigma_{ij}\frac{\partial u_i}{\partial x_j}}_{IV} - \underbrace{\beta^* \rho \omega k}_{V} \tag{2.17}$$

Specific dissipation rate:

$$\underbrace{\frac{\partial \rho \omega}{\partial t}}_{I} + \underbrace{\frac{\partial \rho \omega u_i}{\partial x_i}}_{II} = \underbrace{\frac{\partial}{\partial x_i}\left[\left(\mu + \sigma_{\omega 1}\mu_t\right)\frac{\partial \omega}{\partial x_i}\right]}_{III} + \underbrace{\gamma_1 \frac{\omega}{k}\sigma_{ij}\frac{\partial u_i}{\partial x_j}}_{IV} - \underbrace{\beta_1 \rho \omega^2}_{V} \tag{2.18}$$

The turbulent stress tensor possesses the following form

$$\sigma_{ij} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial \bar{x}_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k \delta_{ij}$$

The nondimensional eddy viscosity is defined as:

$$\mu_t = \frac{\rho k}{\omega}$$

The calibrated constants of the model are [107]:

$$\sigma_{k1} = 0.5, \quad \sigma_{\omega 1} = 0.5, \quad \beta_1 = 0.0075$$
$$\beta^* = 0.09, \quad \gamma_1 = 5/9.$$

We prescribe the following no-slip wall boundary conditions for ω and k according to Menter [71].

$$\begin{aligned} \omega_{\text{wall}} &= \frac{60\mu}{\rho\beta_1 \left(\bigtriangleup y_1\right)^2} \\ k_{\text{wall}} &= 0 \end{aligned} \tag{2.19}$$

where $\bigtriangleup y_1$ is the distance of the first grid point from the wall. In our implementation, Δy_1 is equal to the height of the wall boundary triangle or quadrilateral, respectively.

At farfield boundaries we determine k from the freestream turbulence intensity Tu_∞, which can be specified by the user

$$k_\infty = \frac{3}{2}\text{Tu}_\infty \left(\vec{v}_\infty\right)^2, \quad (\text{default: } \text{Tu}_\infty = 0.005)$$

The value of ω is calculated from the user defined ratio of eddy viscosity μ_t to molecular viscosity μ depicted as r_μ.

$$\omega_\infty = \frac{\rho_\infty k_\infty}{r_{\mu,\infty}\mu_\infty} \quad \left(\text{default: } r_{\mu,\infty} = 10^{-5}\right)$$

For stability and positivity reasons, we used a special formulation [15] of the model. The equation for ω is reformulated using $\tilde{\omega} = \ln(\omega)$ as working variable instead of the original ω. Hence, ω is guaranteed to be positive, and in addition the near wall distribution of $\tilde{\omega}$ is much more smooth than that of ω. Replacing all appearances of ω in equation (2.18) by $e^{\tilde{\omega}}$ and applying product rule (2.18) the left hand side can be expressed as

$$e^{\tilde{\omega}} \left(\underbrace{\frac{\partial \rho}{\partial t} + \frac{\partial \left(\rho u_i\right)}{\partial x_i}}_{=0} + \frac{\partial \left(\rho\tilde{\omega}\right)}{\partial t} + \frac{\partial \left(\rho u_i \tilde{\omega}\right)}{\partial x_i} - \tilde{\omega} \left(\underbrace{\frac{\partial \rho}{\partial t} + \frac{\partial \left(\rho u_i\right)}{\partial x_i}}_{=0} \right) \right) = e^{\tilde{\omega}} \left(\frac{\partial \left(\rho\tilde{\omega}\right)}{\partial t} + \frac{\partial \left(\rho u_i \tilde{\omega}\right)}{\partial x_i} \right) \tag{2.20}$$

The diffusion term on the right hand side becomes

$$\frac{\partial \left(\left(\rho\mu + \sigma_{\omega 1}\mu_t\right) e^{\tilde{\omega}} \frac{\partial \tilde{\omega}}{\partial x_i} \right)}{\partial x_i} = e^{\tilde{\omega}} \left(\rho\mu + \sigma_{\omega 1}\mu_t\right) \left(\frac{\partial}{\partial x_i}\frac{\partial \tilde{\omega}}{\partial x_i} + \frac{\partial \tilde{\omega}}{\partial x_i}\frac{\partial \tilde{\omega}}{\partial x_i} \right) \tag{2.21}$$

The production and destruction terms become

$$e^{\tilde{\omega}}\left(\gamma_1 \frac{1}{k}\sigma_{ij}\frac{\partial u_i}{\partial x_j} - \beta_1 \rho e^{\tilde{\omega}}\right) \tag{2.22}$$

Combining (2.20), (2.21) and (2.22) we obtain the $\ln(\omega)$-formulation of the original ω equation

$$\frac{\partial\left(\rho\tilde{\omega}\right)}{\partial t} + \frac{\partial\left(\rho u_i \tilde{\omega}\right)}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\mu + \sigma_{\omega 1}\mu_t\right)\frac{\partial \tilde{\omega}}{\partial x_i}\right] + \left(\mu + \sigma_{\omega 1}\mu_t\right)\frac{\partial \tilde{\omega}}{\partial x_i}\frac{\partial \tilde{\omega}}{\partial x_i} + \gamma_1 \frac{1}{k}\sigma_{ij}\frac{\partial u_i}{\partial x_j} - \beta_1 \rho e^{\tilde{\omega}} \tag{2.23}$$

Transition is handled by activating and deactivating the respective source terms in the user specified regions. Since k can become negative, we limit it to zero, in order to ensure positivity

$$k_{lim} = \max\left(0, k\right).$$

The limitation of turbulence quantities like k in the discontinuous Galerkin framework will be described in full detail in section 3.3.

2.3.3. Spalart-Allmaras turbulence model

The Spalart-Allmaras model [91, 92] is a one-equation model, which is developed especially for aerodynamic flows. Unlike early one-equation models, which are based on the transport equation of the turbulent kinetic energy k, Spalart and Allmaras developed a model equation for the eddy viscosity itself. The model inherently provides the necessary turbulent velocity and length scale and is thus complete. The model proved to be superior to algebraic modells, and in addition the computational effort, compared with two-equation modells, like the $k-\omega$ model described in the previous section, is roughly halfed. The eddy viscosity model, composed of the Lagrangian derivative $(I + II)$, a diffusion part (III), a production part (IV) , destruction part (V) and a so-called trip term (VI) is given in equation (2.24). Transport of eddy viscosity:

$$\begin{aligned}\underbrace{\frac{\partial \rho\tilde{\nu}}{\partial t}}_{I} + \underbrace{\frac{\partial \rho\tilde{\nu}u_i}{\partial x_i}}_{II} = \underbrace{\frac{1}{\sigma}\left[\frac{\partial}{\partial x_i}\left(\left(\mu + \rho\tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial x_i}\right) + \rho c_{b2}\frac{\partial\tilde{\nu}}{\partial x_i}\frac{\partial\tilde{\nu}}{\partial x_i}\right]}_{III} + \\ \underbrace{c_{b1}\left(1 - f_{t2}\right)\rho\tilde{S}\tilde{\nu}}_{IV} - \underbrace{\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\frac{1}{\rho}\left(\frac{\rho\tilde{\nu}}{D}\right)^2}_{V} + \underbrace{\rho f_{t1}\Delta U^2}_{VI}\end{aligned} \tag{2.24}$$

The eddy viscosity is related to the working variable $\tilde{\nu}$ as

$$\mu_t = \rho\nu_t = \rho\tilde{\nu}f_{v1}$$

The production term is proportional to

$$\tilde{S} = |\Omega| + \frac{\tilde{\nu}}{\kappa^2 D^2} f_{v2} \tag{2.25}$$

where $|\Omega|$ is the magnitude of the vorticity vector Ω

$$\Omega = \begin{pmatrix} \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \\ \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \end{pmatrix}$$

and D the distance to the nearest wall. The need of the wall distance D is a fundamental difference compared to the $k-\omega$ model which could manage without information about D.

The definition of the model is completed by the auxiliary relations:

$$f_w = \left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6} \right)^{\frac{1}{6}}, \tag{2.26}$$

$$g = r + c_{w2}\left(r^6 - r\right), \tag{2.27}$$

$$r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 D^2}, 10 \right), \tag{2.28}$$

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}, \tag{2.29}$$

$$f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}, \tag{2.30}$$

$$f_{t1} = c_{t1} g_t \exp\left(-c_{t2} \frac{\Omega_t^2}{\Delta U^2} \left[D^2 + g_t^2 D_t^2 \right] \right), \tag{2.31}$$

$$g_t = \min\left(0.1, \frac{\Delta U}{|\Omega_t| \Delta x_t} \right), \tag{2.32}$$

$$f_{t2} = c_{t3} \exp\left(-c_{t4} \chi^2 \right). \tag{2.33}$$

The model requires the knowledge of the transition points, either by an educated guess or by knowing the experimental trip points. The calibration functions f_{t1} and f_{t2} are designed to perform transition from laminar to turbulent flow at the prescribed positions.

Here $|\Omega_t|$ is the magnitude of the vorticity of the nearest trip point , ΔU is the velocity difference with respect to the nearest trip point and D_t is the distance to this point.

Alternatively, it is possible to set $f_{t1} = f_{t2} = 0$ and let the transition take place due to numerical reasons. In this work, transition is handled by activating and deactivating the respective source terms in the user specified regions.

The model is closed by the constants:

$$\begin{array}{cccc} c_{b1} = 0.1355, & c_{b2} = 0.622, & \sigma = \frac{2}{3}, & \kappa = 0.41, \\ c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma}, & c_{w2} = 0.3, & c_{w3} = 2.0, & c_{v1} = 7.1, \\ c_{t1} = 1.0, & c_{t2} = 2.0, & c_{t3} = 1.2, & c_{t4} = 0.5. \end{array}$$

The no-slip boundary condition for the working variable $\tilde{\nu}$ is simply

$$\tilde{\nu}_{\text{wall}} = 0.$$

and the freestream or farfield value for $\tilde{\nu}$ in this work is set to

$$\tilde{\nu}_\infty = \frac{\nu_\infty}{100}.$$

2.4. Nondimensional form of equations

In order to reduce the errors due to finite precision of computers and in addition to decrease the condition number of the linear system, if an implicit time integration method is performed, we have to be sure that all variables are approximately of the same order of magnitude. This can be achieved by normalising the governing equations. For the normalisation we use the following five reference quantities: length, density, velocity, viscosity and temperature. The choice of the respective quantities is summarised in table 2.1.

Quantity		Reference	
Length	L_{ref}	Problem dependent (chord length, cylinder diameter, etc.)	c, d
Density	ρ_{ref}	Freestream density	ρ_∞
Velocity	v_{ref}	Freestream speed of sound	a_∞
Viscosity	μ_{ref}	Freestream viscosity	μ_∞
Temperature	T_{ref}	Freestream temperature	T_∞

Table 2.1.: Reference quantities used for non-dimensionalisation

If the non-dimensional quantities are depicted with an overbar ($^-$), further desired normalised quantities are referenced in the following manner:

$$\bar{t} = \frac{a_\infty t}{L_{ref}}, \quad \bar{p} = \frac{p}{\rho a_\infty^2}, \quad \bar{e} = \frac{e}{a_\infty^2}, \quad \bar{H} = \frac{H}{a_\infty^2}, \quad \bar{a} = \frac{a}{a_\infty}, \quad \bar{k} = \frac{k}{a_\infty^2}, \quad \bar{\omega} = \omega \frac{L_{ref}}{a_\infty}, \quad \bar{\tilde{\nu}} = \frac{\tilde{\nu}}{\mu_\infty}$$

The nondimensional equation of state reads:

$$\bar{p} = \frac{1}{\gamma} \bar{\rho} \bar{T}$$

The dimensionless reference numbers are defined as

- Mach number $Ma_\infty = \frac{U\infty}{a_\infty}$
- Reynoldsnumber[1] $Re_\infty = \frac{\rho_\infty a_\infty L_{ref}}{\mu_\infty}$
- Prandtl number $Pr = \frac{\mu_\infty c_p}{\lambda}$

In the normalised main flow equations, the viscous stress tensor (2.6) and the heat flux (2.8) have been slightly changed:

$$\bar{\tau}_{ij} = \frac{1}{Re_\infty}\bar{\mu}\left(\frac{\partial \bar{u}_i}{\partial \bar{x}_j} + \frac{\partial \bar{u}_j}{\partial \bar{x}_i} - \frac{2}{3}\frac{\partial \bar{u}_k}{\partial \bar{x}_k}\delta_{ij}\right)$$

$$\bar{q}_j = -\frac{1}{Re_\infty Pr}\frac{\bar{\mu}}{(\gamma-1)}\frac{\partial \bar{T}}{\partial \bar{x}_j}$$

In the Wilcox $k-\omega$ turbulence model all normalised diffusive terms multiply by $\frac{1}{Re_\infty}$ now

$$\frac{\partial \bar{\rho}\bar{k}}{\partial \bar{t}} + \frac{\partial \bar{\rho}\bar{k}\bar{u}_i}{\partial \bar{x}_i} = \frac{1}{Re_\infty}\frac{\partial}{\partial \bar{x}_i}\left[\left(\bar{\mu} + \sigma_{k1}\bar{\mu}_t\right)\frac{\partial \bar{k}}{\partial \bar{x}_i}\right] + \sigma_{ij}\frac{\partial \bar{u}_i}{\partial \bar{x}_j} - \beta^*\bar{\rho}e^{\bar{\tilde{\omega}}}\bar{k} \tag{2.34}$$

$$\begin{aligned}\frac{\partial\left(\bar{\rho}\bar{\tilde{\omega}}\right)}{\partial \bar{t}} + \frac{\partial\left(\bar{\rho}\bar{u}_i\bar{\tilde{\omega}}\right)}{\partial \bar{x}_i} &= \frac{1}{Re_\infty}\frac{\partial}{\partial \bar{x}_i}\left[\left(\bar{\mu} + \sigma_{\omega 1}\bar{\mu}_t\right)\frac{\partial \bar{\tilde{\omega}}}{\partial \bar{x}_i}\right] + \frac{\left(\bar{\mu} + \sigma_{\omega 1}\bar{\mu}_t\right)}{Re_\infty}\frac{\partial \bar{\tilde{\omega}}}{\partial \bar{x}_i}\frac{\partial \bar{\tilde{\omega}}}{\partial \bar{x}_i} \\ &\quad + \gamma_1\frac{1}{\bar{k}}\sigma_{ij}\frac{\partial \bar{u}_i}{\partial \bar{x}_j} - \beta_1\bar{\rho}e^{\bar{\tilde{\omega}}}\end{aligned} \tag{2.35}$$

The normalised turbulent stress tensor and the dimensionless eddy viscosity are defined as:

$$\bar{\sigma}_{ij} = \frac{1}{Re_\infty}\bar{\mu}_t\left(\frac{\partial \bar{u}_i}{\partial \bar{x}_j} + \frac{\partial \bar{u}_j}{\partial \bar{x}_i} - \frac{2}{3}\frac{\partial \bar{u}_k}{\partial \bar{x}_k}\delta_{ij}\right) - \frac{2}{3}\bar{\rho}\bar{k}\delta_{ij}$$

$$\bar{\mu}_t = Re_\infty\bar{\rho}\bar{k}e^{-\bar{\tilde{\omega}}}.$$

The normalised wall boundary conditon for $\tilde{\omega}$ reads

$$\bar{\tilde{\omega}}_{\text{wall}} = \ln\left(\frac{60\bar{\mu}}{Re_\infty\bar{\rho}\beta_1\left(\bigtriangleup \bar{y}_1\right)^2}\right) \tag{2.36}$$

[1]The classical farfield Reynolds number is linked to the reference Reynolds number by $Re_{\infty,\text{classical}} = Re_\infty Ma_\infty$

For the SA model, the equations (2.24), (2.25) and (2.28) modify to

$$\frac{\partial \bar{\rho}\bar{\tilde{\nu}}}{\partial \bar{t}} + \frac{\partial \bar{\rho}\bar{\tilde{\nu}}u_i}{\partial \bar{x}_i} = \frac{1}{Re_\infty}\frac{1}{\sigma}\left[\frac{\partial}{\partial \bar{x}_i}\left((\bar{\mu}+\bar{\rho}\bar{\tilde{\nu}})\frac{\partial \bar{\tilde{\nu}}}{\partial \bar{x}_i}\right) + \bar{\rho}c_{b2}\frac{\partial \bar{\tilde{\nu}}}{\partial \bar{x}_i}\frac{\partial \bar{\tilde{\nu}}}{\partial \bar{x}_i}\right] +$$
$$c_{b1}\left(1-f_{t2}\right)\bar{\rho}\bar{\tilde{S}}\bar{\tilde{\nu}} - \frac{1}{Re_\infty}\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\frac{1}{\bar{\rho}}\left(\frac{\bar{\rho}\bar{\tilde{\nu}}}{\bar{D}}\right)^2 + \bar{\rho}Re_\infty f_{t1}\Delta\bar{U}^2 \qquad (2.37)$$

$$\bar{\tilde{S}} = \left|\bar{\Omega}\right| + \frac{1}{Re_\infty}\frac{\bar{\tilde{\nu}}}{\kappa^2\bar{D}^2}f_{v2} \qquad (2.38)$$

$$\bar{r} = \min\left(\frac{\bar{\tilde{\nu}}}{Re_\infty\bar{\tilde{S}}\kappa^2\bar{D}^2}, 10\right). \qquad (2.39)$$

The desired nondimensional quantities for the (freestream) initialisation of the flowfield are:

$$\begin{aligned}
\bar{\rho}_\infty &= 1.0 \\
\begin{pmatrix} \bar{\rho}\bar{u} \\ \bar{\rho}\bar{v} \\ \bar{\rho}\bar{w} \end{pmatrix}_\infty &= \mathrm{Ma}_\infty \begin{pmatrix} \cos\,\alpha\cos\beta \\ \cos\alpha\,\sin\beta \\ \sin\alpha \end{pmatrix} \\
\bar{\rho}_\infty\bar{E}_\infty &= \frac{1}{\gamma\left(\gamma-1\right)} + \frac{1}{2}Ma_\infty^2
\end{aligned}$$

where α is the angle of attack and β stands for the yaw angle.

For turbulent simulations with the two-equation model of Wilcox, the freestream values for k and ω are prescribed by:

$$\begin{aligned}
\bar{k}_\infty &= \frac{3}{2}\mathrm{Tu}_\infty Ma_\infty^2, \quad (\text{default: } \mathrm{Tu}_\infty = 0.005) \\
\bar{\omega}_\infty &= Re_\infty\frac{\bar{k}_\infty}{r_\mu} \quad \left(\text{default: } r_\mu = 10^{-5}\right)
\end{aligned}$$

The nondimensional working variable $\tilde{\nu}$ is initialised with

$$\bar{\tilde{\nu}}_\infty = 0.01.$$

The dimensionless Sutherland formula for the molecular viscosity reads:

$$\bar{\mu}(\bar{T}) = \bar{T}^{\frac{3}{2}}\frac{1+S/T_\infty}{\bar{T}+S/T_\infty}$$

The thermal conductivity coefficient is computed with

$$\bar{\lambda}(\bar{\mu}) = \frac{\gamma}{\gamma - 1}\frac{1}{Pr}\bar{\mu}, \text{ where } \bar{q}_j = -\frac{1}{Re_\infty}\bar{\lambda}\frac{\partial \bar{T}}{\partial \bar{x}_j}$$

Other dimensionless freestream quantities are:

$$\begin{aligned} \bar{p}_\infty &= \frac{1}{\kappa} \\ \bar{T}_\infty &= 1.0 \\ \bar{\mu}_\infty &= 1.0 \end{aligned}$$

3. Discontinuous Galerkin discretisation in space

In the present work we use the DG method only to discretise the governing equations of chapter 2 in space. In principle, it is possible to apply the DG approach in space and time, but in this work we use classical explicit and implicit integration schemes (see chapter 4) for time marching. The interested reader is referred for example to the papers of van der Vegt and van der Ven [98, 59] for the details of the space-time DG method.

The outline of this Chapter is the following. In section 3.1 and section 3.2, the DG based spatial discretisation methods for the Euler equations and the NS equations used are described in full detail. A special section 3.3 is dedicated to the peculiarities concerning the discretisation of the respective turbulence models. Section 3.4 deals with the weak handling of boundary conditions in the DG framework. In section 3.5 the chosen space for the basis and test functions for several grid topologies as well as the used geometrical transformation laws and integration rules are described. The subject of the last section 3.6 is the geometrically high-order treatment of curved surface geometries.

3.1. Formulation for the Euler equations

Multiplying the differential form of the Euler equations (2.1) with an arbitrary test or weight function v and integrating over a domain Ω we obtain its weighted residual form

$$\int_{\Omega} v \left(\frac{\partial U}{\partial t} + \nabla \cdot \mathcal{F}_i(U) \right) d\Omega = 0 \tag{3.1}$$

Now we perform integration by parts (Gaussian divergence theorem in higher dimension) on the advection term and we get the basic form of the DG approach for the system of Euler equations—the weak formulation of the problem (2.1).

$$\int_{\Omega} \left(v \frac{\partial U}{\partial t} \right) d\Omega + \oint_{\partial\Omega} v \mathcal{F}_i \cdot \vec{n} \, d\sigma - \int_{\Omega} \nabla v \cdot \mathcal{F}_i d\Omega = 0 \tag{3.2}$$

where $\partial\Omega$ denotes the boundary of the solution domain Ω and $\vec{n}$ is the outward pointing normal unit vector.

We split the domain into a collection of arbitrary non overlapping elements E

$$\mathcal{T}_h = \{E\}$$

The (discretisation) elements E, we will work with, are lines in one dimension and

triangles or quadrilaterals in two spatial dimensions. Now, equation (3.2) can be written as summation over all elements E

$$\sum_E \left[\int_E \left(v \frac{\partial U}{\partial t} \right) d\Omega + \oint_{\partial E} v \mathcal{F}_i \cdot \vec{n} \, d\sigma - \int_E \nabla v \cdot \mathcal{F}_i d\Omega \right] = 0$$

In order to derive a discretisation in space we have to identify a test function v and we have to choose an ansatz for the numerical solution u_h. We first define the finite element space V_h:

$$V_h := \left\{ v_h \in L^2(\Omega) : v \mid_{hE} \in P^k(E) \quad \forall E \in \mathcal{T}_h \right\}$$

Here, $P^k(E)$ denotes the local space of polynomial functions of degree at most k

$$P^k := \{ p(x) \mid p(x) \text{ is polynomial of degree } \leq k \}.$$

$L^2(\Omega)$ represents the space of functions, which are squared Lebesgue integrable over the domain Ω.

In this work we choose the following approach. The solution u inside each element is approximated by a linear combination of the test functions v:

$$u(x,t)_h = \sum_{k=0}^{n} U_k(t) b_k(x), \tag{3.3}$$

$$v(x)_h = b_k(x), \; k = 0..n \tag{3.4}$$

where the expansion coefficients $U_k(t)$ denote the degrees of freedom (DOF) of the numerical solution in an element E. The $n+1$ shape or basis functions $b_k(x)$ are a base for the polynomial functions P^k. Note, that a separation ansatz in space and time is used for the solution u_h, where the degrees of freedom $U_k = U_k(t)$ are functions of time and the basis functions $b_k = b_k(x)$ are functions of space only. A DG approach in space and time would result in scalar degrees of freedom U_k and basis functions which are dependent in both, space and time $b_k = b_k(x,t)$. This method is known as the space-time DG method, see [98, 59] for details. Polynomial expansions are used traditionally for FE methods, because polynomials naturally correspond to Taylor series expansions of analytical functions and polynomials can be easily integrated with discrete integration rules. The chosen polynomial basis functions for the element types used in this work are described in section 3.5. As we can observe in (3.3) and (3.4), we chose identical spaces for the test and basis functions, which is the typical case of the Galerkin finite element scheme. As described in section 1.3, in contrast to the Galerkin finite element method in the DG framework there is no global continuity requirement for u_h and v_h—leading to the frequently used term discontinuous Galerkin method, see figure 3.1. Another interesting fact is, that we arrive at the classical first order cell-centered finite volume scheme, if we choose a constant ansatz DG by using P^0elements and the single trivial test function $v = 1$.

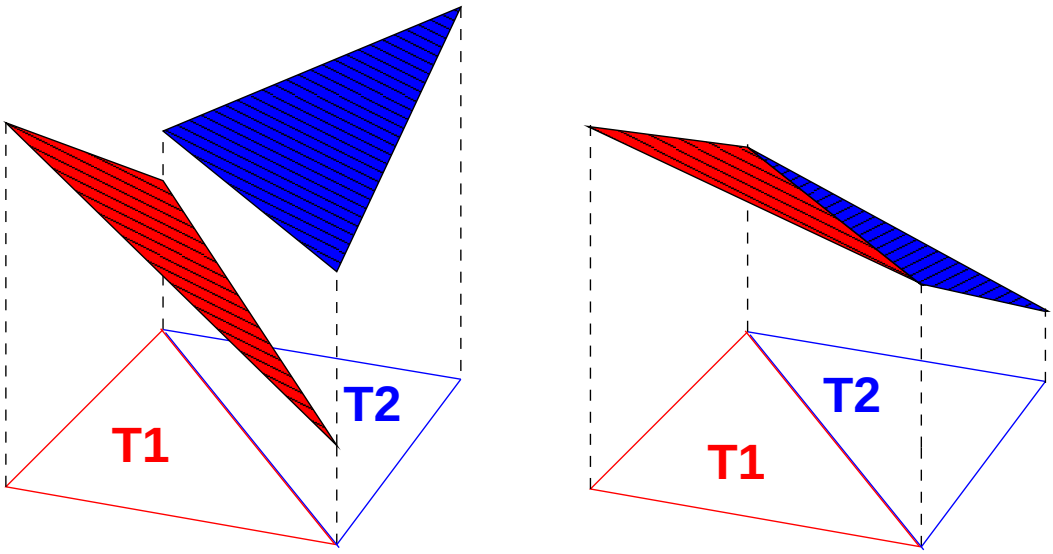


Figure 3.1.: Discontinuous (left) and continuous (right) Galerkin method interim solutions, two triangular P^1-elements T1 and T2.

By admitting only the functions u_h and v_h defined by (3.3) and (3.4) we obtain the following semi-discrete system of n equations for a generic element E:

$$\frac{d}{dt}\underbrace{\int_E b_k u_h d\Omega}_{I} + \underbrace{\oint_{\partial E} b_k \mathcal{F}_i \cdot \vec{n}\, d\sigma}_{II} - \int_E \nabla b_k \cdot \mathcal{F}_i d\Omega = 0,\ 0 \leq k \leq n \tag{3.5}$$

The first volume integral (part I of (3.5)) is generally written as the product $M \cdot U$, where M is the element mass matrix and $U = (U_1\, U_2\, \ldots U_n)^T$ is the vector composed of the solutions degrees of freedom. The elements of M are defined as

$$M_{kj} = \int_E b_k b_j d\Omega.$$

In the case of orthogonal basis functions

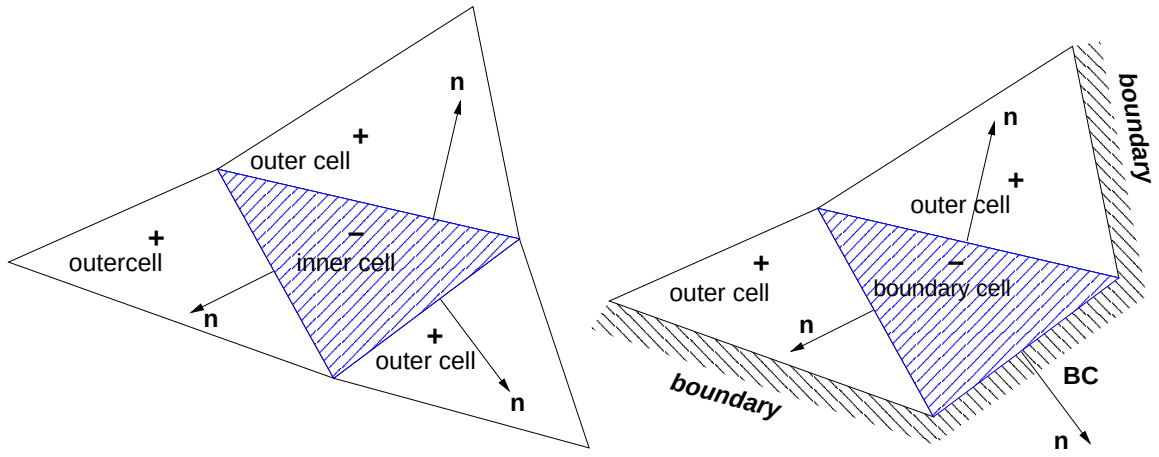
$$\int_E b_k b_j d\Omega = 0,\ k \neq j$$

the elemental mass matrices possess diagonal form.

Since the solution u_h is allowed to be discontinuous at the element boundaries, the inner face integral (part II) is handled with a numerical flux $\mathcal{H}_i\left(u_h^-, u_h^+\right)$, which approximates the physical flux

$$\oint_{\partial E} b_k \mathcal{F}_i \cdot \vec{n}\, \mathrm{d}\sigma \quad \approx \quad \oint_{\partial E} b_k \mathcal{H}_i(u_h^-, u_h^+) \cdot \vec{n}\, \mathrm{d}\sigma.$$

Here, the $(\cdot)^-$ and $(\cdot)^+$ notation is used to indicate the trace value taken from the interior and exterior of the element, respectively (see figure 3.2(a)).



(a) Inner cell

(b) Boundary cell

Figure 3.2.: Outward pointing normal vector and interior and exterior cell convention

The final form of the DG discretisation for the Euler equations is

$$\frac{d}{dt}\int_E b_k u_h d\Omega + \underbrace{\sum_{e\in\partial E\setminus\partial\Omega}\oint b_k \mathcal{H}_i\cdot\vec{n}\,d\sigma}_{\text{inner fluxes}} + \underbrace{\sum_{e\in\partial E\cap\partial\Omega}\oint b_k \mathcal{H}_i^b\cdot\vec{n}\,d\sigma}_{\text{boundary fluxes}} - \int_E \nabla b_k\cdot\mathcal{F}_i d\Omega = 0,\ 0\le k\le n \tag{3.6}$$

Now, as can be seen in (3.6), the boundary integral has been split in domain internal and domain external boundaries. All boundary conditions will be imposed in a weak manner. We construct an exterior boundary state $u_h^b(u_h^-, u_{BC})$, which is a function of the interior state u_h^- and the known physical boundary data u_{BC}, see figure 3.2(b). Hence, the numerical boundary flux is computed as $\mathcal{H}_i^b = \mathcal{H}_i\left(u_h^-, u_h^b\right)$. All boundary conditions, used later in this work, are discussed in detail in section 3.4.

The time-wise evolution of the discontinuous initial solution (u_h^- and u_h^+ or u_h^- and u_h^b) between two elements is the—from the theory of the FV method—well known Riemann problem and therefore, the full spectrum of Riemann solver theory can be applied, see for example [94]. In this work we implemented the approximate Riemann solvers of Lax-Friedrich [64], Roe [84], and Harten-Lax-vanLeer [48] and in addition the exact Godunov [46] Riemann solver for producing reference solutions.

All face and element integrals are solved numerically by Gaussian integration, where the approximation order of the integration rule naturally depends on the polynomial order n of the chosen ansatz and test function space.

3.2. Formulation for the Navier-Stokes equations

If we want to use the DG approach for computing application-oriented flow fields, we necessarily have to account for

- viscous effects and thermal conductivity as well as
- some kind of turbulence or subgrid-scale model.

However, the inclusion of the above listed effects introduces second order derivatives into the governing aerodynamic model equations. In the case of laminar flow (2.4) the viscous flux vector $\mathcal{F}_v = \mathcal{F}_v(U, \nabla U)$, and in case of turbulent flow (2.17), (2.18), (2.24) the diffusive as well as the source terms $S = S(U, \nabla U)$ depend on the gradient ∇U of the solution itself. For this reason, in our DG formulation we additionally have to handle not only jumps in the solution, like in the case of the Euler equations, but we also have to handle jumps in the first order derivatives, which complicates the situation considerably. We first want to clarify this problem by presenting the main facts with reference to an easy model problem. This will be the subject of the next section, whereafter the discussion for the NS equations will be continued.

3.2.1. Model problem

The problem of handling second order derivatives with the DG method will be analysed using as an example a purely diffusive model problem. Consider the heat equation in one space dimension

$$u_t - u_{xx} = 0 \tag{3.7}$$

where periodic boundary conditions are used. The initial solution is a sinusoidal distribution with amplitude 1.0. The domain extends over one wave length ($0 \leq x \leq 2\pi$). The exact analytical solution of (3.7) is the sinusoidal distribution damped in time

$$u\left(x,t\right) = e^{-t}\sin\left(x\right), \quad 0 \leq x \leq 2\pi.$$

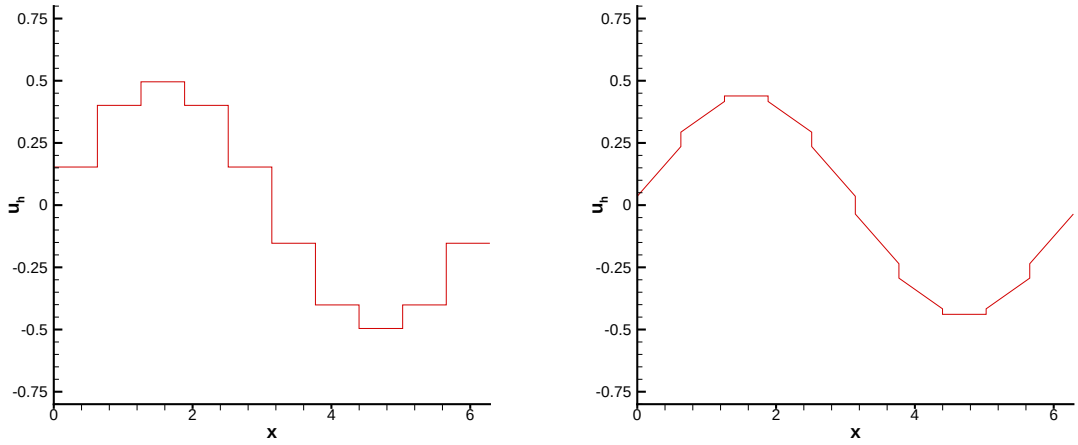
At first sight, a naive generalisation of the DG method, is to follow the same way as we have done for purely hyperbolic problems, like the Euler equations presented in section 3.1. This would finally result in the following scheme

$$\frac{d}{dt}\int_E b_k u_h d\Omega + \oint_{\partial E} b_k \frac{\partial u_h}{\partial x}\cdot\vec{n}\, d\sigma - \int_E \nabla b_k \cdot \frac{\partial u_h}{\partial x} d\Omega = 0,\ 0 \leq k \leq n \tag{3.8}$$

As one can see, the only difference between (3.5) and (3.8) is that u_h is replaced by $\frac{\partial u_h}{\partial x}$ in the second and third integrals. In order to handle discontinuous derivatives in the boundary integral, we again have to introduce the numerical flux principle for u_x now. Due to the lack of upwind mechanism, central fluxes are a natural way to define these

$$\hat{u}_x = \frac{1}{2}\left(\left(\frac{\partial u_h}{\partial x}\right)^- + \left(\frac{\partial u_h}{\partial x}\right)^+\right). \tag{3.9}$$

Now, consider for example the following interim solutions at a time level ($t_1 = 0.8$), where P^0 and P^1 line elements are used for discretisation, see figure 3.3. If we first analyse the case of constant P^0-elements, the analytical gradient $\frac{\partial u_h}{\partial x}$ by definition is zero in all elements.



(a) P^0 elements

(b) P^1 elements

Figure 3.3.: Interim solutions at time level ($t_1 = 0.8$)

Hence, the problem is obvious, because the resulting semi-discretised equation becomes

$$\frac{d}{dt}\left(u_h\right) = 0.$$

The scheme does not recognise the jumps in the solution, or more precisely formulated, the first order scheme is not consistent. Another problem arises, if P^1 elements are used. The solution gradient is not zero anymore, but it may be discontinuous across the cell boundaries, and the second order scheme also produces inconsistent solutions. Hence, the above naive scheme (3.8) with central numerical fluxes for the inter element boundaries is inconsistent in general, resulting in amplitude errors of the numerical solution of the heat equation [110].

In the case of standard FV methods, the usual procedure to determine the required gradients is to apply reconstruction. But, as described earlier (in the introduction section 1.2.2), the size of the required reconstruction stencil would again depend on the desired spatial order for the approximated (continuous) solution gradient. That is exactly the drawback we want to get rid of, when using the DG method instead of the FV method. The compactness of the scheme should strictly be maintained, if higher-order derivatives are contained in the flow model equations. However, there are investigators, who try to apply the reconstruction idea for DG anyway, giving up the locality of the

original method. To some degree—if the reconstruction stencil includes only the imediate neighbours—such schemes can be compared to the alternative compact methods which will be described in the following.

At present, only a few approaches exist to handle higher order (≥ 2)[1] derivatives in the DG framework. Historically caused, there developed two different viewpoints, affected by the associated research communities, namely the traditional finite element and the traditional DG community. The two approaches mirror the primal formulation and the flux formulation of the respective schemes, where the primal formulation is the typical viewpoint of the finite element people and the flux formulation is the preferred notation of the traditional DG followers. At first sight, the schemes appear completely different, but Arnold et al. [8] showed, that they are conceptionally similar, if the flux formulated schemes are converted into the primal formulation and vice versa. In the primal formulation the scheme is brought into the so-called bilinear form B of the equations. The properties of B are important for the analysis of the method. Arnold et al. proved several properties of the available schemes (convergency, error estimates, etc.) by transforming the schemes from the flux formulation to the primal formulation. However, in this thesis, we will concentrate on the flux formulation, which, in our opinion is easier to understand and straightforward to implement.

The flux formulation is obtained by adopting the mixed formulation from the standard FE method. The first step is to introduce the derivative (or a gradient) as new additional unknown variable Θ. The resulting coupled first-order system looks like

$$\frac{\partial u}{\partial x} - \Theta = 0 \tag{3.10}$$

$$\frac{\partial u}{\partial t} + \frac{\partial \Theta}{\partial x} = 0 \tag{3.11}$$

The next step is to discretise both equations with the DG approach, resulting in

$$\int_E \tau \Theta d\Omega - \oint_{\partial E} \tau u \, \vec{n} \, d\sigma + \int_E \nabla \tau u d\Omega = 0 \tag{3.12}$$

$$\int_E \left(v \frac{\partial u}{\partial t} \right) d\Omega - \oint_{\partial E} v \Theta \cdot \vec{n} \, d\sigma + \int_E \nabla v \cdot \Theta d\Omega = 0 \tag{3.13}$$

If we choose the same discontinuous test functions and ansatz functions for the solution gradient Θ as for the solution u itself, we arrive at the following scheme

$$\int_E b_k \Theta_h d\Omega - \underbrace{\oint_{\partial E} b_k u_h \, \vec{n} \, d\sigma}_{I} + \int_E \nabla b_k u_h d\Omega = 0, \; 0 \leq k \leq n \tag{3.14}$$

[1] Note, that due to the inherent integral formulation of a DG scheme (and integration by parts) the degree of the occurring derivatives is reduced by one.

$$\int_E \left(b_k \frac{\partial u_h}{\partial t} \right) d\Omega + \underbrace{\oint_{\partial E} b_k \Theta_h \cdot \vec{n}\, d\sigma}_{II} + \int_E \nabla b_k \cdot \Theta_h d\Omega \;=\; 0,\ 0 \le k \le n \qquad (3.15)$$

Note, that the choice to use the same finite element space ($\tau = v$) in (3.14) and (3.15) does not lead to the stability problems, which are known from the mixed continuous Galerkin method [56].

In order to handle the discontinuities occurring in the boundary integrals I and II we again, like for the Euler part, have to define numerical fluxes:

$$\begin{aligned} \oint_{\partial E} b_k\, u_h\, \vec{n}\, \mathrm{d}\sigma &\approx \oint_{\partial E} b_k\, \hat{u}(u_h^+, u_h^-) \vec{n}\, \mathrm{d}\sigma \\ \oint_{\partial E} b_k \Theta_h \cdot \vec{n}\, \mathrm{d}\sigma &\approx \oint_{\partial E} b_k \hat{\Theta}(u_h^+, \Theta_h^+, u_h^-, \Theta_h^-) \cdot \vec{n}\, \mathrm{d}\sigma \end{aligned}$$

The final mixed DG scheme, in flux formulation form, is

$$\int_E b_k \Theta_h d\Omega - \sum_{e \in E} \oint b_k \hat{u}(u_h^+, u_h^-)\, \vec{n}\, d\sigma + \int_E \nabla b_k u_h d\Omega \;=\; 0 \qquad (3.16)$$

$$\int_E \left(b_k \frac{\partial u_h}{\partial t} \right) d\Omega + \sum_{e \in E} \oint b_k \hat{\Theta}(u_h^+, \Theta_h^+, u_h^-, \Theta_h^-) \cdot \vec{n}\, d\sigma + \int_E \nabla b_k \cdot \Theta_h d\Omega \;=\; 0 \qquad (3.17)$$

Most proposed fluxes—all which have been published so far—are summarised in Table 3.2.1. Table 3.2.1 is adopted from [8], where Arnold et al. consider the purely elliptic model problem

$$-\frac{\partial^2 u}{\partial x^2} = f \qquad \text{in } \Omega \qquad u = 0 \text{ on } \partial\Omega, \qquad (3.18)$$

Here, f should stand for a known function. We only want to briefly summarise the main properties of the schemes here, in order to explain the special choice (schemes no. 3 and 5) we made in this work. The interested reader is refered to [8, 7], where the schemes are analysed in a rigorous mathematical framework. In order to understand the formulations in Table 3.2.1, we have to introduce the jump [ ] and average { } operators with the following meanings. For a scalar quantity s, the operators are given by:

$$\begin{aligned} [s] &= s^+ \vec{n}^+ + s^- \vec{n}^- = \vec{n} \left(s^+ - s^- \right) \\ \{s\} &= \frac{1}{2} \left(s^+ + s^- \right). \end{aligned}$$

And for a vector quantity $\vec{\varphi}$, are given by

$$[\vec{\varphi}] \;=\; \vec{\varphi}^+ \cdot \vec{n}^+ + \vec{\varphi}^- \cdot \vec{n}^- = \vec{n} \cdot \left(\vec{\varphi}^+ - \vec{\varphi}^- \right)$$

$$\{\vec{\varphi}\} \;=\; \frac{1}{2}\left(\vec{\varphi}^{+} + \vec{\varphi}^{-}\right).$$

No.	Method	$\hat{u}$	$\hat{\Theta}$
1	Bassi-Rebay (BR1)	$\{u_h\}$	$\{\Theta_h\}$
2	Brezzi et al. 1	$\{u_h\}$	$\{\Theta_h\} - \alpha_r([u_h])$
3	**LDG**	$\{u_h\} - \beta \cdot [u_h]$	$\{\Theta_h\} + \beta[\Theta_h] - \alpha_j([u_h])$
4	IP	$\{u_h\}$	$\{\nabla_h u_h\} - \alpha_j([u_h])$
5	**Bassi et al. (BR2)**	$\{u_h\}$	$\{\nabla_h u_h\} - \alpha_r([u_h])$
6	Baumann-Oden	$\{u_h\} + n_K \cdot [u_h]$	$\{\nabla_h u_h\}$
7	NIPG	$\{u_h\} + n_K \cdot [u_h]$	$\{\nabla_h u_h\} - \alpha_j([u_h])$
8	Babuska-Zlamal	$(u_h\|_K)\partial K$	$-\alpha_j([u_h])$
9	Brezzi et al. 2	$(u_h\|_K)\partial K$	$-\alpha_r([u_h])$

Table 3.1.: Several DG methods and their interior numerical fluxes, taken from the studies of Arnold et al. [8, 7]

Furthermore, the functional operators α_r and α_j are penalty terms defined like

$$\alpha_r(\phi) = -\eta_e \{r_e(\phi)\} \;\text{ with }\; \int_\Omega r_e(\phi) \cdot \tau \, d\Omega = \int_e \phi \cdot \{\tau\} \, d\sigma$$

$$\alpha_j(\phi) = \mu[\phi] \text{ with } \mu = \eta_e h_e^{-1} \tag{3.19}$$

Here $r_e(\phi)$ is the so-called lifting operator and, η_e is a positive number on an edge e and h_e is the element size. The inclusion of penalty terms into the flux $\hat{\Theta}$ is crucial for obtaining a stable scheme. The methods 1 and 6 do not contain any penalty terms and therefore are only weakly stable. All the methods containing penalty terms α_r or α_j can also be interpreted as interior penalty methods. The penalty method was originally introduced by Nitsche [73] in order to prescribe Dirichlet boundary conditions weakly, without incorporating the boundary conditions into the finite element space. This is done by cleverly adding a penalty term to the original formulation, which (automatically) forces the solution to satisfy the boundary conditions. This concept was adopted by Arnold [6] for the penalisation of interior discontinuities, where the name interior penalty (IP) method originates/stems from. The more physical interpretation of the penalty method is, that due to the additional penalty term, the diffusivity at jumps is increased artificially and consequently the jumps are smoothed out, or, in other words, the jump is penalised. A more detailed theoretical overview on IP type schemes is given by Arnold et al. [8].

It was shown by theoretical analysis [8], that two main requirements for the numerical fluxes have to be met, in order to obtain a stable scheme, which also achieves optimal order of convergence $\mathcal{O}\left(h^{p+1}\right)$. That is, the fluxes $\hat{u}$ and $\hat{\Theta}$ should be consistent as well as conservative. A numerical flux is said to be conservative, if the numerical flux is single-valued on the inter element boundaries. A numerical flux is consistent, if for the

special case of smooth continuous inter element boundary values, the numerical flux is identical to the analytic flux function at these boundaries.

Concerning the schemes 1 to 3 the numerical flux in the auxiliary equation $\hat{\Theta}$ depends on Θ_h itself, which produces less sparse stiffness matrices, containing more non-zero entries. In other words, the discretisation stencil is less compact than that of the schemes 4 to 9.

The schemes 3 to 7 achieve optimal order of convergence $\mathcal{O}\left(h^{p+1}\right)$ in L_2. The first Bassi-Rebay (BR1) method achieves a convergence rate of $\mathcal{O}\left(h^{p}\right)$ for p odd and $\mathcal{O}\left(h^{p+1}\right)$, if p is even. In contrast, the Baumann-Oden method and the NIPG method methods achieve optimal order of convergence $\mathcal{O}\left(h^{p+1}\right)$ for odd p and suboptimal convergence $\mathcal{O}\left(h^{p}\right)$ for p even. In fact, the Bauman-Oden method is inconsistent for P^0elements.

The IP and NIPG method need a problem dependent stabilisation parameter (μ in equation (3.19)), whereas the second Bassi-Rebay (BR2) method seems to be the more general approach, since the computation of the penalty term α_r is more straightforward than the choice of the penalty function μ in the α_j penalisation term. For example, the stabilisation factor of the IP method is dependent on the mesh regularity and polynomial order.

As reported above the stencil of the LDG scheme is non-compact. In fact, the second neighbours are additionally included in the stencil. However, in practice the auxiliary equation (3.16) is solved before the main equation in every timestep and the auxiliary variable Θ is stored as extra degree of freedom. In the case of an explicit time stepping scheme, this two-step algorithm is compact in every step. Therefore, the only disadvantage is the extra memory required for storing the Θ. LDG achieves uniform convergency rates - BR1 rates are dependent whether polynomial order is even or odd.

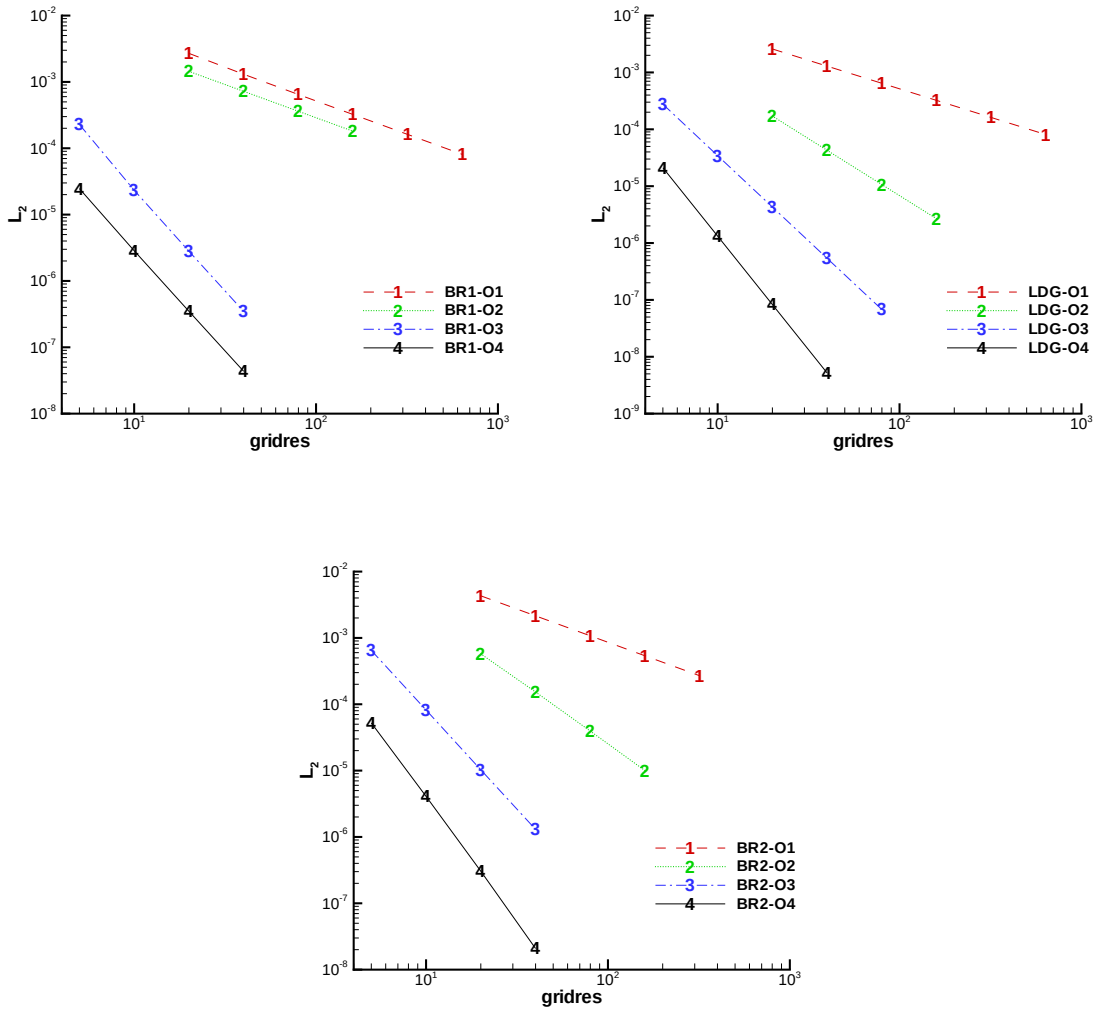


Figure 3.4.: Comparison of rate of convergence for BR1, LDG and BR2 scheme.

Under the prevailing circumstances, we finally have chosen the bold face and in red highlighted schemes no. 3 and 5, the LDG and the BR2 approach for implementing into our DG C++ code. A first validation of our implementation is done by implementing the BR1, LDG and BR2 schemes in one dimension. The convergence behaviour, obtained by analysing the heat equation problem (3.7), is shown in figure 3.4, see appendix D.1 for tabulated results. We can verify the theoretical facts of [8, 7] , summarised above. The LDG as well as the BR2 scheme achive optimal order of convergence for all ansatz orders, whereas the BR1 method only achieves suboptimal orders of convergence. We can state here, that the implementation of further schemes is easy anyway, because of the distinct similarities of the DG schemes and the flexible (object-oriented) code design.

3.2.2. Mixed discontinuous Galerkin formulation for the NS equations

Following the formulation of the second derivative dicretisation, we now introduce the flux formulation for the system of NS equations. The first step again is to introduce the gradient as new additional unknown variable Θ.

The resulting coupled first-order system is

$$\nabla U - \Theta = 0 \tag{3.20}$$

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathcal{F}_i(U) = \nabla \cdot \mathcal{F}_v(U, \Theta) \tag{3.21}$$

In fact, the viscous flux tensor $\mathcal{F}_v$ is a function of the conservative state variable vector U and the gradient of the primitive variables, see equation 2.5. Hence we work with Θ, wich is given by

$$\Theta = \nabla \begin{pmatrix} u \\ v \\ w \\ T \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \\ T_x & T_y & T_z \end{pmatrix}$$

Another approach is, to choose the gradient of the conservative state vector as new unknown Θ, and reformulate the viscous flux function $\mathcal{F}_v = \mathcal{F}_v(U, \nabla U)$ as a function of the conservative state and gradient of the conservative variables, resulting in an equation like $\Theta = \mathcal{A}_v \nabla U$, where $\mathcal{A}_v = \frac{\partial \mathcal{F}_v(U, \nabla U)}{\partial \nabla U}$ is the viscous Jacobian matrix [14, 59].

The next step is to discretise both equations with the DG approach, resulting in

$$\int_E \tau \Theta d\Omega - \oint_{\partial E} \tau\, U\, \vec{n}\, d\sigma - \int_E \nabla \tau U d\Omega = 0 \tag{3.22}$$

$$\underbrace{\int_E \left(v \frac{\partial U}{\partial t} \right) d\Omega + \oint_{\partial E} v \mathcal{F}_i \cdot \vec{n}\, d\sigma - \int_E \nabla v \cdot \mathcal{F}_i d\Omega}_{\text{Euler}}$$

$$- \oint_{\partial E} v \mathcal{F}_v \cdot \vec{n}\, d\sigma + \int_E \nabla v \cdot \mathcal{F}_v d\Omega = 0 \tag{3.23}$$

where the left hand side of the second equation is the Euler part, which has been discussed in detail in a previous section. If we choose the same discontinuous test functions and ansatz functions for the solution gradient Θ as for the solution U itself, we arrive at the following scheme

$$\int_E b_k \Theta_h d\Omega - \underbrace{\oint_{\partial E} b_k u_h\, \vec{n}\, d\sigma}_{I} + \int_E \nabla b_k u_h d\Omega = 0,\ 0 \leq k \leq n \tag{3.24}$$

$$\text{Euler} - \underbrace{\oint_{\partial E} b_k \mathcal{F}_v \cdot \vec{n}\, d\sigma}_{II} + \int_E \nabla b_k \cdot \mathcal{F}_v d\Omega \;=\; 0,\; 0 \le k \le n \tag{3.25}$$

Note, that the choice to use the same finite element space ($\tau = v$) in (3.24) and (3.25) does not cause stability problems encountered in the continuous Galerkin method.

In order to handle the discontinuities occurring in the boundary integrals I and II we again, similar to the Euler part, have to define the following numerical fluxes:

$$\begin{aligned}
\oint_{\partial E} b_k\, u_h\, \vec{n}\, \mathrm{d}\sigma &\approx \oint_{\partial E} b_k\, \mathcal{H}_{aux}(u_h^+, u_h^-)\vec{n}\, \mathrm{d}\sigma \\
\oint_{\partial E} b_k \mathcal{F}_v \cdot \vec{n}\, \mathrm{d}\sigma &\approx \oint_{\partial E} b_k \mathcal{H}_v(u_h^+, \Theta_h^+, u_h^-, \Theta_h^-) \cdot \vec{n}\, \mathrm{d}\sigma
\end{aligned}$$

If we split the element boundary integrals into inner face integrals and domain boundary face integrals, we finally arrive at the mixed DG formulation of the Navier-Stokes equations

$$\begin{aligned}
\int_E b_k \Theta_h dE - \underbrace{\sum_{e \in E \setminus \partial\Omega} \oint b_k \mathcal{H}_{aux}\, \vec{n}\, d\sigma}_{\text{inner fluxes}} - \underbrace{\sum_{e \cap \partial\Omega} \oint b_k \mathcal{H}^b_{aux}\, \vec{n}\, d\sigma}_{\text{boundary fluxes}} & \\
+ \int_E \nabla b_k u_h d\Omega \;=\; 0,\; 0 \le k \le n & \quad (3.26) \\
\text{Euler} - \underbrace{\sum_{e \in E \setminus \partial\Omega} \oint b_k \mathcal{H}_v \cdot \vec{n}\, d\sigma}_{\text{inner fluxes}} + \underbrace{\sum_{e \cap \partial\Omega} \oint b_k \mathcal{H}^b_v \cdot \vec{n}\, d\sigma}_{\text{boundary fluxes}} & \\
+ \int_E \nabla b_k \cdot \mathcal{F}_v d\Omega \;=\; 0,\; 0 \le k \le n & \quad (3.27)
\end{aligned}$$

So the resulting scheme is a two-step method. First, at the beginning of every timestep, the simple auxiliary equation is solved, where the gradients Θ_h are updated with the newest flow quantities u_h. In the second step, the main equation for the flow field is solved, using the values of Θ_h obtained from step one.

The boundary fluxes $\mathcal{H}^b_{aux}$ and $\mathcal{H}^b_v$ are treated weakly by constructing the exterior states from the interior states and known boundary data. Details are discussed in Section 3.4. The choice of the fluxes $\mathcal{H}_{aux}$ and $\mathcal{H}_v$ is the crucial part of the weak formulation for DG methods for higher order (> 1) derivatives, since there is no counterpart or experience from the FV method, one can adopt here, like we have done for the Euler part before. Therefore the choices of numerical fluxes $\mathcal{H}_{aux}$ and $\mathcal{H}_v$ for different approaches, has been adopted from theoretical and numerical studies of purely diffusive model problems, discussed in 3.2.1.

3.2.3. BR1 and Local DG method

In our code, we utilised the first and third schemes of Table3.2.1. For BR1, a central discretisation is proposed for the auxiliary as well as for the viscous fluxes:

$$\begin{aligned} \mathcal{H}_{aux} &= \frac{1}{2}\left(u_h^+ + u_h^-\right) \\ \mathcal{H}_v &= \frac{1}{2}\left(\mathcal{F}_v^+\left(u_h^+, \Theta_h^+\right) + \mathcal{F}_v^-\left(u_h^-, \Theta_h^-\right)\right). \end{aligned}$$

In the literature this scheme is refered to as the first Bassi-Rebay scheme and we will use the name BR1 method. Cockburn and Shu demonstrated several deficiencies of the BR1 scheme [33]:

- convergence order of only k for odd ansatz
- spread stencil

Both problems can be remedied by choosing one-sided fluxes in opposite directions in the following form. This choice of the fluxes belongs to the class of LDG methods, where the stabilisation term α_j vanishes with $\mu = 0$:

$$\mathcal{H}_{aux} = u_h^+ \text{ and } \mathcal{H}_v = \mathcal{F}_v^-\left(u_h^-, \Theta_h^-\right) \text{ with } \beta = -\tfrac{1}{2},\ \mu = 0$$

or alternatively

$$\mathcal{H}_{aux} = u_h^- \text{ and } \mathcal{H}_v = \mathcal{F}_v^+\left(u_h^+, \Theta_h^+\right) \text{ with } \beta = \tfrac{1}{2},\ \mu = 0.$$

3.2.4. Second Bassi-Rebay method (BR2)

The high memory requirements of the LDG method, especially for an implicit scheme in time, associated with a huge overall system matrix is the main practical handicap of the LDG method. For example, in three dimensions the number of unknowns increases from 5 to 17 for laminar flow and from 7 to 25 for the $k-\omega$ turbulence model, and the number of system matrix entries increases approximately by the square of this factor. In order to get rid of the memory demanding extra degrees of freedom Θ, Bassi and Rebay introduced a new scheme, also known as the second Bassi-Rebay scheme (BR2) [16]. They introduced/derived a formulation for the auxiliary variable Θ, where Θ can be expressed as a sum of the solution gradient ∇U and a "correction" R. The correction gradient R takes into account the effect of interface discontinuities. The equation for Θ reads

$$\int_E v\Theta d\Omega = \underbrace{\int_E v\nabla u_h d\Omega}_{I:\text{"real" gradient}} + \underbrace{\oint_{\partial E} v\,\frac{1}{2}(u_h^+ - u_h^-)\vec{n}\,d\sigma}_{II:\text{"correction" } R \text{ of gradient}} \tag{3.28}$$

(3.28) is a formulation of the form $\Theta = \Theta(u, \nabla u)$. Consequently, in a weak sense, or at a the discrete level one can write

$$\Theta = \nabla u_h + R.$$

The way to solve part (II) in (3.28) is rarely discussed in the literature so far. One possibility is to express R as polynomial expansion.

$$R(x,t)_h = \sum_{k=1}^{n} R_k(t) b_k(x),$$

In other words, we again apply the same DG method as we do for the main flow equations. Then the definition of R, also mathematically referred to as the (global) lifting operator, follows from the element boundary integral (II) as

$$\int_E b_k R_h d\Omega = \oint_{\partial E} b_k \frac{1}{2}(u_h^+ - u_h^-)\vec{n}\, d\sigma. \tag{3.29}$$

If we split the face integral into inner and outer boundaries R is computed as

$$\int_E b_k R_h d\Omega = \int_{\partial E \setminus \partial\Omega} b_k \frac{1}{2}(u_h^+ - u_h^-)\vec{n}\, d\sigma + \int_{\partial E \cap \partial\Omega} b_k\, (u_h^b - u_h^-)\vec{n}\, d\sigma. \tag{3.30}$$

where the exterior state u_h^b is constructed from the interior state and the known boundary data.

It has been reported in the literature, that using the (global) lifting operator, can lead to unsatisfactory convergence rates for polynomial approximations of odd order, see [33, 16]. This problem can be cured by using the so called local lifting operator r_h for the correction of the boundary solution gradients $(\nabla U)^-$and $(\nabla U)^+$, respectively. The local lifting operator for inner faces is defined by

$$\int_E b_k r_h\, d\Omega = \oint_{\partial E \setminus \partial\Omega} b_k \frac{1}{2}(u_h^+ - u_h^-)\vec{n}\, d\sigma. \tag{3.31}$$

On boundary faces the local lifting operator is defined by

$$\int_E b_k r_h\, d\Omega = \oint_{\partial E \cap \partial\Omega} b_k\, (u_h^b - u_h^-)\vec{n}\, d\sigma \tag{3.32}$$

that is, an integration on a single face only, and the relation between local and global lifting operator reads

$$R_h = \sum_{e \in \partial E} r_h.$$

Now, with the help of equations (3.29) and (3.32), we can express Θ in the mixed DG formulation with the sum of the solution gradient ∇u_h and a "correction" R_h or r_h, respectively. Doing the following replacements

$$\begin{aligned} \Theta_h^+ &= \nabla u_h^+ + r_h^+ \\ \Theta_h^- &= \nabla u_h^- + r_h^- \\ \Theta_h &= \nabla u_h + R_h \end{aligned}$$

and using a central numerical viscous flux again delivers the BR2 scheme

$$\begin{aligned} \text{Euler} - \sum_{e \in \partial E} b_k \frac{1}{2} \left(\mathcal{F}_v^+(u_h^+, \nabla u_h^+ + r_h^+) + \mathcal{F}_v^-(u_h^-, \nabla u_h^- + r_h^-) \right) \cdot \vec{n} \, d\sigma \\ + \int_E \nabla b_k \cdot \mathcal{F}_v(u_h, \nabla u_h + R_h) \, dE &= 0 \end{aligned} \quad (3.33)$$

Due to the replacements, the method can be reduced to a one-step method, like in the Euler case, since the corrections R_h and $r_h$are only needed as intermediate values, which means, that they have not to be stored seperately, as the auxiliary variable Θ_h in the BR1 and LDG methods.

The scheme is more compact than the LDG method, because now the residual of an inner cell just depends on the direct neighbor cells. As aforementioned, the BR2 approach can also be interpreted as reconstruction schemes, but with the crucial advantage, that the stencil is compact and the resulting scheme is really high-order, independent of the grid distortion or quality.

In our implementation, we work with the parameter $\eta_e = 1$, even if Brezzi et al. proved stability, if η_e is chosen equal to the number of faces. This proof is not strict and we performed all our simulations with $\eta_e = 1$ and did not encounter any stability problems.

3.3. Turbulence modeling

3.3.1. Discretisation of the RANS equations

The main flow equations and the turbulence model equations are solved in a completely coupled manner. This means, that the conservative state vectors and the flux tensors are extended by the quantities of the respective turbulence model. This results in the following state vectors for the $k - \omega$ model or SA model, respectively.

$$U_{k-\omega} = \left(\begin{array}{c} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \\ \hline \\ \rho k \\ \rho\tilde{\omega} \end{array}\right), \qquad U_{SA} = \left(\begin{array}{c} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \\ \hline \\ \rho\tilde{\nu} \end{array}\right)$$

The same extension holds for the auxiliary variable for the gradient Θ

$$\Theta_{k-\omega} = \nabla \left(\begin{array}{c} u \\ v \\ w \\ T \\ \hline \\ k \\ \tilde{\omega} \end{array}\right) = \left(\begin{array}{ccc} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \\ T_x & T_y & T_z \\ \hline \\ k_x & k_y & k_z \\ \tilde{\omega}_x & \tilde{\omega}_y & \tilde{\omega}_z \end{array}\right), \quad \Theta_{SA} = \left(\begin{array}{ccc} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \\ T_x & T_y & T_z \\ \hline \\ \tilde{\nu}_x & \tilde{\nu}_y & \tilde{\nu}_z \end{array}\right)$$

and the flux vectors, here only written for the $x-$components for brevity, are

$$F^x_{i,k-\omega} = \left(\begin{array}{c} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho e_t + p)\, u \\ \hline \\ \rho uk \\ \rho u\tilde{\omega} \end{array}\right), \quad F^x_{v,k-\omega} = \left(\begin{array}{c} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x \\ \hline \\ (\mu + \sigma_{k1}\mu_t)\, k_x \\ (\mu + \sigma_{\omega 1}\mu_t)\, \tilde{\omega}_x \end{array}\right)$$

$$F^x_{i,SA} = \left(\begin{array}{c} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho e_t + p)\, u \\ \hline \\ \rho u\tilde{\nu} \end{array}\right), \quad F^x_{v,SA} = \left(\begin{array}{c} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x \\ \hline \\ \frac{1}{\sigma}\left(\mu + \rho\tilde{\nu}\right)\tilde{\nu}_x \end{array}\right)$$

Due to the production and destruction terms in the turbulence model equations, we have to extend the LDG and BR2 schemes presented in section 3.1 and 3.2 by source

terms S. This is straightforward, since we only have to add the test function weighted source term at the respective right hand sides.

The LDG scheme for the RANS equations results in

$$\begin{aligned}\int_E b_k \Theta_h d\Omega - \oint_{\partial E} b_k \mathcal{H}_{aux}\,\vec{n}\,d\sigma - \oint_{\partial E} b_k \mathcal{H}^b_{aux}\,\vec{n}\,d\sigma \\ + \int_E \nabla b_k \nabla u_h d\Omega d\Omega &= 0,\ 0 \le k \le n \qquad (3.34)\\ \text{Euler} - \oint_{\partial E} b_k \mathcal{H}_v \cdot \vec{n}\,d\sigma + \oint_{\partial E \cap \partial\Omega} b_k \mathcal{H}^b_v \cdot \vec{n}\,d\sigma \\ + \int_E \nabla b_k \cdot \mathcal{F}_v d\Omega - \int_E b_k S(u_h, \Theta_h) d\Omega &= ,\ 0 \le k \le n \qquad (3.35)\end{aligned}$$

The BR2 scheme for the RANS equations results in

$$\begin{aligned}\text{Euler} - \sum_{e \in \partial E} b_k \frac{1}{2}\left(\mathcal{F}_v^+(U^+, \nabla U^+ + r_h^+) + \mathcal{F}_v^-(U^-, \nabla U^- + r_h^-)\right) \cdot \vec{n}\,\mathrm{d}\sigma \\ + \int_E \nabla v_k \cdot \mathcal{F}_v(u_h, \nabla u_h + R_h)\,\mathrm{d}E - \int_E b_k S(u_h, \nabla u_h + R_h) d\Omega &= 0\end{aligned} \qquad (3.36)$$

3.3.2. Limiting

It is well known from the FV method, as well as from the FD method, that discretising turbulence model (transport) equations in a robust manner is a difficult task. The crux of the matter are the stiff source terms, which complicate the situation a lot, especially during startup of the simulation. In principle, the problem can be solved by drastically reducing the discretisation timestep. Another possibility is, to apply an implicit time integration, whereby the severe time step barrier can be partially overcome. However, all these modifications/tricks cannot prevent that, for a few cells, the turbulence variables k and $\tilde{\nu}$ could become negative and thus unphysical.

This can be demonstrated with an easy example. In figure 3.5 the L_2-projection of the cubic distribution $\mathrm{f}(\xi, \eta) = 100\xi^3 + \xi^2$ is shown for (triangular) P^0, P^1 and P^2 elements. The function $\mathrm{f}(\xi, \eta)$ can be exactly mapped into the FE space with P^3 elements, represented by the solid line. As expected, the approximation quality of f improves with increased polynomial ansatz order. However, the important point here is, that the linear as well as the quadratic projections partially fall below zero, whereas the constant approximation remains positive in the complete interval. Hence, P^0 elements—nothing else than the cell-centered FV approach—are better suited than higher order elements concerning positivity. This is a very important fact and will be further examined in the results chapter.

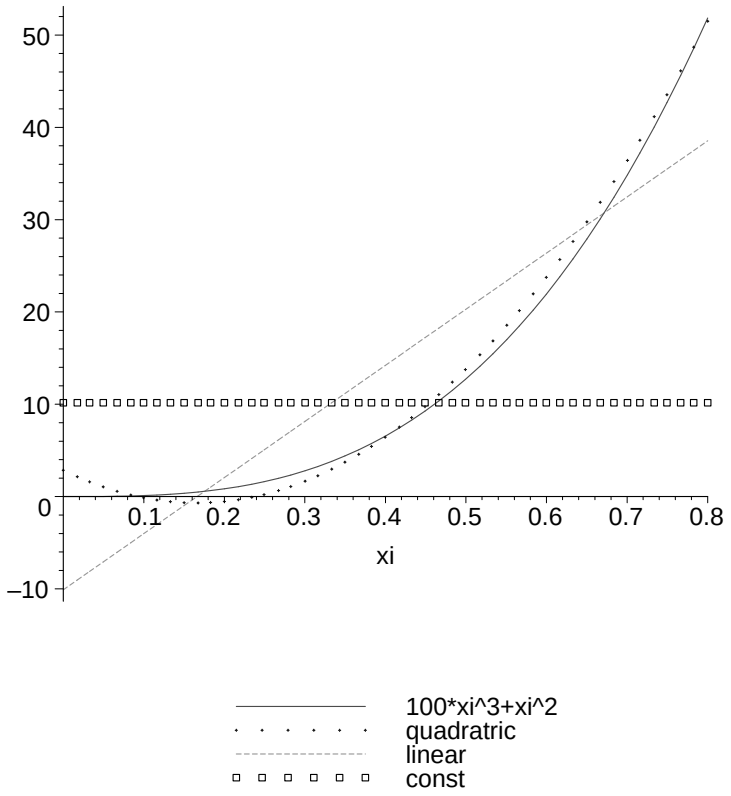


Figure 3.5.: L2 projection of the cubic function $\mathrm{f}(\xi, \eta) = 100\xi^3 + \xi^2$ into FE space.

Negative values of k and $\tilde{\nu}$ can lead to stability problems of the turbulence model, especially for the SA model. If small enough time steps are used, the problem can be cured. But this is conflictive to the idea of the implicit approach, which should allow high CFL numbers. In order to circumvent these problems, it is an ordinary practice to limit the variables to small positive values or zero, respectively.

Here we want to distinguish between two approaches. The first one, introduced by Bassi and Rebay [15] in the first DG implementation with the $\mathrm{k} - \omega$ model, should be depicted as soft limiting in the following. In [15] the influence of negative values on further computations, for example the eddy viscosity, is avoided by limiting

$$\mu_t = \max\left(Re_\infty \frac{\rho k}{\omega},\, 0\right).$$

However, negative values of k are generally allowed in the solution vector u_h.

The second approach, called hard limiting, is, that each time step negative values are limited in the solution vector u_h. The soft limitation fails for the SA model, because the model itself seems to react unstably to negative $\tilde{\nu}$ values. Consequently, for the SA model hard limiting seems to be the more robust method.

The hard limitation process for the case of DG approach is more involved than that for the FV method. For P^0-elements, the situation is identical to the FV limiting

approach, where the integral cell average is the only degree of freedom, which has to be respected. If higher degrees of polynomial approximation are used, the limiting becomes more complex.

For the linear (triangular grids) case the solution should not fall below zero at the element corners:

$$\begin{aligned} u_0 - u_1 - u_2 &\geq 0 \\ u_0 + u_1 - u_2 &\geq 0 \\ u_0 + 2u_2 &\geq 0 \end{aligned} \tag{3.37}$$

Condition (3.37) leads to the following positivity conditions

$$u_0 \geq 0$$

$$u_2 \leq u_0 \quad \text{and} \quad u_2 \geq -\frac{u_0}{2} \tag{3.38}$$

$$u_1 \leq u_0 - u_2 \quad \text{and} \quad u_1 \geq u_2 - u_0 \tag{3.39}$$

If u_0, which represents the integral cell average, becomes negative, all degrees of freedom are set to zero ($u_0 = u_1 = u_2$). If u_0 is positive, we limit u_2 and u_1 in turn according to (3.38) and (3.39).

For the linear (quadrilateral grids) case the solution should not fall below zero at the element corners:

$$\begin{aligned} u_0 - u_1 + u_2 &\geq 0 \\ u_0 - u_1 - u_2 &\geq 0 \\ u_0 + u_1 - u_2 &\geq 0 \\ u_0 + u_1 + u_2 &\geq 0 \end{aligned} \tag{3.40}$$

Here we determine the minimum value of the permutations $\pm u_1 \pm u_2$ and then reduce the gradients u_1 and u_2 with the same scaling

$$\begin{aligned} u_1 &= s\, u_1 \\ u_2 &= s\, u_2 \end{aligned}$$

where s can be computed as

$$s = \frac{-u_0}{\min\left(-u_1 + u_2, -u_1 - u_2, u_1 - u_2, u_1 + u_2\right)}, \quad 0 \leq \mathrm{s} \leq 1.$$

This limitation has the property, that the main trend of the solution is maintained, because the inclination of the plane represented by the degrees of freedom u_1 and u_2 is

scaled isotropically, see for example Figure 3.6.

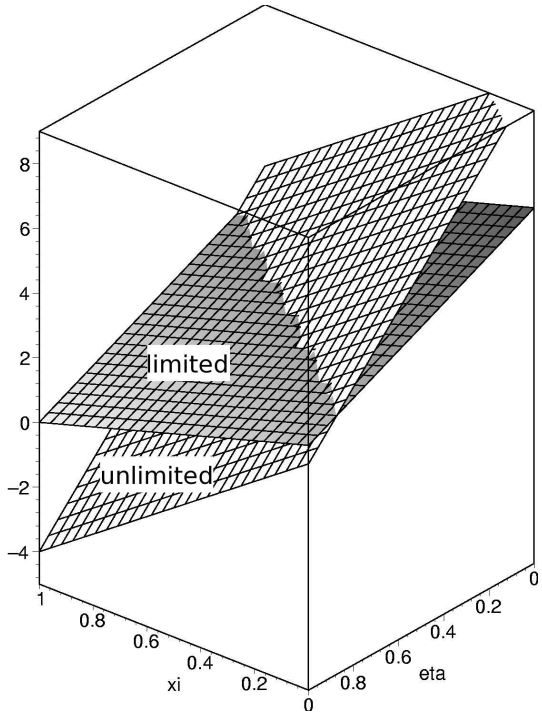


Figure 3.6.: Limitation of the linear quadrilateral element with the cell distribution $u_h(\xi, \eta) = 10 - 6\xi - 8\eta$.

For higher order elements ($P^2$and P^3) the problem is more complex. In order to check positivity, we have to determine the global minimum in every cell. The global minimum is determined by searching for the minimum of the edge, corner and possibly local inner cell extrema. Inner extrema for P^3 elements are expensive to find, because two coupled nonlinear equations have to be solved in order to fulfill the necessary extremum condition. For higher polynomial approximations (P^4, P^5, ...) limiting combined with the process of global minimum search in every cell may become a dominant part of the overall scheme and therefore questionable. If the global minimum, in a triangle or quadrilateral undershoot zero, all higher order degrees of freedom are zeroed out and linear limitation for the remaining approximation, as described above, is performed.

As long as the integral cell average u_0 is not modified the limitation strategy is conservative, in the sense, that no turbulence is produced or destroyed through limitation. Only for the cases, where even the average cell value becomes negative, conservativity has to be given up. But these cases are rare, as will be shown in the results chapter.

3.3.3. Wall distance

As can be seen from the equations (2.25) and (2.28), the computation of the production and destruction terms of the SA model depends on the distance to the wall. In practice,

the closest distance of a Gaussian cell point to a wall is needed for the integration.

The method we used for determination of these distances is straightforward and restricted to two-dimensional grids.

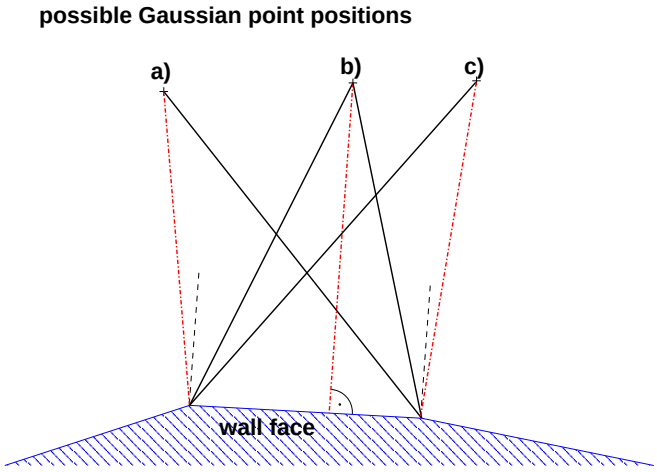


Figure 3.7.: Possible cases for minimum wall distance of a Gaussian cell point

An outer loop over all wall boundary faces is performed, wherein an inner loop over all Gaussian cell points is performed. In the inner loop the closest distance to the current (outer loop) wall face is determined. There are three possible cases, for the shortest connection vector between wall face and Gaussian point, see figure 3.7. The right choice can easily be made by evaluating the scalar product of the wall face vector $\vec{t}$ and the left and right connection vectors $\vec{l}$ and $\vec{r}$, respectively. The closest connection d is

$$d = \begin{cases} \left|\vec{l}\right| & \vec{l}\cdot\vec{t} < 0 & \text{case a)} \\ \left|\vec{l}\cdot\vec{n}_1\right| & \vec{l}\cdot\vec{t} \geq 0 \text{ and } \vec{r}\cdot\vec{t} \leq 0 & \text{case b)} \\ \left|\vec{r}\right| & \vec{r}\cdot\vec{t} > 0 & \text{case c)} \end{cases}$$

where $\vec{n}_1$ is the unit wall face normal vector.

3.4. Boundary conditions

All boundary conditions will be imposed in a weak manner. We construct an exterior boundary state $u_h^b(u_h^-, u_{BC})$, which is a function of the interior state u_h^- and the known physical boundary data u_{BC}, see figure 3.2(b).

3.4.1. Farfield

If the boundary is far enough away from the body, at these kind of boundaries freestream flow conditions are prescribed

$$u_h^b = \begin{pmatrix} \rho_\infty \\ u_\infty \\ v_\infty \\ w_\infty \\ (\rho E)_\infty \end{pmatrix}, \quad u_{h,k-\omega}^b = \left(\begin{array}{c} \rho_\infty \\ u_\infty \\ v_\infty \\ w_\infty \\ (\rho E)_\infty \\ \hline \\ \rho k_\infty \\ \rho \ln \omega_\infty \end{array}\right), \quad u_{h,SA}^b = \left(\begin{array}{c} \rho_\infty \\ u_\infty \\ v_\infty \\ w_\infty \\ (\rho E)_\infty \\ \hline \\ \rho \tilde{\nu}_\infty \end{array}\right)$$

where the freestream values of k_∞, $\tilde{\omega}_\infty$ and $\tilde{\nu}_\infty$ are described in section 2.3.2 and 2.3.3, respectively.

The numerical inviscid and viscous as well as the auxiliary farfield boundary fluxes are computed as

$$\begin{aligned} \mathcal{H}_i^b &= \mathcal{H}_i\left(u_h^-, u_h^b\right) \\ \mathcal{H}_{aux}^b &= \mathcal{H}_{aux}\left(u_h^-, u_h^b\right) \\ \mathcal{H}_v^b &= \mathcal{H}_v\left(u_h^-, \Theta_h^-, u_h^b, \Theta_h^-\right) \\ \mathcal{H}_v^b &= \mathcal{H}_v\left(u_h^-, \nabla u_h^- + r_h^-, u_h^b, \nabla u_h^- + r_h^-\right). \end{aligned}$$

3.4.2. No-slip wall

We prescribe zero velocity at the wall

$$u^b = v^b = w^b = 0$$

Furthermore, we assume adiabatic walls, which is nothing else than zero heatflux at the wall, leading to the condition of zero normal temperature gradient at the wall

$$\left(\frac{\partial T}{\partial \vec{n}}\right)^b = (\nabla T)^b \cdot \frac{\vec{n}}{|\vec{n}|} = 0 \tag{3.41}$$

The rest of the required variables is taken from the interior data

$$u_h^b = \begin{pmatrix} \rho^- \\ 0 \\ 0 \\ 0 \\ (\rho E)^- \end{pmatrix}, \quad u_{h,k-\omega}^b = \left(\begin{array}{c} \rho^- \\ 0 \\ 0 \\ 0 \\ \hline (\rho E)^- \\ \\ 0 \\ \rho \ln \omega_{\text{wall}} \end{array}\right), \quad u_{h,SA}^b = \left(\begin{array}{c} \rho^- \\ 0 \\ 0 \\ 0 \\ \hline (\rho E)^- \\ \\ 0 \end{array}\right)$$

where ω_{wall} is set according to Menter, see section 2.3.2. For the auxiliary gradient Θ_h^b and the corrected gradient $(\nabla u_h + r_h)^b$ at the wall boundary we proceed in the same manner, namely, the velocity as well as the turbulence model variable gradients are taken from the interior data and the temperature gradient is set according to (3.41). The numerical inviscid and viscous as well as the auxiliary wall boundary fluxes are computed as

$$\begin{aligned} \mathcal{H}_i^b &= \mathcal{F}_i\left(u_h^b\right) \\ \mathcal{H}_{aux}^b &= u_h^b \\ \mathcal{H}_v^b &= \mathcal{F}_v\left(u_h^b, \Theta_h^b\right) \\ \mathcal{H}_v^b &= \mathcal{F}_v\left(u_h^b, (\nabla u_h + r_h)^b\right). \end{aligned}$$

3.4.3. Slip wall

At slip walls, also refered to as inviscid or Euler walls, flow tangency is enforced

$$\vec{n} \cdot \vec{v} = 0.$$

The resulting boundary flux is the normal pressure contribution to the momentum flux

$$\mathcal{H}_i^b = \begin{pmatrix} 0 \\ p^- \vec{n} \\ 0 \end{pmatrix}.$$

The viscous boundary fluxes are set to be zero.

3.4.4. Extrapolation

At extrapolation boundaries, first order extrapolation is applied, resulting in

$$\begin{aligned} u_h^b &= u_h^- \\ \Theta_h^b &= \Theta_h^- \\ \nabla u_h^b &= \nabla u_h^- \end{aligned}$$

$$r_h^b = r_h^-.$$

The numerical inviscid and viscous as well as the auxiliary extrapolation boundary fluxes are computed as

$$\begin{aligned}
\mathcal{H}_i^b &= \mathcal{H}_i\left(u_h^-, u_h^b\right) \\
\mathcal{H}_{aux}^b &= \mathcal{H}_{aux}\left(u_h^-, u_h^b\right) \\
\mathcal{H}_v^b &= \mathcal{H}_v\left(u_h^-, \Theta_h^-, u_h^b, \Theta_h^b\right) \\
\mathcal{H}_v^b &= \mathcal{H}_v\left(u_h^-, \nabla u_h^- + r_h^-, u_h^b, \nabla u_h^b + r_h^b\right).
\end{aligned}$$

3.5. Elements and basis functions

In principle the DG discretisation can be performed with arbitrary element types. In this work, we choose line, triangular, quadrilateral and tetrahedral elements for one-, two- and three-dimensional discretisations. For the line elements, the choice of the basis functions is straightforward, see section 3.5.3. For two-dimensional and three-dimensional elements, we utilise the family of orthogonal, hierarchical bases, formed from tensor products of Jacobi polynomials. Basis functions for all element types are used up to fourth order approximation accuracy. In order to generalise all operations, which depend on the element geometry, we transform the physical elements to a special reference element. This will be the subject of the next section. Then, the basis functions for the respective reference elements will be described and the integration rules for the element integrals as well as for the element boundary integrals will be presented.

3.5.1. Transformation to computational space

The transformation of general lines, triangles and quadrilaterals will be shortly summarised in the following. The transformation of a reference line into a physical line is trivial and nothing else than scaling the physical line to the unit length and shifting.

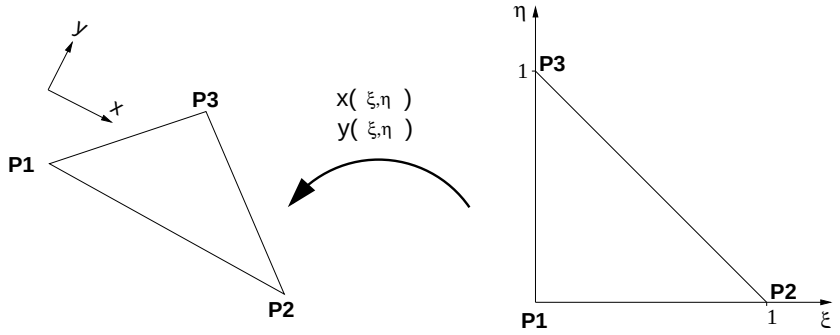
A reference triangle can be mapped from the computational space $(\xi,\, \eta)$ to an arbitrary triangle in the physical space $(x,\, y)$ (see figure 3.8(a)) with the linear transformation

$$\begin{aligned}
x(\xi,\eta) &= x_1 + (x_2 - x_1)\,\xi + (x_3 - x_1)\,\eta \\
y(\xi,\eta) &= y_1 + (y_2 - y_1)\,\xi + (y_3 - y_1)\,\eta
\end{aligned} \tag{3.42}$$

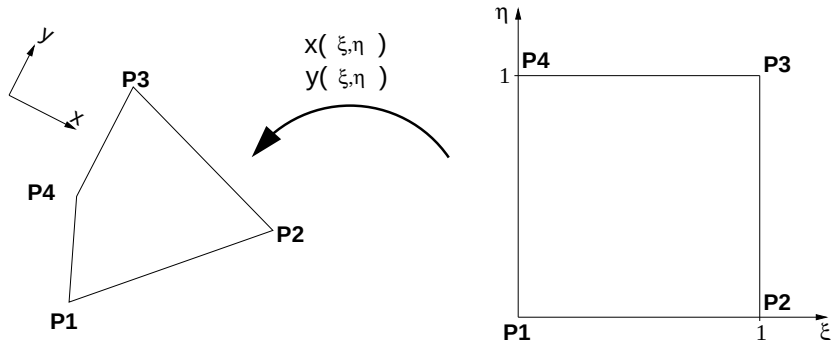
where the Jacobian of the transformation is

$$J = \left|\frac{\partial(X,Y)}{\partial(\xi,\eta)}\right| = \det\begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{pmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) = \text{const.}$$

Thus the Jacobian is independent of ξ and η and identical to the doubled area of the



(a) Triangle



(b) Quadrilateral

Figure 3.8.: Transformation from reference elements to physical elements

original triangle.

The mapping of a general quadrilateral into a reference quadrilateral (see figure 3.8(b)) is not linear. The bilinear mapping is

$$\begin{aligned} x(\xi,\eta) &= x_1(1-\xi)(1-\eta) + x_2\eta(1-\xi) + x_3\xi\eta + x_4\eta(1-\xi) \\ y(\xi,\eta) &= y_1(1-\xi)(1-\eta) + y_2\xi(1-\xi) + y_3\xi\eta + y_4\eta(1-\xi) \end{aligned} \tag{3.43}$$

The Jacobian of the transformation depends now on ξ and η

$$J(\xi,\eta) = \left|\frac{\partial(X,Y)}{\partial(\xi,\eta)}\right| = \det\begin{pmatrix} -x_1+x_2+\eta(x_1-x_2+x_3-x_4) & -y_1+y_2+\eta(y_1-y_2+y_3-y_4) \\ -x_1+x_4+\xi(x_1-x_2+x_3-x_4) & -y_1+y_4+\xi(y_1-y_2+y_3-y_4) \end{pmatrix}.$$

3.5.2. Gaussian integration in the computational space

The numerical integration of cell and face integrals is performed in the reference elements. Therefore the above described transformations have to be considered. Without loss of generality, we exemplarily discuss this for the boundary and cell integrals for the Euler case and source term integrals, occurring in the RANS equations discretisations.

The element boundary integrals are approximated by the weighted sum

$$\begin{aligned} \oint_{\partial E} b_k(\vec{x}(t))\,\mathcal{H}_i(u(\vec{x}(t)))\cdot\vec{n}\,d\sigma &= \int_{t=0}^{1} b_k(t)\,\mathcal{H}_i(u(t))\cdot\vec{n}\,\left|\dot{\vec{x}}(t)\right|dt \\ &= \sum_{l=1}^{L} b_k(t_l)\,\omega_l\mathcal{H}_i(u(t_l))\cdot\vec{n_1}\;|\partial E| \end{aligned}$$

where $\vec{n}$ is the normalised boundary normal vector with $|\vec{n}_1| = 1$.

For the element integrals, Gaussian integration results in

$$\begin{aligned} \int_E \nabla_x b_k(\vec{x}(\xi,\eta))\cdot\mathcal{F}_i(u(\vec{x}(\xi,\eta)))\,dE &= \int_\xi\int_\eta J^{-1}\nabla_\xi b_k(\xi,\eta)\cdot\mathcal{F}_i(\xi,\eta)\;J(\xi,\eta)\,d\xi d\eta \\ &= \sum_{j=1}^{M}\omega_j\;J^{-1}(\xi_j,\eta_j)\,\nabla_\xi b_k(\xi,\eta)\cdot\mathcal{F}_i(\xi,\eta)\;|J(\xi_j,\eta_j)| \end{aligned}$$

The source term integrals are treated in the same manner:

$$\begin{aligned} \int_E b_k(\vec{x}(\xi,\eta))\;S(u(\vec{x}(\xi,\eta)),\nabla u(\xi,\eta))\;dE &= \int_\xi\int_\eta J^{-1}b_k(\xi,\eta)\,S(\xi,\eta)\;J(\xi,\eta)\,d\xi d\eta \\ &= \sum_{j=1}^{M}\omega_j\;J^{-1}(\xi_j,\eta_j)\,b_k(\xi_j,\eta_j)\,S(\xi_j,\eta_j)\;|J(\xi_j,\eta_j)| \end{aligned}$$

The used integration rules for lines, triangles and quadrilaterals are given in appendix C.

3.5.3. Line elements

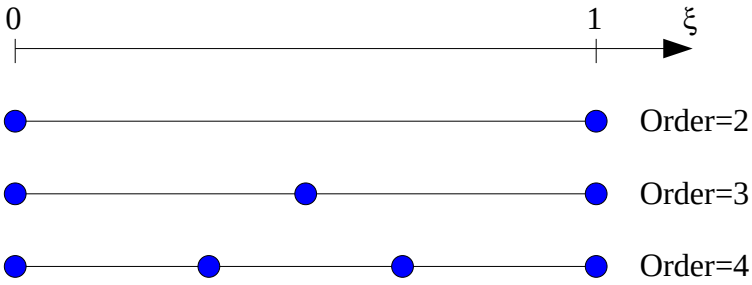
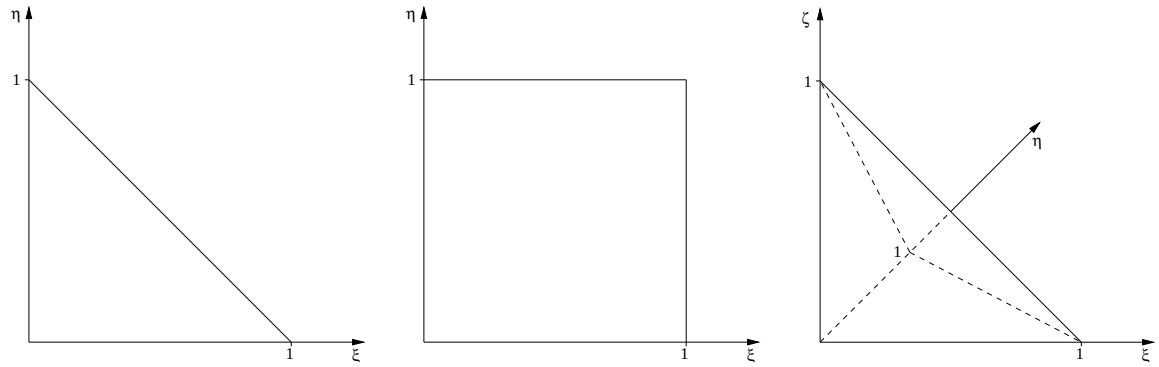


Figure 3.9.: Reference element and position of degrees of freedom (blue points)

The reference element is just a simple line of unit length. The basis functions for lines are nodal based, meaning that the degrees of freedom represent values at distinct internal element positions, see figure 3.9. The resulting basis functions are given in appendix B.

3.5.4. Triangular and quadrilateral elements



(a) Triangle (b) Square (c) Tetrahedron

Figure 3.10.: Reference elements in computational space

The polynomial basis functions according to Karniadakis, Sherwin and Warburton [56, 104] are based on transformation of triangles and tetrahedron to quadrilaterals and hexaeder, respectively. An appealing property of the spectral basis is that it is orthogonal. This reduces computational effort since all mass matrices are diagonal and their inversion is trivial. Another beneficial property is that all bases are hierarchical, which means that increasing the order only adds basis function b_k to the existing basis functions, see appendix B. This can simplify the implementational expenditure of p-adaptation and p-multigrid, since the required transformations between different approximation levels are straightforward.

The basis functions are given in terms of Jacobi polynomials. These are defined as the polynomial solutions of the Sturm-Liouville problem and can be obtained as

$$P_n^{\alpha,\beta}(x) = \frac{(-1)^n}{2^n n!}(1-x)^{\alpha}(1+x)^{-\beta}\frac{d^n}{dx^n}\left\{(1-x)^{\alpha+n}(1+x)^{\beta+n}\right\}$$

The basis functions are constructed with respect to the following three principal functions

$$\phi_p^a(z) = P_p^{0,0}(z), \quad \phi_{pq}^b(z) = \left(\frac{1-z}{2}\right)^i P_q^{2p+1,0}(z), \quad \phi_{pqr}^c(z) = \left(\frac{1-z}{2}\right)^{p+q} P_r^{2p+2q+2,0}(z)$$

For the reference quadrilateral the basis functions b_k are defined in terms of the principal functions as

$$b_k(\xi,\eta) = \phi_p^a(-1+2\xi)\,\phi_{pq}^b(-1+2\eta) \tag{3.44}$$

For the reference triangle the basis functions b_k are defined in terms of the principle functions as

$$b_k(\xi,\eta) = \phi_p^a\left(\frac{2\xi}{1-\eta}-1\right)\phi_{pq}^b(-1+2\eta) \tag{3.45}$$

where, for a polynomial basis of order N, the index ranges of i and j for both (triangles and quadrilaterals) are

$$0 \le p \le N, \qquad 0 \le q \le N, \qquad p+q \le N \tag{3.46}$$

The index k in the basis functions b_k is a function of the pair (p,q) and the design order N:

$$k = p + (N+1)\,q - \frac{q(q-1)}{2}$$

$$0 \le k \le \frac{(N+1)(N+2)}{2} - 1$$

Hence, the number of basis functions is defined by $\frac{(N+1)(N+2)}{2}$, see also table 3.2. It is identical to the number of possible permutations (p,q), which can be formed under the restriction(3.46). If we consider for example a linear approximation ($N=1$), we have three permutations for the pair (p,q). In detail, they are $(0,0)$, $(1,0)$, $(0,1)$, resulting in three basis functions. All polynomial basis functions, needed to construct a complete polynomial basis up to the order of four ($N=3$), are tabulated in appendix B. For tetrahedral elements the basis functions b_k are defined in terms of the principal functions as

$$b_k(\xi,\eta) = \phi_p^a\left(\frac{2\zeta}{1-\xi-\eta}-1\right)\phi_{pq}^b\left(\frac{2\xi}{1-\eta}-1\right)\phi_{pqr}^c(-1+2\eta) \tag{3.47}$$

For a polynomial basis of order N, the index ranges are

$$0 \le p \le N, \qquad 0 \le q \le N, \qquad 0 \le r \le N, \qquad p+q+r \le N$$

Element type/Order N	**0**	**1**	**2**	**3**	*4*	*5*
Line	**1**	**2**	**3**	**4**	*5*	*6*
Triangle	**1**	**3**	**6**	**10**	*15*	*21*
Quadrangle	**1**	**3**	**6**	**10**	*15*	*21*
Tetraeder	**1**	**4**	**10**	**20**	*35*	*56*

Table 3.2.: Number of basis functions, required to form a complete polynomial basis of order N. (Implemented orders up to $N = 3$)

The index k is limited to

$$0 \leq k \leq \frac{(N+1)(N+2)(N+3)}{6} - 1$$

Hence, the required number of polynomial basis functions, in order to form a complete polynomial basis of order N is $\frac{(N+1)(N+2)(N+3)}{6}$, see Table 3.2.

3.5.5. Mass matrices

In chapter 3.1 we already introduced the so called element mass matrix M. The integration required for the elements M_{kj} of course is performed in the computational space

$$M_{kj} = \int_{\xi}\int_{\eta} b_k(\xi,\eta)\, b_j(\xi,\eta)\, |J(\xi_j,\eta_j)|\, d\xi d\eta. \tag{3.48}$$

In the triangular case, the elemental mass matrices possess diagonal form, meaning that

$$\int_{\xi}\int_{\eta} b_k b_j\, |J(\xi_j,\eta_j)|\, d\xi d\eta = \delta_{kj}\, 2A.$$

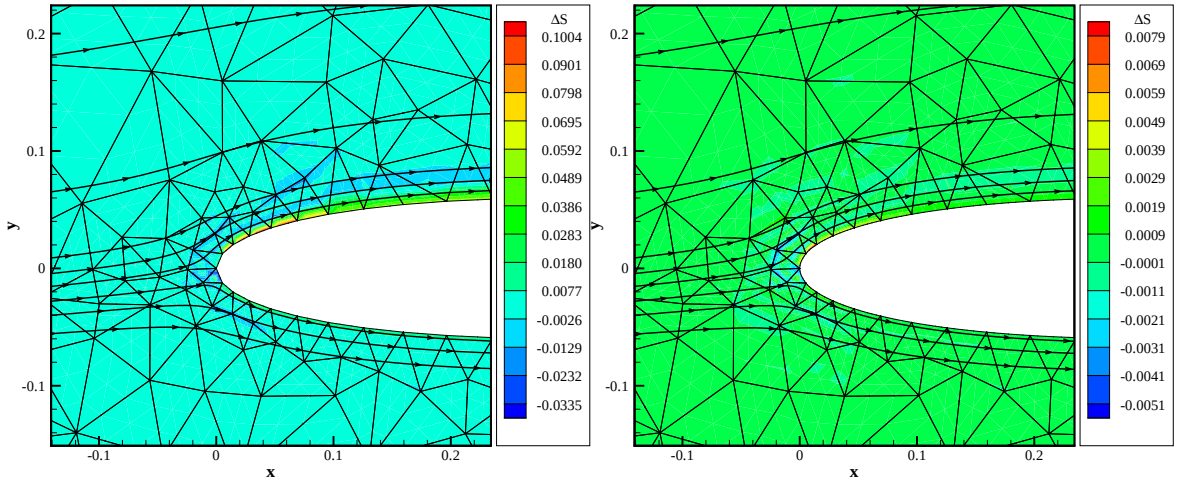
Since the Jacobian is nothing else than the doubled area of the triangle in the physical space.

For quadrilaterals M is only diagonal, if two opposite sides are parallel, otherwise the Jacobian is a function of space and cell geometry and hence M is only symmetric. Therefore, the quadrilateral mass matrices are computed during preprocessing and stored in memory.

3.6. High-order boundaries

When using schemes of order higher than two, it quickly becomes very clear that the discretisation exactly resolves the physics of such a polygonal contour—at the kinks happen separation, entropy and turbulence production, and so on. The entropy production of an inviscid Euler simulation of NACA0012 airfoil flow, is shown for example in figure 3.11(a). If the true surface is indeed smooth, it has to be modeled as such, using

elements with curved boundaries, where the production of entropy is reduced by orders of magnitude near the airfoil boundary, see Figure 3.11(b). Normal vector continuity is therefore strictly required everywhere except for deliberate corners, like for example at a trailing edge of an airfoil.



(a) Straight wall boundary elements

(b) Curved wall boundary elements

Figure 3.11.: Comparison of entropy production for an Euler simulation of the NACA0012 airfoil ($\mathrm{Ma}_\infty = 0.63$, $\alpha = 2\,\mathrm{deg.}$)

Usually, curved elements are classified in sub-, iso- and super-parametrical elements by comparing the polynomial degree of the curved boundary representation N_B with the polynomial degree N of the spatial discretisation of the numerical scheme. In detail we have the mapping

$$\begin{aligned} N_B < N &\quad : \quad \text{subparam. element} \\ N_B = N &\quad : \quad \text{isoparam. element} \\ N_B > N &\quad : \quad \text{superparam. element} \end{aligned}$$

Since the boundary representation, which will be described in the next section, is always of cubic nature ($N_B = 3$) and the currently implemented maximum spatial computation order is four ($N = 3$), all the calculations in this thesis are either performed with super-parametrical or at least with iso-parametrical elements. Hence, as desired, the overall computational order of a simulation is only determined by the polynomial degree N of the test and ansatz functions.

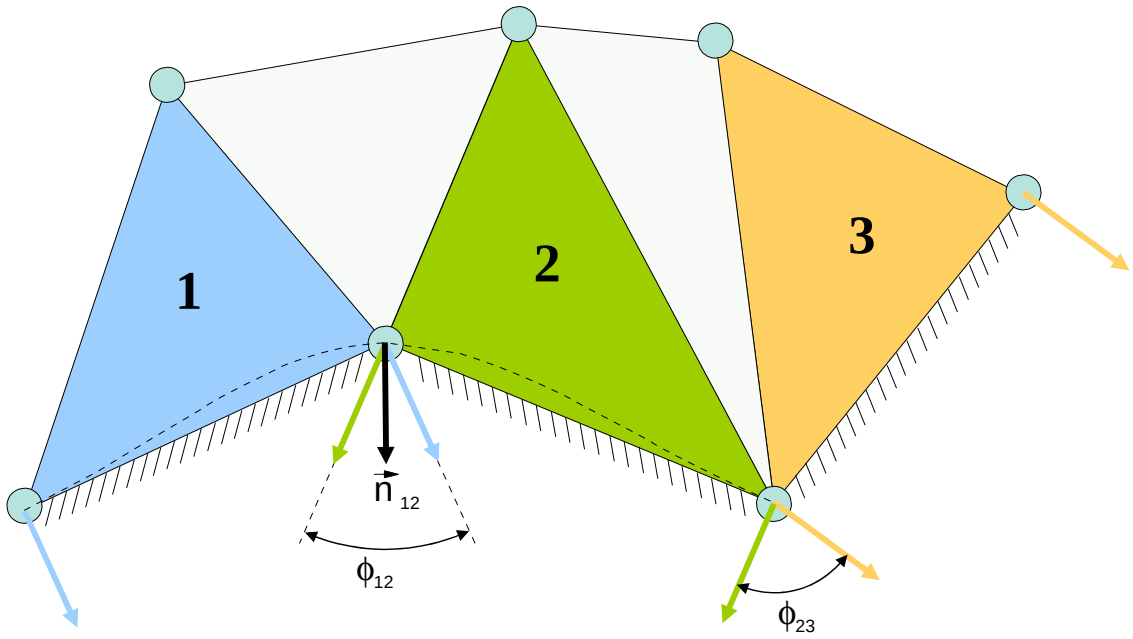


Figure 3.12.: Averaged normal vectors and curvelinear boundary (dashed line).

3.6.1. Boundary representation

In our case curvilinear boundaries are always approximated by piecewise cubic functions with continuous normal vectors across elements, see figure 3.12. The normal vector at the kinks of the polygonal representation is constructed as a (length weighted) average of the adjacent face normals. This can be done locally, without splining the entire surface, in order to avoid global information transfer with the corresponding parallelisation penalties, but at the cost of giving up the minimum curvature condition of a true spline. In fact, if only a grid is given, without further information, a better approximation of the boundary is pretty pointless, because it is impossible to devise the true geometry from just an agglomeration of points. Therefore it seems questionable to approximate an unknown true boundary better than C^1-continuous. A further argument using simple polynomials is, that the transformation of a fully splined boundary element to the computational space is more costly than that of a simple polynomial.

The real piece-wise polynomial ansatz is

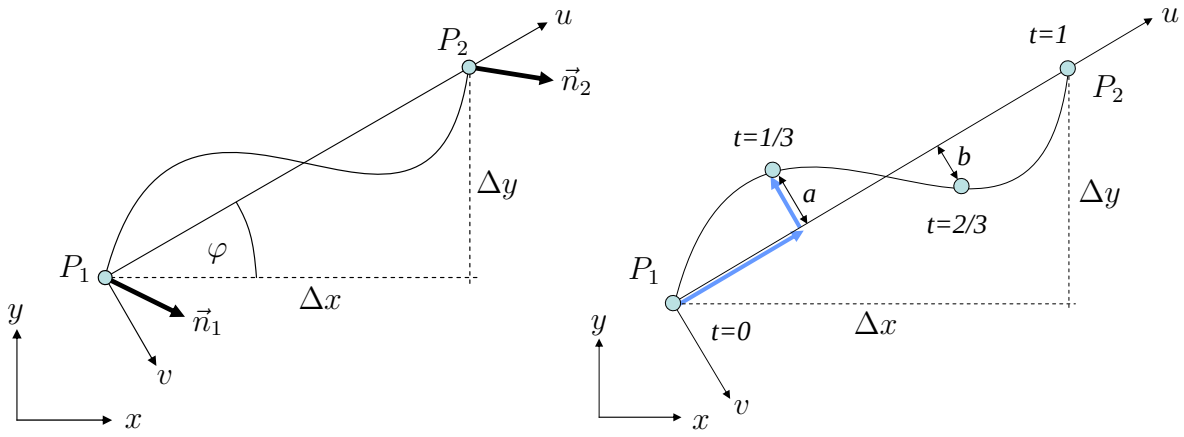
$$y(x) = ax^3 + bx^2 + cx + d.$$

The parameterised representation of such a curved edge is cubic

$$\vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} at^3 + bt^2 + ct + d \\ et^3 + ft^2 + gt + h \end{pmatrix}, \; 0 \leq t \leq 1. \tag{3.49}$$

The coefficients $(a, b, \ldots, h)$ in (3.49) are fully defined by the two endpoints P_1 and P_2

and the two weighted normal vectors $\vec{n}_1$ and $\vec{n}_2$ at the endpoints, see figure 3.13(a).



(a) as function of normal vectors

(b) as function of curvation parameters α and β

Figure 3.13.: Curved boundary representation in the reference system

The coefficients are

$$\begin{aligned} a &= -\frac{\Delta y\ (\Delta y\, n1_2 + \Delta x\, n1_1)}{\Delta y\, n1_1 - \Delta x\, n1_2} - \frac{\Delta y\ (\Delta y\, n2_2 + \Delta x\, n2_1)}{\Delta y\, n2_1 - \Delta x\, n2_2} \\ b &= \frac{2\Delta y\ (\Delta y\, n1_2 + \Delta x\, n1_1)}{\Delta y\, n1_1 - \Delta x\, n1_2} + \frac{\Delta y\ (\Delta y\, n2_2 + \Delta x\, n2_1)}{\Delta y\, n2_1 - \Delta x\, n2_2} \\ c &= \Delta x - \frac{\Delta y\ (\Delta y\, n1_2 + \Delta x\, n1_1)}{\Delta y\, n1_1 - \Delta x\, n1_2} \\ d &= P1_1, \end{aligned}$$

and

$$\begin{aligned} e &= \frac{\Delta x\ (\Delta y\, n1_2 + \Delta x\, n1_1)}{\Delta y\, n1_1 - \Delta x\, n1_2} + \frac{\Delta x\ (\Delta y\, n2_2 + \Delta x\, n2_1)}{\Delta y\, n2_1 - \Delta x\, n2_2} \\ f &= -\frac{2\Delta x\ (\Delta y\, n1_2 + \Delta x\, n1_1)}{\Delta y\, n1_1 - \Delta x\, n1_2} - \frac{\Delta x\ (\Delta y\, n2_2 + \Delta x\, n2_1)}{\Delta y\, n2_1 - \Delta x\, n2_2} \\ g &= \Delta y + \frac{\Delta x\ (\Delta y\, n1_2 + \Delta x\, n1_1)}{\Delta y\, n1_1 - \Delta x\, n1_2} \\ h &= P1_2. \end{aligned}$$

In order to obtain an easier transformation, we express the function $\vec{x}\,(t,\ \vec{n}_1,\ \vec{n}_2,\ P_1,\ P_2)$ in alternative parameters, namely $\vec{x}\,(t,\ \alpha,\ \beta,\ P_1,\ P_2)$ where the parameters α and β are nothing else than the distance between the curved and the straight edge at $t = 1/3$

and $t = 2/3$, respectively, see figure 3.13(b). These two curvature parameters are used instead of the normal vectors $\vec{n}_1$ and $\vec{n}_2$ now and we obtain

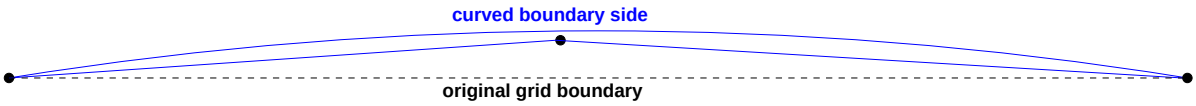
$$\vec{x}(t) = \begin{pmatrix} \frac{27}{2}(\alpha-\beta)\Delta y\, t^3 + \left(18\beta - \frac{45}{2}\alpha\right)\Delta y\, t^2 + \left(\Delta x + 9\left(\alpha - \frac{1}{2}\beta\right)\Delta y\right) t + P1_1 \\ -\frac{27}{2}(\alpha-\beta)\Delta x\, t^3 - \left(18\beta - \frac{45}{2}\alpha\right)\Delta x\, t^2 + \left(\Delta y + 9\left(\alpha - \frac{1}{2}\beta\right)\Delta x\right) t + P1_2 \end{pmatrix} \tag{3.50}$$

as alternative mathematical and compact representation of a curved edge.

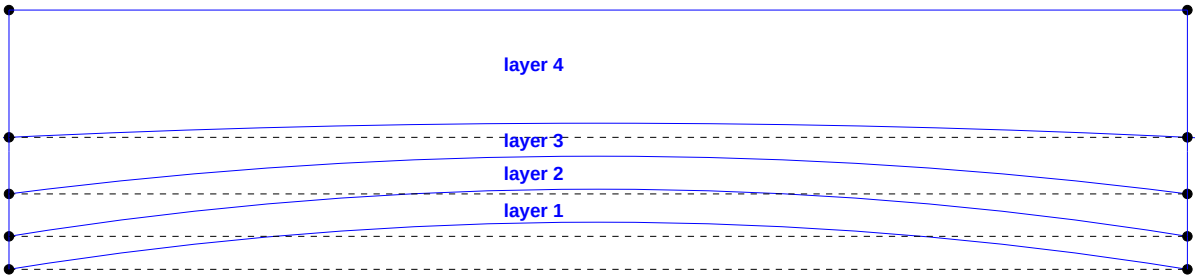
Consequently, the normal vector of a curved edge is a function of the curve parameter t now and can be calculated from (3.50) by simple differentiation

$$\vec{n}(t) = \begin{pmatrix} \frac{dy(t)}{dt} \\ -\frac{dx(t)}{dt} \end{pmatrix} = \begin{pmatrix} \frac{81}{2}(\beta-\alpha)\Delta x\, t^2 + 2\left(\frac{45}{2}\alpha - 18\beta\right)\Delta x\, t + \Delta y + 9\left(\frac{1}{2}\beta - \alpha\right)\Delta x \\ \frac{81}{2}(\beta-\alpha)\Delta y\, t^2 + 2\left(\frac{45}{2}\alpha - 18\beta\right)\Delta y\, t - \Delta x + 9\left(\frac{1}{2}\beta - \alpha\right)\Delta y \end{pmatrix}$$

In high Reynolds number simulations the boundary layer is very thin, such that cells with high aspect ratios are required in order to resolve the boundary layer. Even if high order elements reduce this requirement a little bit, it is still advisable to have some 4-8 cells in wall normal direction in the boundary layer. However, for high aspect ratio cells, the curvature can quickly become larger than the height of the cell, resulting in singular and negative Jacobians of the transformation, see figure 3.14(a). To overcome this problem, it is reasonable to use quadrilaterals in the boundary layer, where the upper edge is curved as well, in order to keep the transformation distortions at an acceptable level. If the curvature of the upper edge is slightly less than the lower one, the entire curvation can be put down within a few layers of curved boundary cells, see figure 3.14(b). Farther out the cells may remain straight, with the resulting increase in computational efficiency.



(a) Degenerated boundary triangle



(b) Quadrilateral grid layers

Figure 3.14.: Boundary layer resolution with triangle and quadrilaterals

We can use the same cubic description for the upper and lower edges as described above for a curved triangle edge, where the upper curved edge is constructed with the same normal vector continuity criteria used for the wall attached edge. In other words, all outer curved layers are treated like the wall attached layer. Logically, this kind of grid deforming technique requires a quadrilateral grid, where the layers extend in a normal manner from the wall. But that is only a minor constraint of the method, because modern hybrid unstructured grid generators anyway fulfill this demand, in order to improve solution quality (for a FV method) in the boundary layer.

In order to assess our compact interpolation method described above, we also implemented the familiar spline approach, where the airfoil contour is approximated with the aid of natural polynomial splines. Since airfoils are often represented by an agglomeration of points $\vec{x}_i$ in a mesh file, we first have to introduce a parameterisation

$$\vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \; 0 \leq t \leq t_{\text{end}}.$$

If necessary, we re-sort the boundary nodes $\vec{x}_i$ counter-clockwise and then we choose a chordal parameterisation, where the parameter values t_i at the meshpoints are defined with the aid of the distances of boundary consecutive points

$$t_0 = 0, \; t_i = t_{i-1} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \quad i = 1, 2, \ldots, n.$$

that, the parameters $x(t)$ and $y(t)$ are splined independently along t. A cubic natural polynomial spline is subject to the following conditions: The curve itself and additionally

the first as well as the second derivative is continuous at the nodal boundary points $\vec{x}_i$. These conditions lead to a tridiagonal system of linear equations, which is strongly diagonal dominant and easy to solve by the well known Thomas algorithm, see for example [80].

3.6.2. Transformation

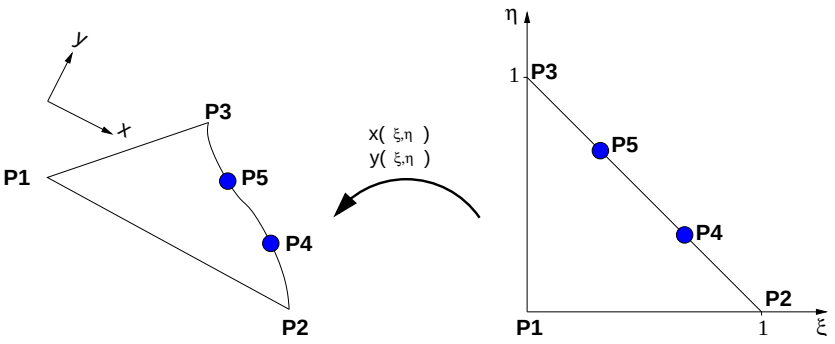
The curved edged elements (triangles and quadrilaterals) prescribed in the previous section are transformed into reference elements for computational reasons again. Since the curvilinear mathematical representations are cubic functions now, we have to increase the degree N of the polynomial transformation

$$\begin{aligned} x(\xi,\eta) &= \sum_{m=0}^{N}\sum_{n=0}^{N-m}\gamma_{mn}\xi^m\eta^n \\ y(\xi,\eta) &= \sum_{m=0}^{N}\sum_{n=0}^{N-m}\delta_{mn}\xi^m\eta^n \end{aligned} \tag{3.51}$$

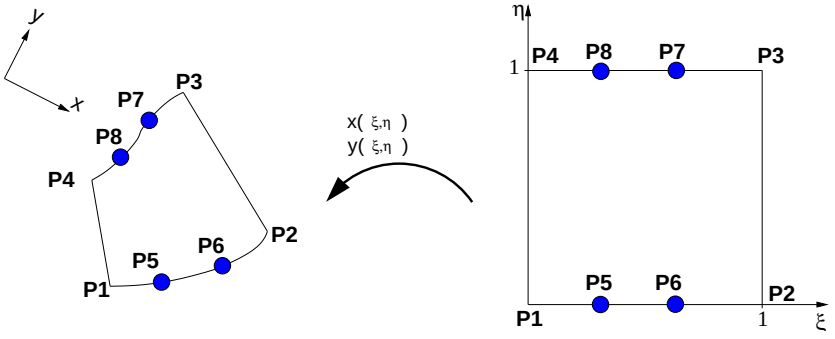
If the curvature is approximated with cubic functions as prescribed above a cubic polynomial ($N = 3$) is also required for transformation to the computational space. For the special case that only one edge of a boundary triangle is curved, this transformation polynomial strongly simplifies ($\gamma_{0,2} = \gamma_{0,3} = \gamma_{2,0} = \gamma_{3,0} = \delta_{0,2} = \delta_{0,3} = \delta_{2,0} = \delta_{3,0}$) and we obtain the transformation law

$$\begin{aligned} x(\xi,\eta) &= \gamma_{0,0} + \gamma_{0,1}\eta + \gamma_{1,0}\xi + \gamma_{1,1}\xi\,\eta + \gamma_{1,2}\xi\,\eta^2 + \gamma_{2,1}\xi^2\eta \\ y(\xi,\eta) &= \delta_{0,0} + \delta_{0,1}\eta + \delta_{1,0}\xi + \delta_{1,1}\xi\,\eta + \delta_{1,2}\xi\,\eta^2 + \delta_{2,1}\xi^2\eta \end{aligned} \tag{3.52}$$

The unknown coefficients γ and δ can be obtained by mapping of several points, where the original and transformed coordinates are known. Here, the system (3.52) is fully determined with the corner points and two points on the curved edge, see Figure 3.15(a).



(a) Boundary triangle (one side curved)



(b) Boundary quadrilateral (two sides curved)

Figure 3.15.: Transformation of the reference elements to curved elements (extra points in blue needed for curvilinear transformations)

In principal, two parameters (e.g. $\gamma_{2,1}$ and $\delta_{2,1}$) can be arbitrarily chosen. The resulting coefficients are

$$\begin{aligned}
\gamma_{0,0} &= P1_1 \\
\gamma_{0,1} &= -P1_1 + P3_1 \\
\gamma_{1,0} &= -P1_1 + P2_1 \\
\gamma_{1,1} &= -\frac{9}{2}P2_1 + 9P4_1 - \frac{9}{2}P5_1 - \gamma_{2,1} \\
\gamma_{1,2} &= \frac{9}{2}P2_1 - \frac{27}{2}P4_1 + \frac{27}{2}P5_1 - \frac{9}{2}P3_1 + \gamma_{2,1}
\end{aligned}$$

$$\begin{aligned}
\delta_{0,0} &= P1_2 \\
\delta_{0,1} &= -P1_2 + P3_2 \\
\delta_{1,0} &= -P1_2 + P2_2
\end{aligned}$$

$$\begin{aligned}
\delta_{1,1} &= -\frac{9}{2}P2_2 + 9P4_2 - \frac{9}{2}P5_2 - \delta_{2,1} \\
\delta_{1,2} &= \frac{9}{2}P2_2 - \frac{27}{2}P4_2 + \frac{27}{2}P5_2 - \frac{9}{2}P3_2 + \delta_{2,1}
\end{aligned}$$

The coefficients $\gamma_{2,1}$ and $\delta_{2,1}$ are set to zero, in order to minimise the computational effort.

Now, we can eliminate the coordinates of the points $P4$ and $P5$ by using the curvation parameters α and β introduced in the previous section

$$\begin{aligned}
P4 &= \vec{x}(t=1/3) = \begin{pmatrix} P2_1 \\ P2_2 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} + \alpha\begin{pmatrix} \Delta y \\ -\Delta x \end{pmatrix} \\
P5 &= \vec{x}(t=2/3) = \begin{pmatrix} P2_1 \\ P2_2 \end{pmatrix} + \frac{2}{3}\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} + \beta\begin{pmatrix} \Delta y \\ -\Delta x \end{pmatrix} \\
\Delta x &= P3_1 - P2_1 \\
\Delta y &= P3_2 - P2_2
\end{aligned}$$

The resulting cubic transformation is

$$\begin{aligned}
x(\xi,\eta,\alpha,\beta) &= P1_1 + (-P1_1 + P3_1)\,\eta + (-P1_1 + P2_1)\,\xi \\
&\quad + \left(9\alpha\,(P3_2 - P2_2) - \frac{9}{2}\beta\,(P3_2 - P2_2)\right)\xi\,\eta \\
&\quad + \left(-\frac{27}{2}\alpha\,(P3_2 - P2_2) + \frac{27}{2}\beta\,(P3_2 - P2_2)\right)\xi\,\eta^2 \\
y(\xi,\eta,\alpha,\beta) &= P1_2 + (-P1_2 + P3_2)\,\eta + (-P1_2 + P2_2)\,\xi \\
&\quad + \left(-9\alpha\,(P3_1 - P2_1) + \frac{9}{2}\beta\,(P3_1 - P2_1)\right)\xi\,\eta \\
&\quad + \left(\frac{27}{2}\alpha\,(P3_1 - P2_1) - \frac{27}{2}\beta\,(P3_1 - P2_1)\right)\xi\,\eta^2
\end{aligned}$$

The calculation of the Jacobian $J(\xi,\eta,\alpha,\beta) = \left|\frac{\partial(X,Y)}{\partial(\xi,\eta)}\right|$ is a little bit involved, but straightforward with the help of a symbolic algebraic package like Maple.

The transformation of the two-sided curved quadrilaterals into computational space is performed with an incomplete fourth order polynomial ($N = 4$) of the form

$$\begin{aligned}
x(\xi,\eta) &= \gamma_{0,0} + \gamma_{0,1}\eta + \gamma_{0,2}\eta^2 + \gamma_{0,3}\eta^3 + \gamma_{1,0}\xi + \gamma_{1,1}\xi\,\eta \\
&\quad + \gamma_{1,2}\xi\,\eta^2 + \gamma_{1,3}\xi\,\eta^3 + \gamma_{2,0}\xi^2 + \gamma_{2,1}\xi^2\eta + \gamma_{3,0}\xi^3 + \gamma_{3,1}\xi^3\eta \\
y(\xi,\eta) &= \delta_{0,0} + \delta_{0,1}\eta + \delta_{0,2}\eta^2 + \delta_{0,3}\eta^3 + \delta_{1,0}\xi + \delta_{1,1}\xi\,\eta \\
&\quad + \delta_{1,2}\xi\,\eta^2 + \delta_{1,3}\xi\,\eta^3 + \delta_{2,0}\xi^2 + \delta_{2,1}\xi^2\eta + \delta_{3,0}\xi^3 + \delta_{3,1}\xi^3\eta
\end{aligned} \tag{3.53}$$

The coefficients are determined by the mapping of the four corner points and two points on the sides, respectively. Eliminating the points on the curved edges by using

the curvation parameters α, β, γ and δ the resulting transformation looks like

$$\begin{aligned}
x(\xi,\eta) &= \frac{27}{2}\left(-\alpha\,(X2_2 - X1_2) + \beta\,(X2_2 - X1_2) + \gamma\,(X3_2 - X4_2) - \delta\,(X3_2 - X4_2)\right)\xi^3\eta \\
&+ \left(\frac{27}{2}\,\alpha\,(X2_2 - X1_2) - \frac{27}{2}\,\beta\,(X2_2 - X1_2)\right)\xi^3 \\
&+ \left(\frac{45}{2}\,\alpha\,(X2_2 - X1_2) - 18\,\beta\,(X2_2 - X1_2)\left(18\delta - \frac{45}{2}\gamma\right)(X3_2 - X4_2)\right)\xi^2\eta \\
&+ \left(-\frac{45}{2}\,\alpha\,(X2_2 - X1_2) + 18\,\beta\,(X2_2 - X1_2)\right)\xi^2 \\
&+ \left(X1_1 - X2_1 + X3_1 - X4_1 - 9\left(\left(\alpha - \frac{\beta}{2}\right)(X2_2 - X1_2) + \left(\frac{\delta}{2} - \gamma\right)(X3_2 - X4_2)\right)\right)\xi\eta \\
&+ \left(-X1_1 + X2_1 + 9\,\alpha\,(X2_2 - X1_2) - \frac{9}{2}\,\beta\,(X2_2 - X1_2)\right)\xi \\
&+ (X4_1 - X1_1)\,\eta \\
&+ X1_1
\end{aligned}$$

$$\begin{aligned}
y(\xi,\eta) &= \frac{27}{2}\left(\alpha\,(X2_1 - X1_1) - \beta\,(X2_1 - X1_1) - \gamma\,(X3_1 - X4_1) + \delta\,(X3_1 - X4_1)\right)\xi^3\eta \\
&+ \left(-\frac{27}{2}\,\alpha\,(X2_1 - X1_1) + \frac{27}{2}\,\beta\,(X2_1 - X1_1)\right)\xi^3 \\
&+ \left(-\frac{45}{2}\,\alpha\,(X2_1 - X1_1) + 18\,\beta\,(X2_1 - X1_1) + \left(\frac{45}{2}\gamma - 18\delta\right)(X3_1 - X4_1)\right)\xi^2\eta \\
&+ \left(\frac{45}{2}\,\alpha\,(X2_1 - X1_1) - 18\,\beta\,(X2_1 - X1_1)\right)\xi^2 \\
&+ \left(X1_2 - X2_2 + X3_2 - X4_2 + 9\left(\left(\alpha - \frac{\beta}{2}\right)(X2_1 - X1_1) + \left(\frac{\delta}{2} - \gamma\right)(X3_1 - X4_1)\right)\right)\xi\,\eta \\
&+ \left(-X1_2 + X2_2 - 9\,\alpha\,(X2_1 - X1_1) + \frac{9}{2}\,\beta\,(X2_1 - X1_1)\right)\xi \\
&+ (X4_2 - X1_2)\,\eta \\
&+ X1_2
\end{aligned}$$

The calculation of the Jacobian $J(\xi,\eta,\alpha,\beta,\gamma,\delta) = \left|\frac{\partial(X,Y)}{\partial(\xi,\eta)}\right|$ is again involved, but Maple is helpful one more time.

The Gaussian formula for the integration of the element boundary integrals presented in section 3.5.2 has to be modified for the curved cell geometries. Since the boundary normal vector $\vec{n}(t)$ is a function of the edge parameter t now, the weighted sum changes to

$$\oint_{\partial E} b_k(\vec{x}(t))\,\mathcal{H}_i(u(\vec{x}(t)))\cdot\vec{n}\,d\sigma = \int_{t=0}^{1} b_k(t)\,\mathcal{H}_i(u(t))\cdot\vec{n}\,\left|\dot{\vec{x}}(t)\right|dt$$

$$= \sum_{l=1}^{L} b_k(t_l)\,\omega_l \mathcal{H}_i(u(t_l)) \cdot \vec{n}(t_l)$$

where $\vec{n}$ now is the not normalised boundary normal with $|\vec{n}| = \sqrt{(\dot{x}(t))^2 + (\dot{y}(t))^2}$. The integration of the element integrals is also valid for curved elements and does not have to be changed. The mass matrices of curved triangles are dependent on the cell geometry (curvation parameters α, β, γ and δ) and therefore in general not diagonal anymore. For reason of efficiency all mass matrices are computed and stored at the begin of computation.

Another extra expenditure is the more involved computation of the wall distance for curved boundaries, which is strictly required by the SA turbulence model. We split the curved edge in several straight sub edges and use our original method for simple straight boundary edges. Consequently, the computational effort for the wall distance is increased, but since body and grid deformations are not included in our solution method so far, the distance is just precomputed once and simply stored. As deforming bodies will be an objective of future aeroelastic research, the wall distance computation, which then has to be accomplished every single timestep, has to be accelerated, in order to avoid that the overall computation is mainly dominated by determination of the wall distance. An alternative would be a (wall distance) computation, based on the solution of partial differential equations, see for example [96].

4. Time integration method

The spatial discretisation of the governing equations with the Discontinuous Galerkin method leads to a system of ordinary differential equations (ODE's) in time:

$$M \frac{dU}{dt} = W(U) \tag{4.1}$$

where $U = (U_1, U_2, \ldots U_k, \ldots)^T$ is the global vector of degrees of freedom to be evolved in time. Each U_k itself is a vector, that represents the state degrees of freedom of element k. For example U_k for a turbulent simulation with the SA model or the $\mathrm{k}-\omega$ model is

$$\begin{aligned} U_k^{\mathrm{SA}} &= (\rho_0, \cdots, \rho_m, \rho u_0, \cdots, \rho u_m, \rho v_0, \cdots, \rho v_m, \cdots \rho E_0, \cdots, \rho E_m, \\ &\quad \rho\tilde{\nu}_0, \cdots, \rho\tilde{\nu}_m) \\ U_k{}^{\mathrm{k}-\omega} &= (\rho_0, \cdots, \rho_m, \rho u_0, \cdots, \rho u_m, \rho v_0, \cdots, \rho v_m, \rho E_0, \cdots, \rho E_m, \\ &\quad \rho k_0, \cdots, \rho k_m, \rho\omega_0, \cdots, \rho\omega_m) \end{aligned}$$

where m is the number of degrees of freedom per cell k.

$W = (w_1, w_2, \ldots w_k, \ldots)^T$ is the global residual vector, where again the vectors w_k are the elemental residual contributions of element k. M is the (block-diagonal) global mass matrix consisting of the elemental (diagonal) mass matrices discussed in chapter 3.5.5. The block diagonality of the overall mass matrix as well as the diagonality of the element mass matrices is only guaranteed for straight-edged triangles and for quadrilaterals, where two opposite sides are parallel.

(4.1) is a time-continuous equation for the degrees of freedom, and any integration scheme applicable to ODE's may be used. A distinction is made between explicit and implicit methods.

The explicit methods are straightforward to implement but the computational time step is limited and therefore the convergence to a steady state could be very time consuming. Especially for viscous (turbulent) flow simulations the time step limit is a knock out criterion for an explicit scheme like the Runge-Kutta method described in section 4.1. Note, that in contrast to the continuous Galerkin method, where typically a global assembly process is needed, the advancement in DG is extremely local. That means, that for the case of an explicit time integration, the solution can be advanced locally on the element basis—no global assembly is needed.

Another approach, which can be used to overcome the severe time step barrier, is to use an implicit scheme for the integration of the diffusive terms and an explicit scheme

for the convective ones. Such schemes are known as implicit explicit (IMEX) schemes [58]. These schemes allow to increase the time step from the diffusive DFL to the convective CFL number. This could be sufficient for unsteady simulations with required high accuracy in time, if we think on DNS or LES.

On the other hand, certain fully implicit schemes do not have theoretical time step limitations. Since, in this study, work is mainly concentrated on the simulation of the steady RANS equations a fully implicit scheme is preferred here.

4.1. Explicit time integration method

If we use an explicit approach for the discretisation, the right hand side (RHS) of equation (4.1) is taken at the "old" time level t_n:

$$M \frac{dU}{dt} = W(U^n) \tag{4.2}$$

For time integration we use several one step Runge-Kutta (RK) type schemes. The basic idea of the Runge-Kutta method is to evaluate the RHS of (4.2) at several values of U in the interval $n\Delta t$ and $(n+1)\Delta t$ and to combine them in order to obtain a high-order approximation of U^{n+1}.

The m-stage RK methods, used in this work, can be summarised in the following algorithm:

$$\begin{aligned} k_i &= M^{-1} W\left(U^n + \Delta t\, a_i k_{i-1}\right),\ i = 1, \cdots, m \\ U^{n+1} &= U^n + \Delta t \sum_{i=1}^{m} \left(b_i k_i\right) \end{aligned} \tag{4.3}$$

written here for the case, where W is explicitly independent of time. Here Δt is the timestep and a_i and b_i are constant coefficients for the respective method, which are given in table 4.1. Note, that the k_i in (4.3) are only dependent of k_{i-1}, so the algorithm can be reformulated as:

$$\begin{aligned} k_i &= M^{-1} W\left(U = U^n + \Delta t\, a_i k_{i-1}\right) \\ U^{(0)} &= U^n \\ U^{(i)} &= U^{(i-1)} + \Delta t\, b_i k_i,\ i = 1, \cdots, m \\ U^{(m)} &= U^{n+1} \end{aligned} \tag{4.4}$$

The benefit of (4.4) is, that we don't need to store all k_i anymore.

We restrict to a maximum of four stages, since one has to add one/two more stages to the method to obtain a further increase in the order, shown by Butcher, see table 4.2. This is the reason, why four-stage, fourth-order RK schemes are so popular.

Table 4.1.: Used Runge-Kutta methods

m	$a = (a_1 \cdots a_m)^T$	$b = (b_1 \cdots b_m)^T$	Order
1	0	1	1
2	$(0\,1)^T$	$\left(\frac{1}{2}\,\frac{1}{2}\right)^T$	2
3	$\left(0\,\frac{1}{3}\,\frac{2}{3}\right)^T$	$\left(\frac{1}{4}\,0\,\frac{3}{4}\right)^T$	3
4	$\left(0\,\frac{1}{2}\,\frac{1}{2}\,1\right)^T$	$\left(\frac{1}{6}\,\frac{1}{3}\,\frac{1}{3}\,\frac{1}{6}\right)^T$	4

Table 4.2.: Minimum number of required stages m_{min} for a k-th order RK method by Butcher [24]

Order k	1	2	3	4	5	6	7
m_{min}	1	2	3	4	6	7	9

4.2. Implicit time integration method

If we use an implicit scheme for the discretisation in time, the right hand side (RHS) or residual of equation (4.1) is taken at the "new" unknown time level t_{n+1}:

$$M \frac{dU}{dt} = W\left(U^{n+1}\right) \tag{4.5}$$

For the time discretisation we use a simple backward Euler method:

$$M \frac{U^{n+1} - U^n}{\Delta t} = W\left(U^{n+1}\right) \tag{4.6}$$

As our governing equations are nonlinear, system (4.6) is a set of nonlinear algebraic equations for the degrees of freedom U^{n+1}.

In order to solve these kind of equations we linearise the nonlinear RHS around the global vector of degrees of freedom U:

$$W\left(U^{(k)}\right) \approx W\left(U^{(k-1)}\right) + \left(\frac{\partial W}{\partial U}\right)^{(k-1)} \left(U^{(k)} - U^{(k-1)}\right)$$

where $\left(\frac{\partial W}{\partial U}\right)^{(k-1)}$ is the so called Jacobian matrix of the system, evaluated at the approximation point $k-1$.

Then Newton's method in multi-dimension (also known as Newton-Raphson method) is chosen, and we obtain the following iterative scheme:

$$
\begin{aligned}
&U^{(0)} = U^n \\
&\underbrace{\left[\frac{M}{\Delta t} - \left(\frac{\partial W}{\partial U}\right)^{(k-1)}\right]}_{A} \underbrace{\left(U^{(k)} - U^{(k-1)}\right)}_{x} = \\
&\qquad \underbrace{W\left(U^{(k-1)}\right) - \frac{M}{\Delta t}\left(U^{(k-1)} - U^{(0)}\right)}_{b},\ k = 1 \cdots m \\
&U^{n+1} = U^{(m)}
\end{aligned}
\tag{4.7}
$$

In contrast to the simple one-dimensional Newton scheme, where the root of a single linear equation is calculated in every iteration, in multi-dimension a linear system of equations has to be solved. At the beginning of every time step, we start the Newton iteration loop with the solution U^n at the old time level as initial guess. Then, m iterations are performed in order to find the roots of the nonlinear system (4.6), or, in other words, the flow solution at the new time U^{n+1}.

If the linearisation is exact or sufficiently good, the Newton method delivers optimal quadratic-like convergence and if m tends to infinity $U^{(k)}$ tends to U^{n+1}.

$$U^{(k)}\Big|_{k\to\infty} = U^{n+1}$$

The practical choice of m depends on the desired solution quality, which can be assessed by the (L_2-)norm of the updates $\delta U = U^{(k)} - U^{(k-1)}$.

The overall implicit schedule can be regarded as three nested loops. The outermost loop, is obviously the time-stepping loop. The next inner loop is the Newton iteration loop, which itself contains the iteration loop of the linear system ($A\,x = b$ in (4.7)) solver. The pseudo code for the three loops is given in algorithm 1.

If steady-state problems are examined the iterative newton loop is degraded to one iteration ($k = 1$) for reasons of efficiency, because time-accuracy at the interim time levels is of secondary importance. The crucial part of (4.7) is the assembly of the Jacobian matrix $\frac{\partial W}{\partial U}$, which contains the derivatives of the degree of freedom residuals with respect to the degree of freedom state vectors.

Algorithm 1 Time stepping algorithm

do m=1,	# time-steps	Time loop
do k=1,	# Newton-steps	Newton-iteration loop
do l=1,	# search-steps	GMRES/BISCGSTAB loop

4.3. Jacobians—linearisation

The overall system Jacobian matrix $\frac{\partial W}{\partial U}$ is an $n \times n$ block matrix, where n denotes the number of cells in the computational domain Ω. The blocks itself are $m \times m$ elemental Jacobian matrices, where m is the number of conservative state variables to be computed (e.g. $\rho, \rho u, \rho v, \rho E, \rho\tilde{\nu}$ for a 2D turbulent simulation with SA model) times the number of used degrees of freedom, according to the used polynomial expansions. For the example of a two-dimensional turbulent simulation with the Spalart-Allmaras model (using triangular or quadrilateral elements), these block matrices have the following shape:

$$\begin{aligned}\frac{\partial w_k}{\partial u_j} &= \frac{\partial\left(w(\rho_0),\cdots,w(\rho_m),w(\rho u_0),\cdots,w(\rho u_m),\cdots\cdots,w(\rho\tilde{\nu}_0),\cdots,w(\rho\tilde{\nu}_m)\right)}{\partial\left(\rho_0,\cdots,\rho_m,\rho u_0,\cdots,\rho u_m,\rho v_0,\cdots,\rho v_m,\rho E_0,\cdots,\rho E_m,\rho\tilde{\nu}_0,\cdots,\rho\tilde{\nu}_m\right)} \\ &= \begin{pmatrix} \frac{\partial w(\rho_0)}{\partial\rho_0} & \frac{\partial w(\rho_0)}{\partial\rho_1} & \cdots & \frac{\partial w(\rho_0)}{\partial\rho\tilde{\nu}_m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial w(\rho u_0)}{\partial\rho_0} & \frac{\partial w(\rho u_0)}{\partial\rho_1} & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial w(\rho\tilde{\nu}_m)}{\partial\rho_0} & \frac{\partial w(\rho\tilde{\nu}_m)}{\partial\rho_1} & \cdots & \frac{\partial w(\rho\tilde{\nu}_m)}{\partial\rho\tilde{\nu}_m} \end{pmatrix} m \times m\end{aligned}$$

where the w_k represent the residual of an exemplary cell k and the u_j represent the degrees of freedom of an exemplary cell i contributing to the residual of cell k. Due to the compactness of the BR2 scheme (shown in chapter 3) w_k is only a function of the cell degrees of freedom u_k and their direct neighbours cell degrees of freedom u_i:

$$w_k = w_k\left(u_k, u_i\right)\ i = 1..\text{number of direct neighbour cells}$$

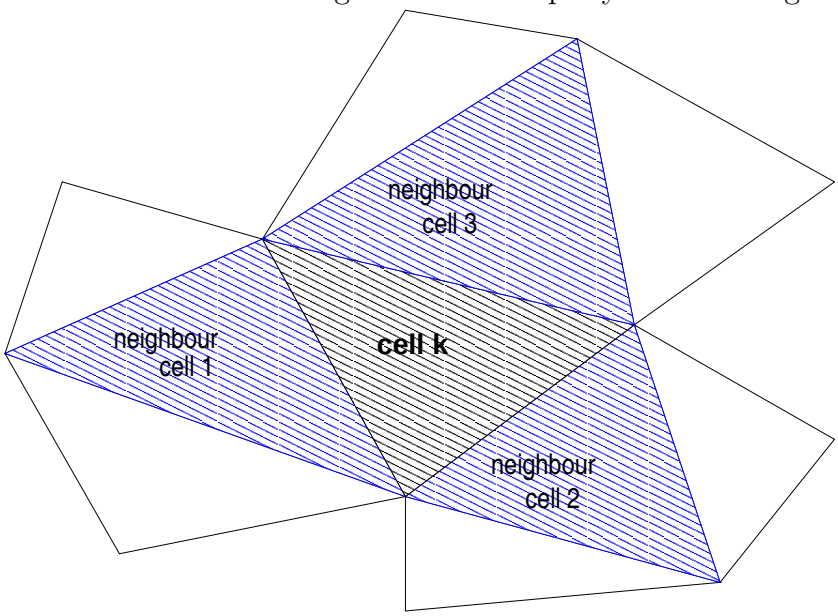
Thus the overall system Jacobian matrix $\frac{\partial W}{\partial U}$ is sparse, since the number of non-zero blocks for each (block) row is equal to the number of direct cell neigbours plus one for inner cells.

This situation can be outlined for the hatched triangle shown in figure 4.1. The contribution of triangle k leads to the following structure in (block) row k of the Jacobian $n \times n$ matrix $\frac{\partial W}{\partial U}$:

$$\frac{\partial W}{\partial U} = \begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0\cdots & \frac{\partial w_k}{\partial u_k} & \cdots 0\cdots & \frac{\partial w_k}{\partial u_1} & \cdots 0\cdots & \frac{\partial w_k}{\partial u_2} & \cdots 0\cdots & \frac{\partial w_k}{\partial u_3} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \leftarrow \text{block row k}$$

The most convenient way to construct the entries $\frac{\partial w_k}{\partial u_i}$ in the global Jacobian is to construct $\frac{\partial w_k}{\partial u_i}$ locally at the cell level first, simultaneously with the distribution of the

Figure 4.1.: Exemplary cell and neighbours



residuals, and then to put them in the correct place in the global Jacobian.

In order to construct the local Jacobian matrices $\frac{\partial w_k}{\partial u_i}$, we have to sum up the contributions arising from the linearisation of the inviscid and viscous numerical fluxes, which appear in the face integrals and the analytical fluxes and source terms, which appear in the cell integrals:

- inviscid analytical flux:

$$\mathcal{F}_i(u)^{n+1} = \mathcal{F}_i(u)^n + \left(\frac{\partial \mathcal{F}_i}{\partial u}\right)\Delta u$$

- inviscid numerical face flux:

$$\mathcal{H}_i\left(u^-, u^+\right)^{n+1} = \mathcal{H}_i\left(u^-, u^+\right)^n + \left(\frac{\partial \mathcal{H}_i}{\partial u^-}\right)\Delta u^- + \left(\frac{\partial \mathcal{H}_i}{\partial u^+}\right)\Delta u^+$$

- viscous analytic flux:

$$\begin{aligned}\mathcal{F}_v\left(u, \nabla u + R\right)^{n+1} &= \mathcal{F}_v\left(u, \nabla u + R\right)^n + \frac{\partial \mathcal{F}_v\left(u, \nabla u + R\right)}{\partial u}\Delta u \\ &+ \frac{\partial \mathcal{F}_v\left(u, \nabla u + R\right)}{\partial (\nabla u + R)}\frac{\partial (\nabla u)}{\partial u}\Delta u + \frac{\partial \mathcal{F}_v\left(u, \nabla u + R\right)}{\partial (\nabla u + R)}\frac{\partial R}{\partial u}\Delta u\end{aligned}$$

- viscid numerical face flux:

$$\begin{aligned}
&\mathcal{H}_v\left(u^-,u^+,(\nabla u)^- + r^-,(\nabla u)^+ + r^+\right)^{n+1} \quad = \\
&\mathcal{H}_v\left(u^-,u^+,(\nabla u)^- + r^-,(\nabla u)^+ + r^+\right)^{n} + \frac{\partial \mathcal{H}_v}{\partial u^-}\Delta u^- + \frac{\partial \mathcal{H}_v}{\partial u^+}\Delta u^+ \\
&+\frac{\partial \mathcal{H}_v}{\partial\left((\nabla u)^- + r^-\right)}\frac{\partial\,(\nabla u)^-}{\partial u^-}\Delta u^- + \frac{\partial \mathcal{H}_v}{\partial\left((\nabla u)^+ + r^+\right)}\frac{\partial\,(\nabla u)^+}{\partial u^+}\Delta u^+ \\
&+\frac{\partial \mathcal{H}_v}{\partial\left((\nabla u)^- + r^-\right)}\frac{\partial r^-}{\partial u^-}\Delta u^- + \frac{\partial \mathcal{H}_v}{\partial\left((\nabla u)^+ + r^+\right)}\frac{\partial r^+}{\partial u^+}\Delta u^+
\end{aligned}$$

- analytic source term:

$$\begin{aligned}
S\left(u,\nabla u + R\right)^{n+1} \quad = \quad & S\left(u,\nabla u + R\right)^{n} + \frac{\partial S\left(u,\nabla u + R\right)}{\partial u}\Delta u \\
&+\frac{\partial S\left(u,\nabla u + R\right)}{\partial(\nabla u + R)}\frac{\partial(\nabla u)}{\partial u}\Delta u + \frac{\partial S\left(u,\nabla u + R\right)}{\partial(\nabla u + R)}\frac{\partial R}{\partial u}\Delta u
\end{aligned}$$

Clearly, the derivatives depend on the specific choice of the numerical flux functions. The most complex part is the linearisation of the viscous fluxes and source terms, since we have to account for the gradient ∇u as well as for the gradient corrections R and r, respectively. All the differentiations needed, are carried out with the aid of the algebraic package Maple.

4.4. Solution of the linear system of equations

The iterative part of the Newton algorithm, see equation (4.7), is nothing else than the m-times solution of a linear system of the form:

$$A\,x - b = 0$$

The available memory of current (super)computers excludes the use of direct solvers for the problems considered and iterative solution methods must be used. In this work, we solve the linear system with the aid of two C++ libraries, namely the Iterative Template Library (ITL) in conjunction with the Matrix Template Library (MTL). The MTL is a high-performance generic component library that provides comprehensive linear algebra functionality for a wide variety of matrix formats. The ITL is also a generic component library that provides iterative methods for solving linear systems. For details the reader is referred to the home pages of the projects [2, 1]. We use a sparse matrix format of the MTL for the storage of our global Jacobian matrix $\frac{\partial W}{\partial U}$. The matrix is not symmetric definite and therefore algorithms for the solution of general matrices must be used. For this purpose ITL is equipped with the so-called Krylov subspace methods GMRES [86] and BICGSTAB [100], which are used in this thesis. In this work we normally use

GMRES for the implicit calculations. A comparison between GMRES and BICGSTAB will be given in the results chapter.

4.4.1. Krylov subspace iterative solvers

GMRES and BICGSTAB are iterative schemes, which per iteration produce a new approximate solution by adding search vectors (corrections) δ to the recent approximation x_{m-1}:

$$x_m = x_{m-1} + \delta$$

The search vectors δ of Krylov subspace techniques are restricted to the m-dimensional Krylov subspace K_m, which is spanned by products of polynomials of A and the residual vector of the last iteration $r_{m-1} = Ax_{m-1} - b$:

$$\delta \in K_m = \text{span}\{r_{m-1}, Ar_{m-1}, A^2 r_{m-1}, \ldots, A^{m-1} r_{m-1}\}$$

The dimension of K_m increases by one at each step of the approximation process. In order to obtain a complete scheme, m constraints must be imposed to extract a search direction δ. Primarily, the different schemes differ in the choice of these constraints, which are prescribed by orthogonality conditions. Hence, (bi)orthogonalisation of particular vectors are one essential part of Krylov subspace methods. Krylov subspace methods form a unifying framework for many of the well known methods for the solution of linear systems. As the main interest of this thesis is the application of the DG method to viscous flows, and not the development of iterative solution techniques for linear systems, further attention will be restricted to general information and the interested reader is refered to the standard references of Saad [86] and van der Vorst [100] for details.

4.4.1.1. Generalised Minimal Residual (GMRES) method

The Generalised Minimal Residual (GMRES) method was developed by Saad and Schulz [86]. The scheme can be derived by reformulating the system into an optimisation problem, where the solution x^* is obtained by minimisation of the normalised residual $r = Ax - b$.

$$x^* = \arg \min_{x \in R^n} ||r||_2.$$

GMRES can be applied to arbitrary regular matrices A and usually GMRES is the most robust of all Krylov methods, but it is also the most expensive in terms of memory, because all computed vectors in the orthogonal sequence have to be retained. In order to reduce storage requirements, a restarted version depicted as GMRES(m) is used in ITL, where after a user defined amount of m iterations the accumulated data is cleared and the intermediate results are used as initial data for the next m iterations. There are no definite rules governing the choice of m. Choosing when to restart is a matter of experience. Normally, we choose $m = 10$ as a rule of thumb, as obtained by several test runs.

4.4.1.2. BiConjugate Gradient Stabilised (BICGSTAB) method

The BiConjugate Gradient Stabilised method was developed by Van der Vorst to solve non-symmetric systems while avoiding the often irregular convergence patterns of the Conjugate Gradient Squared (CGS) method. In spite of these modifications the scheme can still break down, but the method is less memory consuming than GMRES and does not need to be restarted.

4.4.1.3. Preconditioning

The effectiveness of Krylov subspace methods like GMRES and BICGSTAB depends heavily on the preconditioning of the linear system [85]. The preconditioner M transforms the original system $Ax = b$ into a system

$$MAx = Mb$$

with lower condition number, which results in a reduction of the required number of iterations of the Krylov subspace method. ITL provides numerous preconditioners such as Incomplete Lower-Upper (ILU) decomposition, Symmetric Successive Overrelaxation (SSOR) or Jacobi preconditioners. Another interpretation of a preconditioner is, that M should be approximate A^{-1}as close as possible, while still being reasonably cheap to compute.

The simplest preconditioner is the Jacobi preconditioner. There one takes the diagonal part of the matrix A to form a preconditioner by

$$M_{diag} = D^{-1}$$

where

$$\begin{aligned} M_{ij} &= 1/A_{ij} \quad \text{for }, i = j \\ M_{ij} &= 0 \quad \text{, otherwise} \end{aligned}$$

This choice of M is also correspondingly refered to as diagonal preconditioner and it is possible to use it without any extra storage beyond that of the matrix A itself.

The SSOR preconditioner is motivated from the homonymous SSOR scheme, which can be consulted for the iterative solution of the original system. The matrix form reads

$$M = \omega(2-\omega)(D+\omega L)^{-1}D(D+\omega U)^{-1}$$

where ω is the extrapolation factor and L and U are the lower and upper triangular submatrices of A. The implementation in ITL chooses $\omega = 1$ for default, which reduces the SSOR preconditioner to a symmetric Gauss-Seidel preconditioner.

The ILU preconditioner relies on the incomplete lower-upper factorisation of the matrix A:

$$A = LU + F$$

Neglecting the remainder matrix F, one obtains a well invertible matrix $\tilde{A} = LU$ and the inverse $M = \tilde{A}^{-1} = U^{-1}L^{-1}$is the so-called ILU preconditioner. In fact, the ILU preconditioner provided by ITL is an ILU(0) preconditioner, which is constructed from the exact lower-upper factorisation formula of the matrix A, under the condition that the sparsity pattern A is maintained. Additional entries are suppressed.

Another important aspect of preconditioning is the ordering of the unknowns and the associated matrix structure, especially if unstructured grids are used. The effectiveness of ILU and SSOR preconditioner can be strongly influenced by the ordering of unknowns. An often applied method is for example the reverse Cuthill-McGee reordering algortihm to reduce the bandwith of A, which can be shown [85], comes along with improved effectiveness of the above mentioned ILU and SSOR preconditioners.

In this work, we normally use the ILU preconditioner, since it turned out to be most efficient. A comparison between ILU and Jacobi preconditioner will be given in the results chapter.

5. Parallelisation

As mentioned before, the compact form of DG discretisation makes it ideally suited for the implementation on parallel computers. In order to tap the full potential of DG, we included our parallelisation strategy into the program design process from the beginning on, which enormously reduces the later coding complexity. We use the Message Passing Interface (MPI) library for parallel communication, which guarantees maximal flexibility for parallel programming. The programming paradigm is targeted for machines with distributed memory, because currently everything points to, that the increase of computing power will more and more be achieved by interconnecting many processors and not due to the increase of the performance of individual processors. In the following the methodology—domain decomposition, data structures and communication—of the parallelisation is presented.

5.1. Domain decomposition

As the aimed computational platforms for our code are parallel machines with distributed memory, we carry out domain decomposition. This means that the grid is decomposed into subdomains (zones) and every processor is responsible for at least one subdomain. Clever splitting of the domain into a user specified number of zones is a complex issue and thus we are using the public domain software package METIS [57] for the domain decomposition. METIS is a set of programs for partitioning graphs, partitioning finite element meshes, and for producing fill reducing orderings for sparse matrices. The algorithms implemented in METIS are based on multilevel graph partitioning schemes. An example of a triangular grid, with approximately 15.000 cells, around a NACA0012 wing section, which is split into 32 zones, is shown in figure 5.1.

5.2. Data structures

The data structure for the parallelisation is organised with so called connect classes. We differentiate between local and remote connections, where the first stands for communication between zones which are processed on the same CPU, whereas the latter organises communication between zones on different CPU's. On every CPU, every zonal boundary has a local or a remote connect object, respectively. At the beginning of a parallel computation the multi-zone CGNS file is read in on every node/CPU and the zones are deployed on the different CPU's. This deployment is done in a way, that the total cell number per CPU is as equal as possible. This distribution is maintained over

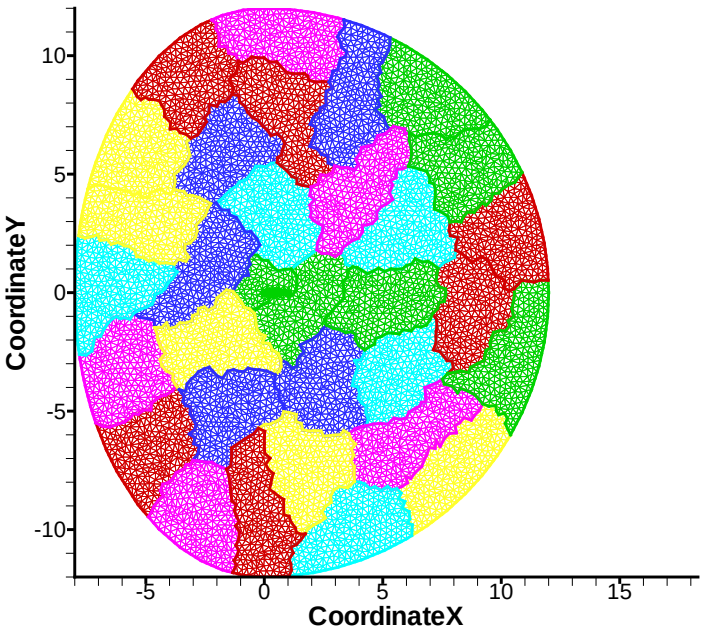


Figure 5.1.: Grid splitting by Metis for a NACA0012 airfoil mesh, 32 zones

the whole computation time. Thus we are working with static load balancing. Now, the respective connect objects can be built, which contain the vector of the local face numbers and the vector of the associated face numbers of the donor zone. Additionally, a connect object possesses a send and a receive buffer, which contain the data that is to be exchanged. The buffers are the consecutively linked boundary face degrees of freedom, which emanate from the cell degrees of freedom by simple projection to the boundary face. Hence, it is obvious, that the relative orientation of the corresponding faces has to be considered, for correct communication.

5.3. Communication

The communication in SUNWinT is also managed by the above mentioned connect objects, which provide the necessary send, receive and wait commands. The remote connect commands are essentially nothing else than the standard MPI commands. By contrast, local connect commands are simply local memory exchanges.

At the beginning of a time-step, the cell states are projected to the partition boundaries and afterwards that data is sent to the other processors, where data is stored in the respective buffers, prescribed above. Note, that only the trace of the solution has to be sent to the neighbouring partition, not the entire cell solution, whereby the communication bandwidth can be hold down especially for higher order calculations. When all zone local work is processed, it is guaranteed by synchronisation points, that the

complete data is received. Afterwards the zonal boundary fluxes are computed. Latency of the prescribed communication is thus hidden by the computation not only of the inner face fluxes as it should be done if a FV method is parallelised, but also by the computation of the cell integrals, which are independent of neighbouring elements. Thus, a maximised overlap of computation and communication is guaranteed. Note, that the partition boundary fluxes are evaluated twice, namely on both involved CPUs. An alternative would be the extra data exchange of the fluxes, which would be evaluated only on one CPU, but this would also increase the latency per time step. As computation power is cheap, and for large problems the evaluation of the lower-dimensional boundaries does not preponderate, we choose the method of doubly evaluated boundary fluxes. The schedule of a parallel time step of a single zone is summarised in algorithm 2.

Algorithm 2 Communication methodology in SUNWinT (a parallel time step of a single zone)

1. Project solution data of cells (adjacent to partition boundaries) to the boundary faces.
2. Send projected data to the neighboring partitions and receive projected data from the neighboring partitions.
3. While 2 is running, compute inner face fluxes and cell integrals.
4. Check if complete data is received—eventually wait.
5. Compute partition boundary fluxes.

The implicit time integration method is not parallelised up to now. Also grids, which contain curvilinear elements split over several zones can not yet be processed parallely.

6. Results

The result chapter is split into inviscid and viscous results. Although the main focus of this thesis is the treatment of viscous flow problems with the DG method, it is sensible to analyse the inviscid part of the scheme separately, since it is a fundamental ingredient of the overall Navier-Stokes and/or RANS scheme, and therefore will influence the viscous results into some extent. For many of the simulated test cases, we used quite coarse grids, if grid resolution requirements for second order CFD methods (FD and FV schemes) are taken into account. However, the use of such coarse grids is indispensable, if costly higher-order methods should have the potential to outperform classical second order methods in terms of efficiency.

6.1. Inviscid results

In this section the different Riemann problem solvers are assessed within the DG framework with the aid of test cases introduced in [94] by Toro. In order to validate the inviscid implementation of the code, convergence analysis on the basis of a Gaussian pulse in density is also performed. Finally, the implicit approach in time is analysed in detail and the high-order boundary treatment is tested by means of a NACA0012 airfoil.

6.1.1. Toro's test cases

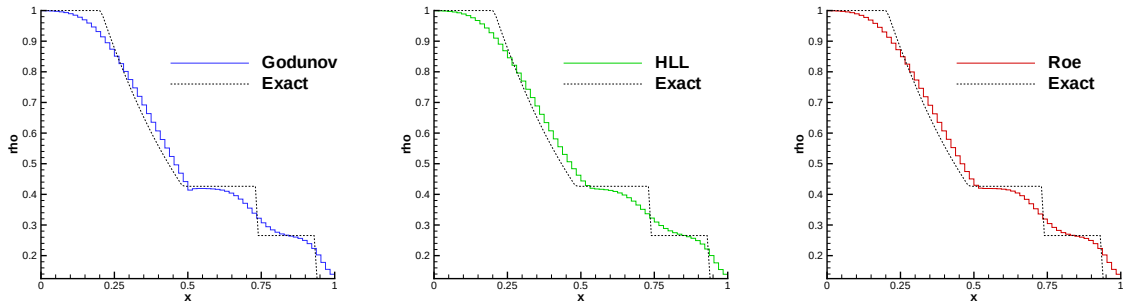
The test cases of Toro [94] are well suited and often used as first test cases for the validation of inviscid CFD codes, because exact solutions can easily be found for the respective problems. All cases are initial value problems, characterised by discontinuous (at $x = 0.5$) start conditions for density, velocity or pressure, see table 6.1.

Testcase	t_{end}	ρ_{l}	p_{l}	u_{l}	ρ_{r}	p_{r}	u_{r}
1	0.25	1.0	1.0	0.0	0.125	0.1	0.0
2	0.15	1.0	0.4	-2.0	1.0	0.4	2.0
3	0.012	1.0	1000.0	0.0	1.0	0.01	0.0
4	0.035	1.0	0.01	0.0	1.0	100.0	0.0
5	0.035	5.992242	460.894	19.5975	5.99242	46.0950	-6.19633

Table 6.1.: Test cases of Toro (left: $x \leq 0.5$, right: $x \geq 0.5$) [94]

The assessment of the influence of the Riemann solver (Godunov, HLL or Roe) is analysed in detail for the test cases number one and two. Test 1 can be regarded as

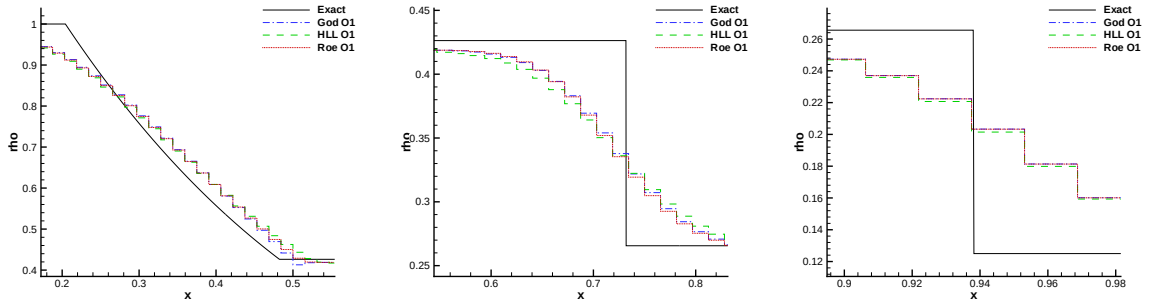
a quasi one-dimensional simulation of a shock wind tunnel test, where a diaphragm, which separates two different gas states, is abruptly destroyed and afterwards shock and expansion waves propagate into the tunnel.



(a) Godunov (b) HLL (c) Roe

Figure 6.1.: Toro's test case no. 1, Riemann solver comparison on 64 lines grid with P^0 line elements.

Results based on P^0-elements obtained with the different fluxes (Godunov, HLL and Roe) are compared with the exact solution, see figure 6.1. As can be seen in the comparison of the Riemann solvers (see figure 6.2), the differences are quite low.



(a) Rarefaction (b) Contact discontinuity (c) Shock

Figure 6.2.: Detailed comparison at rarefraction, contact discontinuity and shock area.

As reported in several publications [81, 49, 76], higher order ($O \geq 2$) solutions containing shocks tend to produce spurious oscillations and getting unstable. For test case 1, solutions based on P^0- up to P^3-elements obtained with the Godunov flux are plotted in figure 6.3. The visualisation is done with the true solution for P^0 and P^1 elements, whereas for higher-order elements an approximation is chosen. P^2 and P^3 elements are

split into two and three parts, respectively, wherein linear visualisation takes place. The solution quality is essentially improved by switching to high-order elements, but spurious oscillations are also introduced in the shock regions. However, a remarkable fact is, that without any inclusion of limitation or artificial damping the solution remains stable.

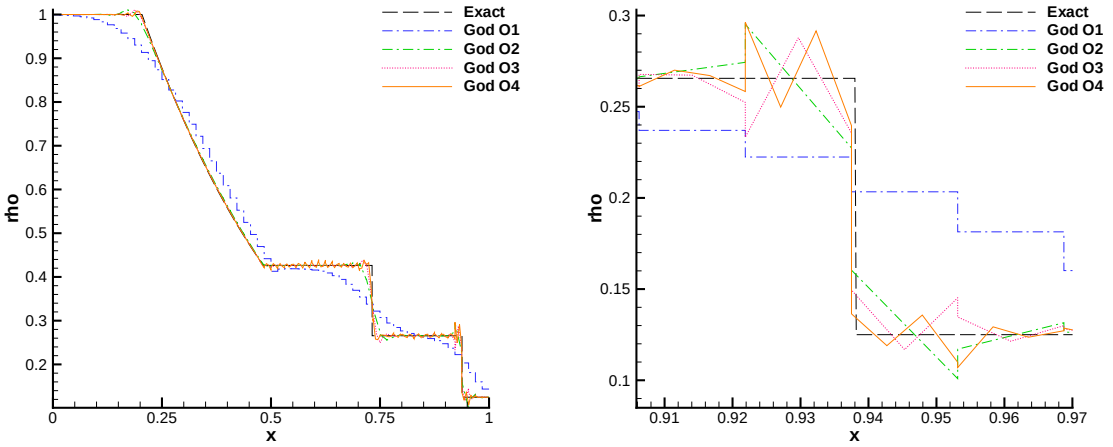


Figure 6.3.: Shock region solution, O1-O4, Godunov

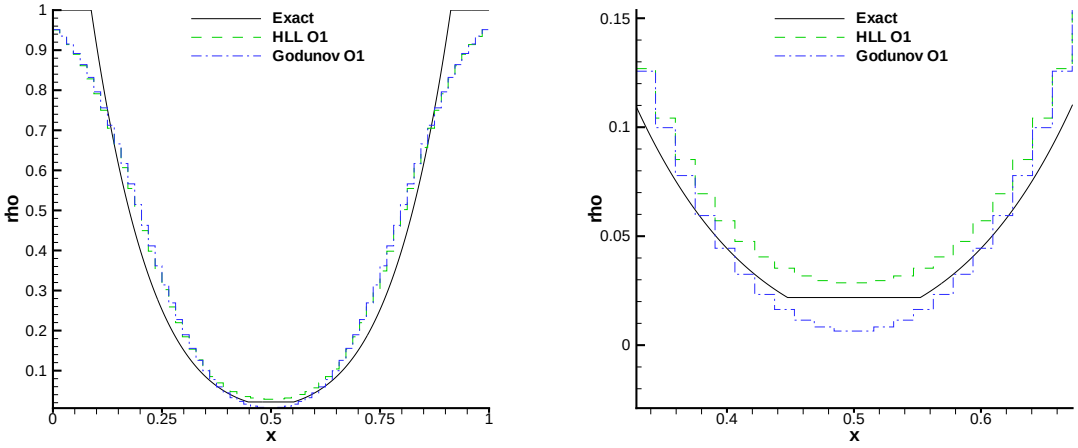


Figure 6.4.: Toro's test case no. 2, Riemann solver comparison on 64 lines grid with P^0 line elements.

The second test case (see table 6.1) is characterised by an abruptly diverging flowfield ($u_l = -2$ and $u_r = 2$ at $x = 0.5$). Consequently, a low pressure region develops between the diverging flow fronts, where (nearly) the state of vacuum is generated, see figure 6.4. Again, like in test case 1, the differences between results obtained with different fluxes (Godunov and HLL) are low. The only exception is the area of minimum pressure, where

the HLL flux overpredicts the density level, and whereas the Godunov flux produces too low density levels, see figure 6.4.

A solution with the Roe Riemann solver can not be obtained due to stability reasons. The problem with the standard Roe flux in this test case is well documented in [94] and could be cured with the inclusion of artificial dissipation, which prevents expansion shocks. Since the Roe flux is not chosen as the standard method for the viscous computations, we did not include this artificial dissipation correction.

However, higher order solutions based on the HLL or Godunov flux formulation cannot be obtained due to stability reasons. If a certain level of viscosity is added to the problem, the higher order solution can be stabilsed. This is associated with the artificial viscosity based stabilisation techniques used in [76, 49]. The effect of viscosity can be demonstrated for instance with the use of the LDG scheme. Here, the maximum "stabilising Reynolds number", we could choose for this test case is $Re = 200$. As can be seen in figure 6.5, the solution quality can be improved by increasing the polynomial ansatz order, if sufficient artificial viscosity is considered.

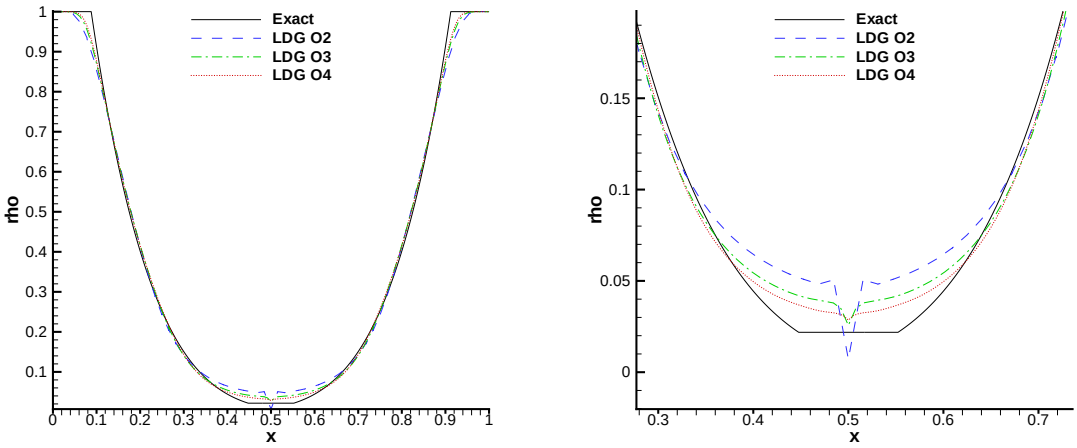


Figure 6.5.: Viscous computation with LDG scheme (Re=200)

The results of the remaining test cases (no. 3, 4 and 5) are shown in the appendix for the sake of completeness. To conclude, the influence of the used Riemann solver on the quality of the results is moderate, if low discretisation orders (O1 and O2) are used, and is vanishingly low, if higher order discretisation orders (O3 and O4) are used.

6.1.2. Gaussian pulse in density

An important step for the verification of the program code is, to check, whether the formal order of accuracy of the DG scheme is achieved in practice. The typical or standard method of choice is to analyse the order of grid convergence p, which is identical to the order of accuracy. Therefore solutions are computed on grids, which systematically differ in grid density or the number of cells, respectively. The formalism details of the convergence studies are explained in appendix D.

We consider a multidimensional Gaussian density fluctuation, while pressure p_0 and velocity u_0 remain constant. The initial distribution in conservative variables resolves to

$$U\left(\rho_o,\, u_0,\, p_0,\, \vec{x}_0,\, \vec{x}\right) = \begin{pmatrix} \rho_0 + a\exp\left[-\ln\left(2\frac{(\vec{x}-\vec{x}_0)^2}{\sigma^2}\right)\right] \\ \rho u_0 \\ 0 \\ 0 \\ \frac{p_0}{\gamma-1} + \frac{(\rho u_0)^2}{2\rho} \end{pmatrix} \tag{6.1}$$

where $\vec{x}_0$ stands for the peak position, σ is the broadness and a the height of the pulse. The constants are prescribed as

$$\begin{aligned} \rho_0 = 1, \qquad & u_0 = 0.25, \qquad & p_0 = 1,\, a = 0.1 \\ x_0 = 0.25\ (1D)\,, \quad & \vec{x}_0 = (0.25,\, 0.25)^T\ (2D) \quad & ,\sigma = 0.04 \end{aligned}$$

The domain we used are $[0, 1]$ in 1D and $[0, 1] \times [0, 0.5]$ for the 2D studies. The two-dimensional grids we used are shown in figure 6.6. As can be seen, for the two-dimensional analysis, we differ between triangular and quadrilateral grids.

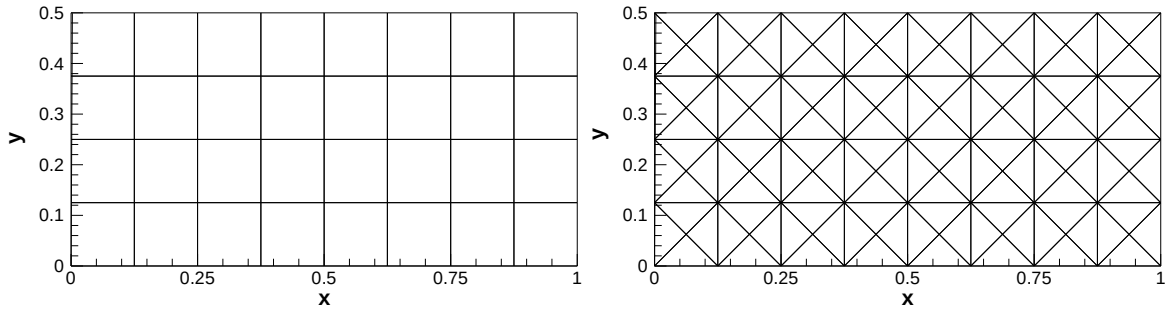


Figure 6.6.: Different grid types for the convergency analysis

The exact solution of the initial value problem (6.1) is trivial and can be obtained by moving the Gaussian distribution with the prescribed convective flow speed u_0. No secondary flow effects occur, since we prescribe constant pressure and heat fluxes are neglected in the present case of the Euler equations.

Since the propagation of the Gaussian pressure distribution is an unsteady process,

we have to choose a temporal and spatial discretisation, which possess the same formal order of accuracy, in order to guarantee an overall accuracy of p. We could also have used a lower order scheme in time, independent of the spatial discretisation order. However, then we must have chosen suitably low time steps, in order not to mask the spatial error by the temporal error. Hence, we combined the identical order in space and time by using appropriate $m-$stage Runge-Kutta time integration methods (see section 4.1) with the respective P^n-elements. The explicit time step is set to 80% of the stability limit in all computations.

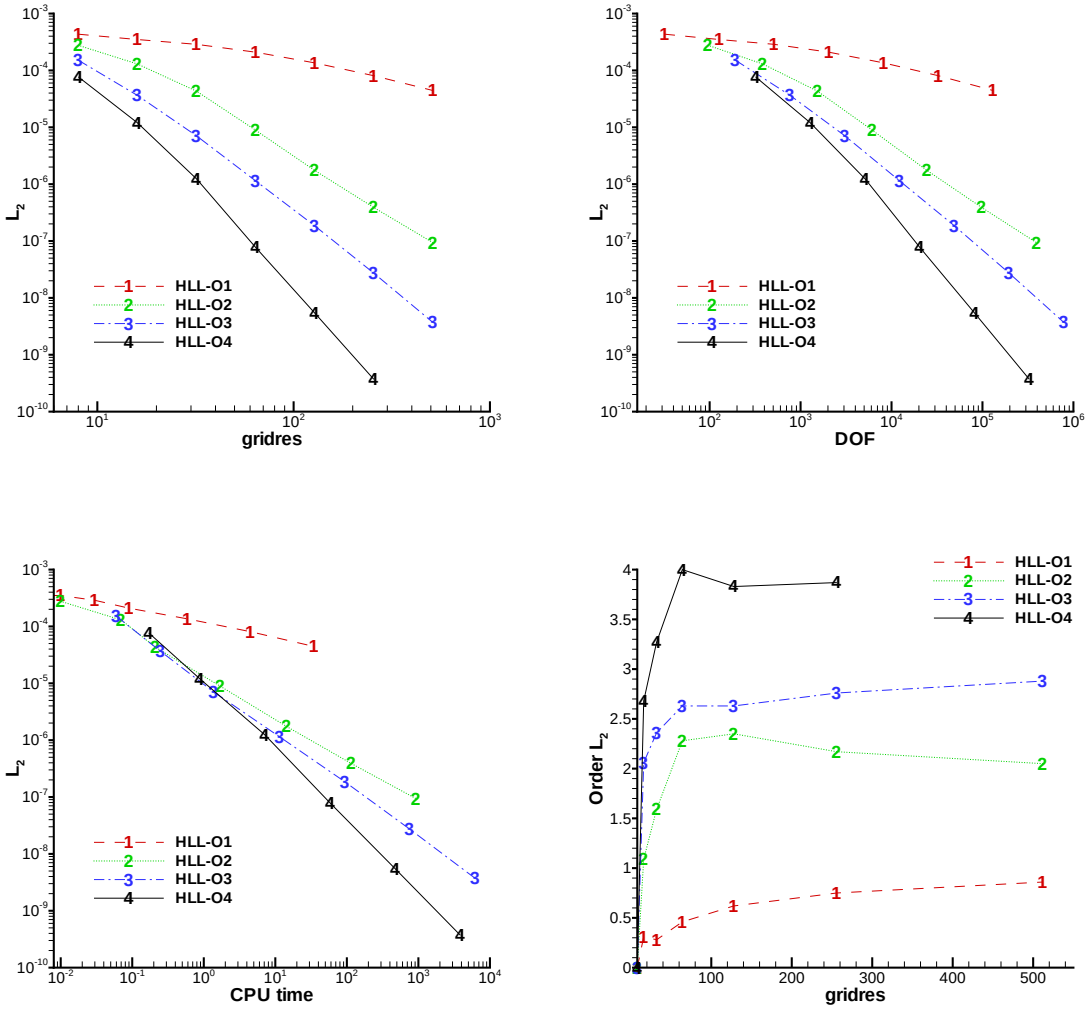


Figure 6.7.: Convergence rates for quadrilaterals, HLL flux

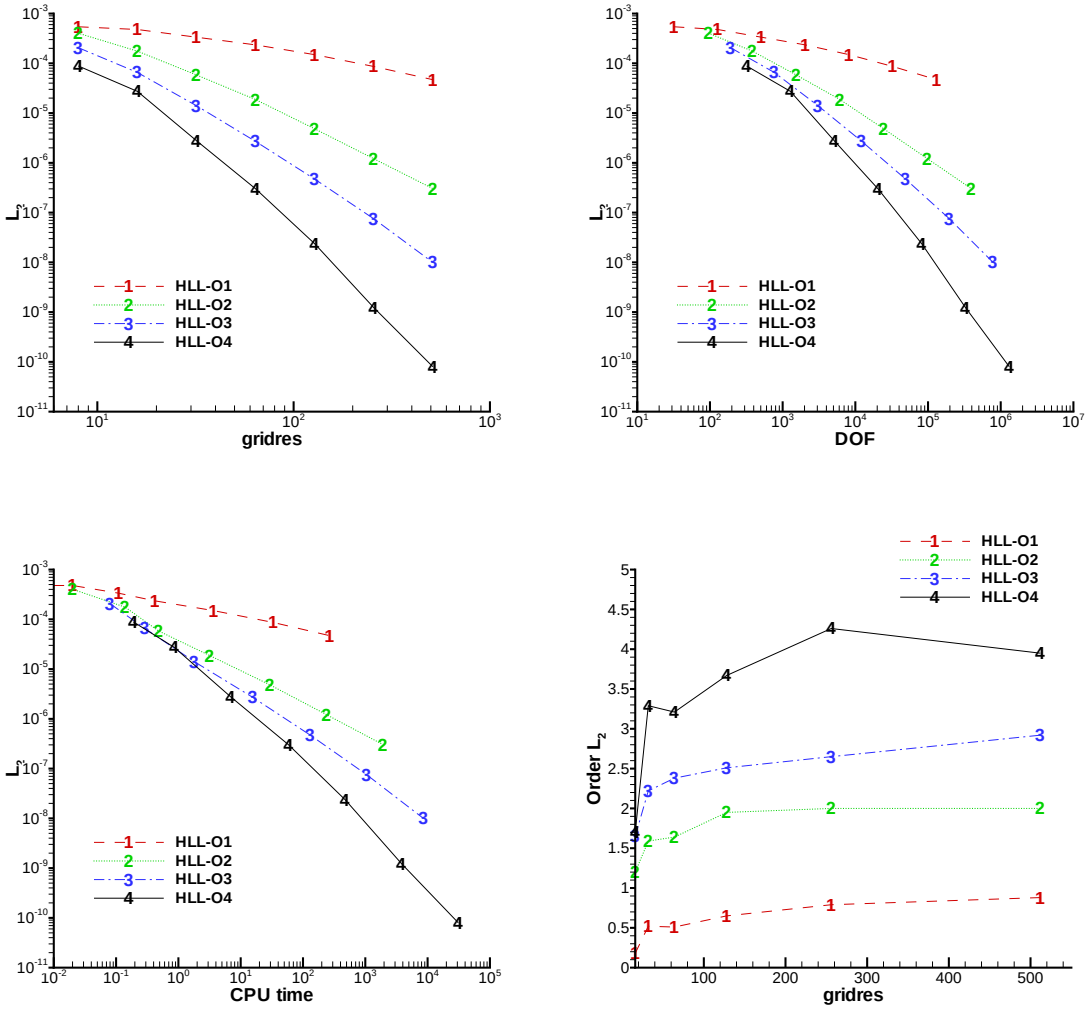


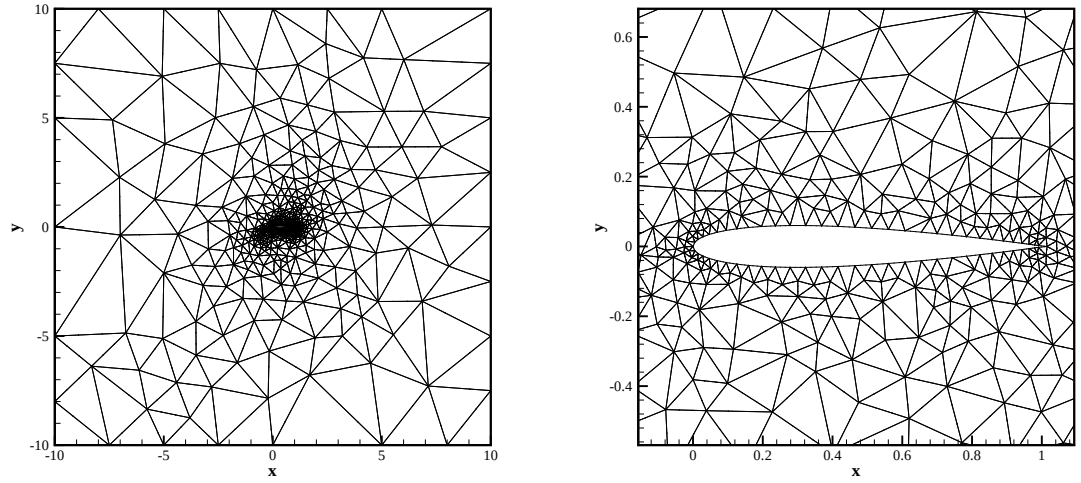
Figure 6.8.: Convergence rates for triangles, HLL flux

In figures 6.7 and 6.8 the L_2-error of the method is plotted against the grid density, the degrees of freedom and the CPU time. Furthermore, the formal order of accuracy is shown as function of the grid density. It can be seen clearly, that the method achieves optimal order of convergence $\mathcal{O}\left(h^{p+1}\right)$ on triangular and quadrilateral grids. If we take the memory requirements (DOF) or the CPU time as measure of the effort, we see that the higher order schemes perform better than the lower order ones, if a certain desired error level is undershot.

Additional results, like convergency tables and further error norms, for the convergence studies in this work are summarised in appendix D.

6.1.3. NACA0012

Next, the functionality of the high-order boundary implementation will be assessed. Therefore the flow around a NACA0012 airfoil is computed ($\mathrm{Ma}_\infty = 0.63$, $\alpha = 2^0$). In figure 6.9, the unstructured triangular grid, which is used for both, the straight boundary and the high-order boundary based computation, is shown. All results, discussed in the following analysis, are obtained with fourth order accurate P^3-elements.



(a) Full grid

(b) Airfoil near grid

Figure 6.9.: NACA0012 grid with approx. 1100 cells, where 62 cells are distributed along the airfoil boundary

6.1.3.1. Straight boundary

As can be seen in the contour plots of pressure, see figure 6.10, the straight boundary dicretisation produces wiggles near the airfoil boundary. This is obvious, since due to the high-order discretisation, every kink in the polygonial airfoil contour is seen by the flow. This leads to the strong oscillations near the more curved leading edge contour of the airfoil and the, compared to that, smooth distribution at the low curved areas of the airfoil.

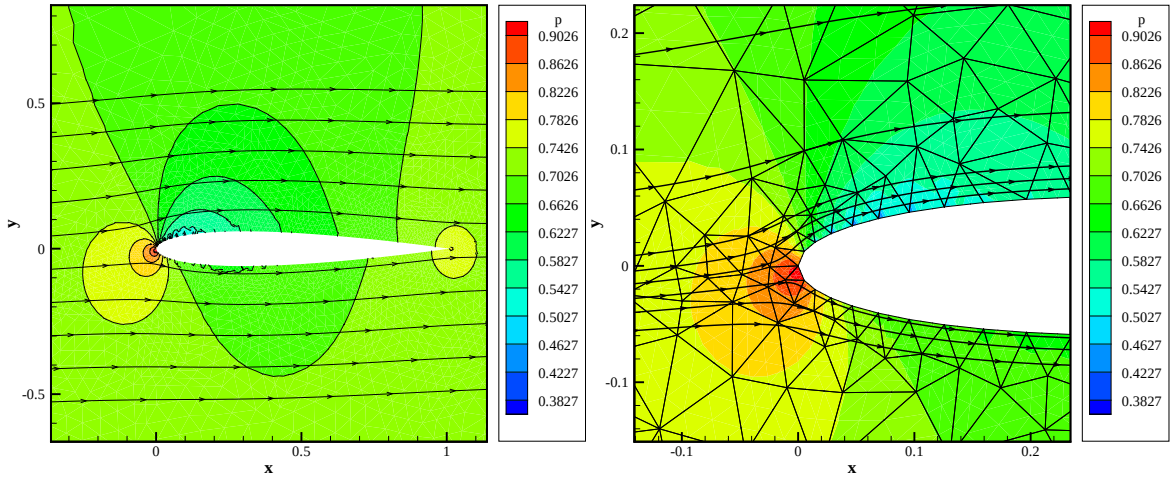
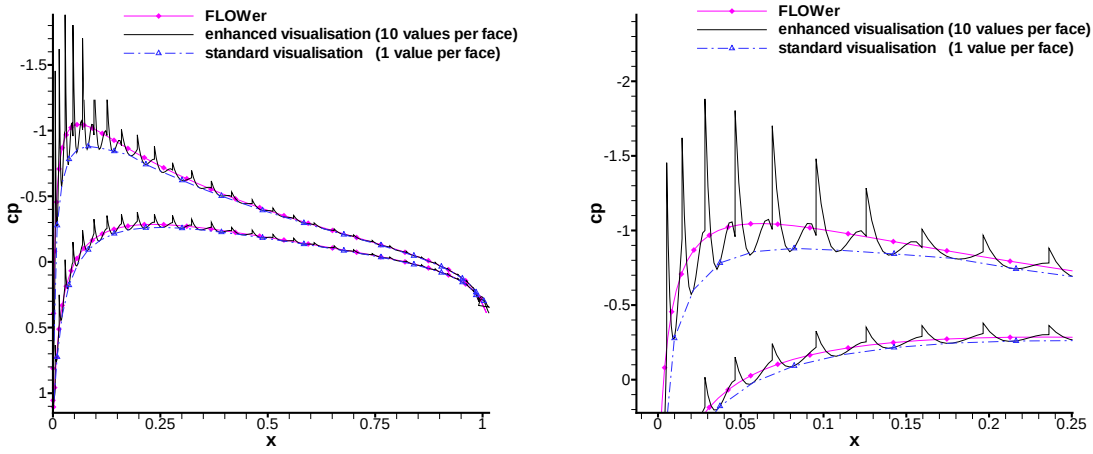


Figure 6.10.: Pressure contours and streamlines obtained with straight P^3-triangles for inviscid flow around the NACA0012 airfoil,$\mathrm{Ma}_\infty = 0.63$, $\alpha = 2^0$.

In figure 6.11 these oscillations can also be observed in the surface pressure coefficients. It turned out that the standard visualisation (one value at the face center) is not sufficient, because oscillations are filtered out and the distribution seems to be smooth. Therefore we enhanced the visualisation by further face points, dependent on the length of the face. The result (here with 10 intermediate points per face) is shown in figure 6.11 as well. In order to assess the results obtained with our DG code, we have done a simulation with the DLR FLOWer code [62] on a fine structured quadrilateral C- mesh (180×50). These reference results are also shown in figure 6.11.



(a) 1 value in the face center

(b) 10 intermediate values per face

Figure 6.11.: Surface pressure distribution with straight P^3-triangles

6.1.3.2. High-order boundary

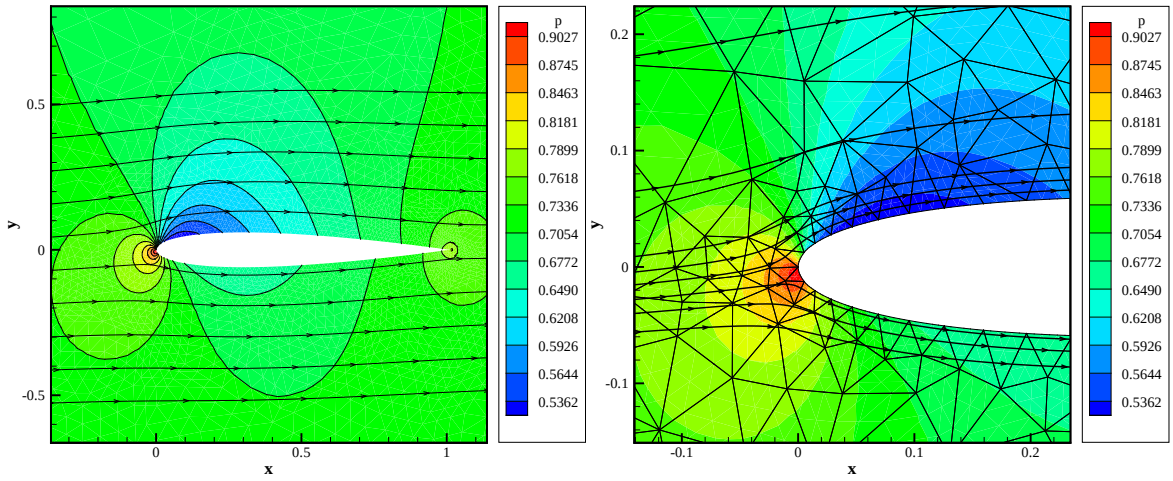
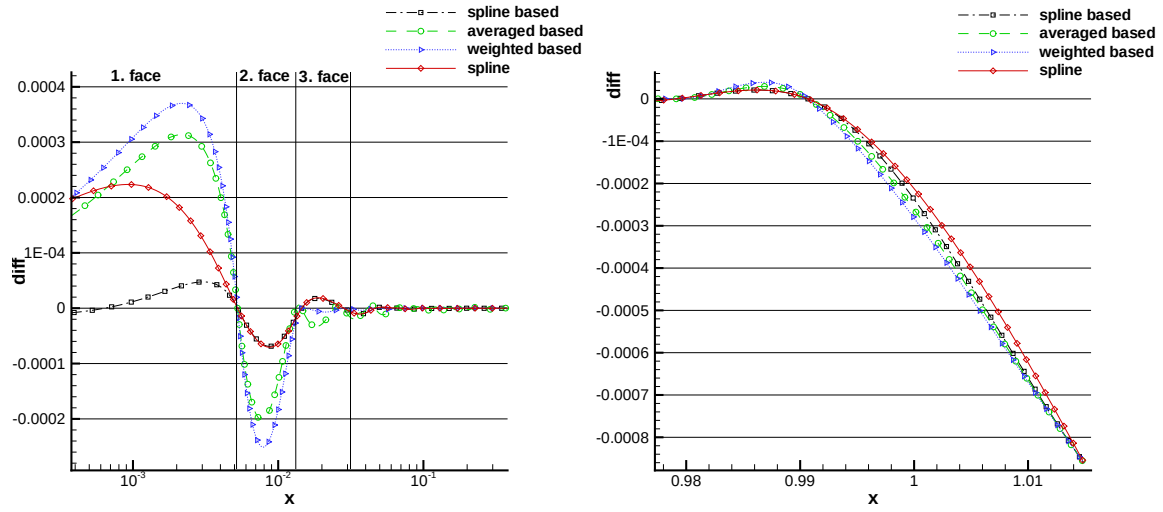


Figure 6.12.: Pressure contours with high-order P^3-triangles for inviscid flow around the NACA0012 airfoil,Ma $= 0.63$, $\alpha = 2^0$.

Since the exact NACA0012 geometry is described by a fifth order polynomial (see appendix E.1), we cannot represent it exactly with the chosen piecewise cubic interpolation method. Figure 6.13 shows the deviations of the averaged and weighted normal vector based interpolation to the real NACA geometry. For comparison, we also plot the error of the spline normal vector based interpolation based and the spline.



(a) Leading edge area

(b) Trailing edge area

Figure 6.13.: Deviations of the piecewise cubic approximations to the exact NACA0012 airfoil

It is is obvious that the biggest deviations between the exact and the approximated geometries as well as the biggest differences between the interpolation methods are located at the leading edge. The best approximation (in the first two cells) is obtained with the interpolation based on the normal vectors of the splined airfoil. The third and fourth boundary face are slightly better approximated by the interpolation based on the weighted normal vectors.

Next, we want to analyse the different interpolation methods by comparing their first and second derivatives, namely

$$\begin{aligned} \frac{\partial y}{\partial x} &= \frac{\partial y/\partial t}{\partial x/\partial t} \\ \frac{\partial^2 y}{\partial x^2} &= \frac{\frac{\partial^2 y}{\partial t^2} - \frac{\partial y/\partial t}{\partial^2 x/\partial t^2}}{\left(\frac{\partial x}{\partial t}\right)^2} \end{aligned}$$

Unfortunately, the compact interpolation methods (averaged or weighted normals) generate discontinuous second order derivatives at the leading edge area. The interpolation based on the normal vectors of the splined geometry also experiences discontinuities in that area, but the height of the jumps is fundamentally reduced, see figure 6.14.

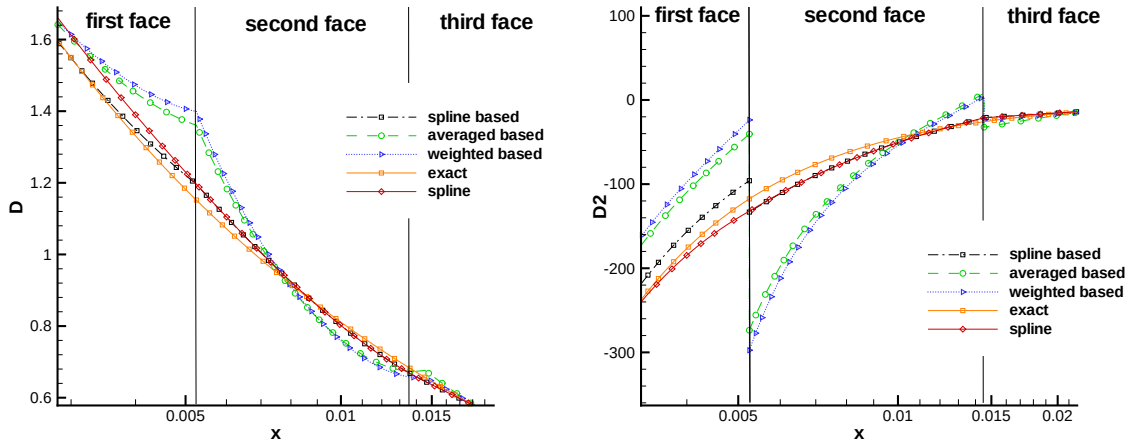


Figure 6.14.: Comparison of first and second derivatives for the different geometry interpolation methods

These deviations can be explained due to the fact, that the piece-wise cubic representations are constructed in the respective edge tangential system. In other words, we use only one cubic polynomial in the tangential system, whereas we use two independent splines (one in x and one in y-direction) for the spline interpolation. Hence, the spline itself is continuous in the first and second derivative with respect to t, but not mandatory in x.

The jumps in the second order derivatives are directly reflected in the surface pressure distribution of the airfoil, see figure 6.15. Consequently, using the spline normal vector interpolation method, the resulting pressure distribution is much smoother.

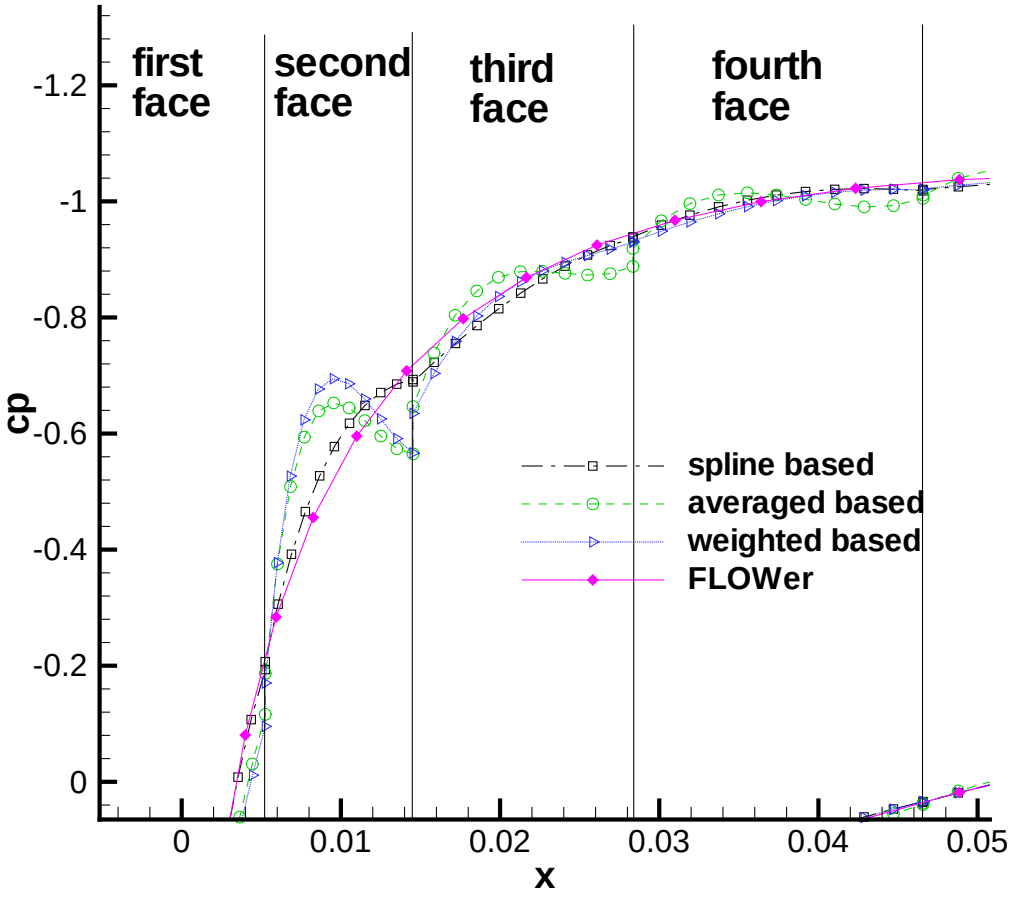


Figure 6.15.: Surface pressure distribution with high-order P^3-triangles for the different geometry interpolation methods

To conclude, the compact interpolation method seems to be sufficient for standard airfoils, like NACA0012, but further tests have to be run for viscous flow cases, see next chapter.

Finally, we want to compare global coefficients, see table 6.2. The coefficients taken from the FLOWer results should be viewed as reference. The computations based on the high order curvilinear boundary treatment deliver better results than that computed with straight boundaries. As expected, the differences between the different interpolation techniques are low.

	Elements	c_L	c_D
FLOWer	180×50	0.3201	$0.9135 \cdot 10^{-3}$
straight boundaries	1100	0.3170	$1.5487 \cdot 10^{-3}$
averaged based	1100	0.3190	$1.1544 \cdot 10^{-3}$
weighted based	1100	0.3189	$2.6947 \cdot 10^{-5}$
spline based	1100	0.3187	$1.2043 \cdot 10^{-4}$

Table 6.2.: Lift and drag coefficient for NACA0012 airfoil for straight and high-order boundary treatment with P^3-triangles.

6.1.4. Comparison of Krylov Subspace techniques

The application of Newton Krylov subspace techniques for CFD based on the FV technique is widely spread in the CFD solver community, see for example [20]. In this section, we want to compare the Krylov subspace techniques, namely the BICGSTAB and the GMRES methods in the framework of fully implicit DG discretisations. Another aspect to be shortly discussed is the influence of the used preconditioner. The test case we used for that, is again the subsonic flow around the NACA0012 airfoil, also described in section 6.1.3. Here we used a structured 32×16 C-mesh, see figure 6.16. The airfoil boundary is treated with the curved high-order approach, based on the weighted normal vectors, described in the previous section.

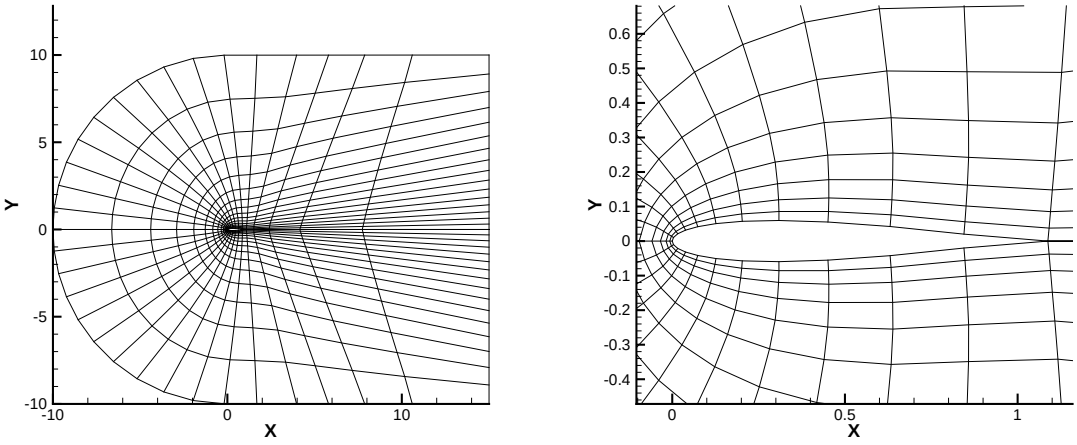
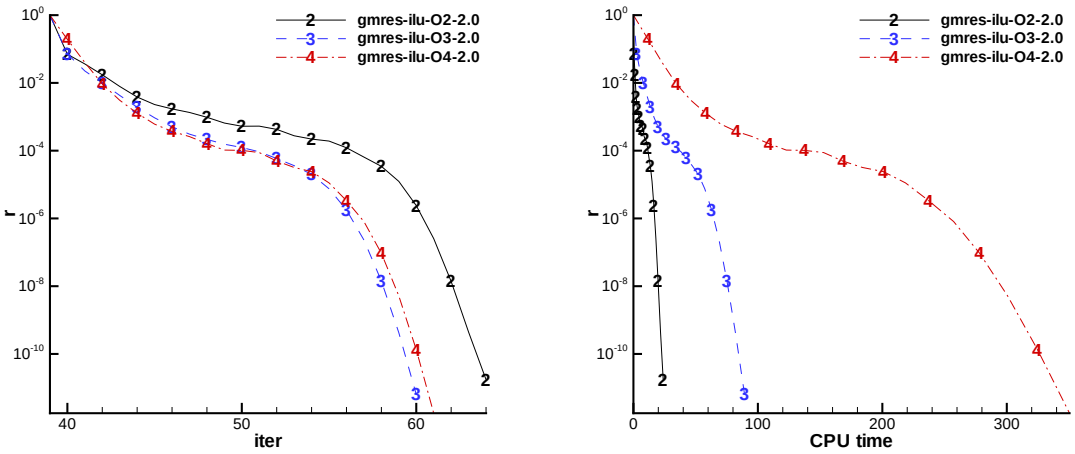


Figure 6.16.: Structured 32×16 C-mesh

The $\mathcal{O}2$-, $\mathcal{O}3$- and $\mathcal{O}4$- computations are initialised with the fully converged $\mathcal{O}1$-solution, which takes only very few seconds in CPU time. All results are obtained with one Newton iteration per timestep. All GMRES based calculations are restarted after 10 iterations in the GMRES loop. In figure 6.17 the history of the L^2-norm of the density residual (all degrees of freedom) is plotted against the number of time steps



(a) as a function of time steps

(b) as a function of CPU time

Figure 6.17.: Residual history for GMRES method (restart after 10 iterations) for different spatial orders ($\beta = 2.0$)

(iterations) 6.17(a) as well as against the CPU time 6.17(b). As can be seen, for this test case, the third order version is the most efficient in terms of timesteps/iterations. However, in terms of CPU time needed to reduce the residual, the expected behavior can be observed. We chose a similar law for the calculation of the implicit timesteps as in [82], where the timestep is increased reversed proportional to the reduction of the residual of two sequential iterations

$$\Delta t^{n+1} = max\left[\Delta t^n \left(\frac{r^n}{r^{n-1}}\right), \Delta t^n \cdot \beta\right]$$

For reasons of robustness, we added a limitation factor β for the increase. For all the calculations in this section we restarted all computations with a CFL number of one, and the maximum allowed increase of preceding timesteps is 100 percent ($\beta = 2.0$). The history of the timestep (expressed in terms of CFL multiples) is shown in figure 6.18(a). It can be seen, that we reach CFL numbers of several hundred and thousand after only very few iterations. In order to better assess the residual histories of figure 6.17, we added the history of the lift coefficient in figure 6.18(b). All calculations ($\mathcal{O}2$-$\mathcal{O}4$) are well converged in terms of the lift coefficient.

In figure 6.19 the performance of the GMRES and BICGSTAB method is compared. The differences in terms of CPU time are relatively small. The second, third and fourth order computations based on the BICGSTAB method are 12, 4 and 1 percent faster than the computations based on the GMRES method, see also table 6.3. For comparison the CPU times for the explicit (first order in time) computations with fixed CFL number are added to the table. As expected for steady calculations, the explicit approach is much

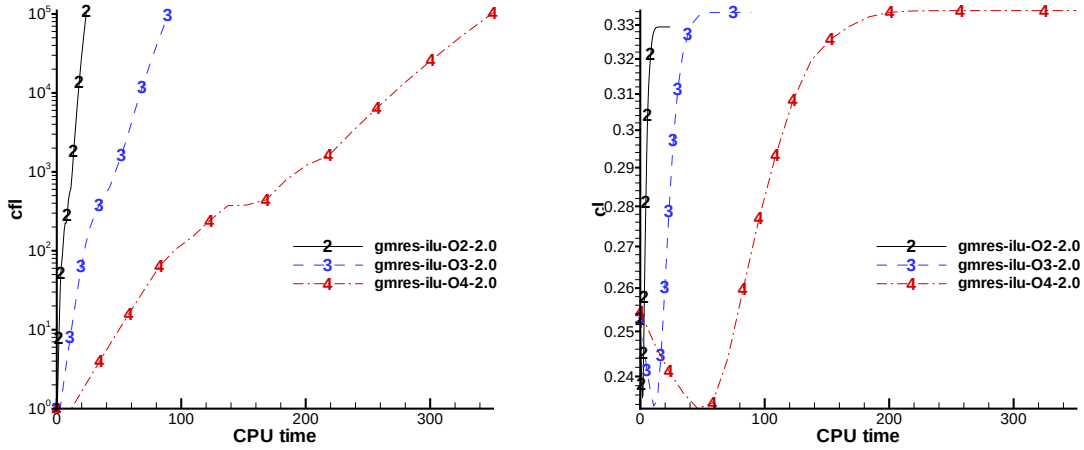


Figure 6.18.: Lift coefficient and timestep history (GMRES)

Solver / Order	2	3	4
GMRES CPU time	26.4 s	97.6 s	378.0 s
Iterations	2209	1739	1999
BICGSTAB CPU time	23.5 s	93.7 s	374.1 s
Iterations	1938	1618	1989
explicit CPU time	1377.3 s	>>7200 s	>>21600 s
CFL number	0.4	0.15	0.08

Table 6.3.: Comparison of CPU time needed for explicit and implicit time-stepping approaches

more expensive than the implicit one. Therefore, the third and fourth order solutions are not computed to the same level of accuracy and stopped after several hours of computation time.

The influence of the different preconditioners on the required Krylov iterations is visualised in figure 6.20. The Jacobi preconditioner turns out to be less efficient than the ILU one for all spatial calculation orders. In case of the ILU preconditioner, the timestep can be strongly increased. In contrast to that, there seems to be a barrier for the timestep, if the Jacobi preconditioner is used. By doing several test runs, we identified maximum possible CFL numbers of 80, 30 and 20 for the $\mathcal{O}2$, $\mathcal{O}3$ and $\mathcal{O}4$ calculations, see figure 6.20. Despite of the Jacobi preconditioner being much more efficient in terms of CPU time per iteration, the decrease of the GMRES loop residual requires a lot more iterations, see figure 6.21 .

Finally, we observed in several tests (not shown here), that the ordering of cells can strongly influence the performance of the ILU preconditioner, whereas the Jacobi pre-

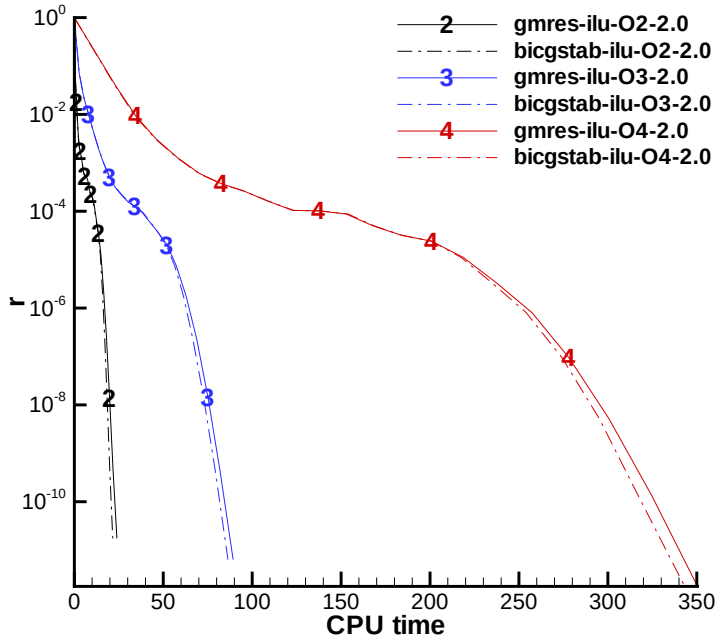


Figure 6.19.: Comparison of Newton-loop residual history for BICGSTAB and GMRES

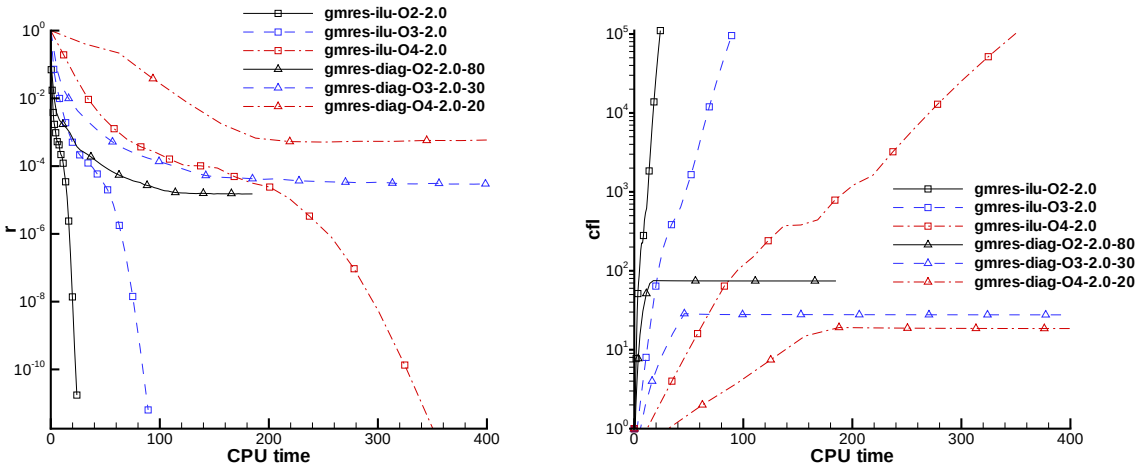


Figure 6.20.: Residual and timestep history with Jacobi(diag) and ILU preconditioners

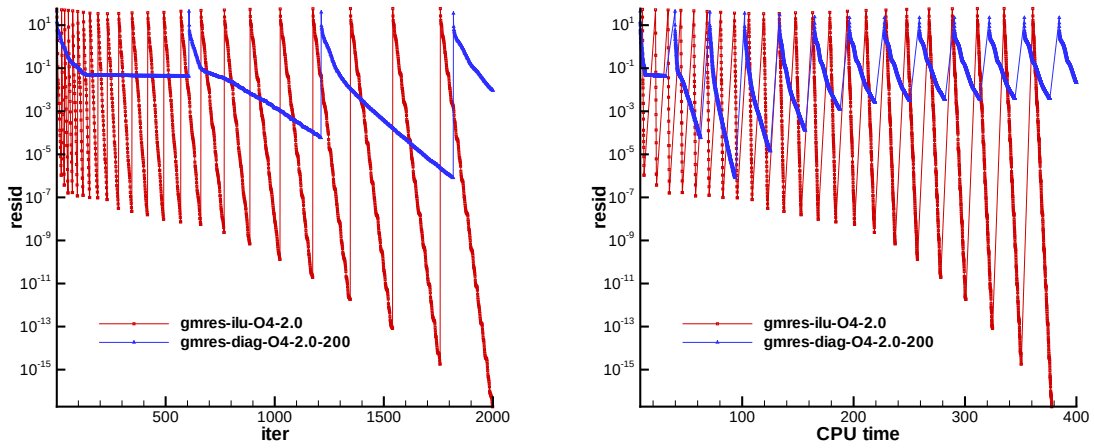


Figure 6.21.: GMRES loop residual history in terms of iterations and CPU time

conditioner is independent of the ordering of the unknowns as expected. To conclude, the combination between GMRES or BICGSTAB method and ILU preconditioner delivers very robust and also very efficient results in terms of CPU time for our current implicit time-stepping algorithm.

As mentioned above, all results are obtained with one Newton iteration per timestep. Tests with more than one (Newton) iteration turned out to be less efficient. The reason for that is the very time-consuming analytical construction of the Jacobian. If a matrix-free solver is used, several Newton loops can make sense, since the cost of the construction of the Jacobian can be strongly reduced [82].

6.2. Viscous results

In this chapter viscous results are presented for several standard test cases. The chapter is split into the laminar and the turbulent computations. First of all, like for the inviscid part, a validation of the achieved convergence rates is done. After that, the unsteady performance of the code is tested by the simulation of the laminar flow around a circular cylinder. Next, the quality of boundary layer predictions of our DG implementation is analysed with the aid of a flat plate flow. The last part of laminar results consists of a classical laminar test case, a low Reynolds number flow around the NACA0012 airfoil.

The turbulent results part is organised in a similar fashion to the laminar part. First the prediction of partially laminar and turbulent boundary layers is assessed by simulating a flat plate flow. After that, the curved quadrilateral layer method is tested in conjunction with the flow field around the well known Aerospatiale (Onera-A) airfoil.

6.2.1. Laminar results

6.2.1.1. Convergence study for the Navier-Stokes equations

As shown for the inviscid results, we here want to demonstrate the high-order behaviour of the LDG and BR2 scheme for the case of viscous calculations. The test case chosen therefore is the Poiseuille flow, which is nothing else than a pressure gradient driven tube flow. An analytical solution for that problem can be derived, if incompressible flow (and constant viscosity) is assumed. The velocity distribution is parabolic with respect to the cross section of the tube and constant along the tube axis for mass conservation reasons. The pressure decreases linearly along the tube axis and is constant across its section.

The (incompressible) solution in conservative variables reads

$$U_{\text{exact}}^{\text{inc}} = \begin{pmatrix} \rho_0 \\ \frac{1}{2\mu_0}\left(\frac{dp}{dx}\right) y\,(y-b) \\ 0 \\ \frac{\rho_0}{\gamma-1}\left(p_0 + \left(\frac{dp}{dx}\right) x\right) + \frac{\rho}{2}u^2 \end{pmatrix} \tag{6.2}$$

Since the solution scheme, that we want to analyse, is based on the compressible Navier-Stokes equations, we have to slightly modify these equations, by introducing a source term S on the right hand side of equation 2.4, in order to get the incompressible solution with the compressible equations. The source term is obtained as the remainder, if the compressible NS equations are evaluated using the incompressible solution $U_{\text{exact}}^{\text{inc}}$. The derivation of S results in

$$S = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{2\mu}\left(\frac{dp}{dx}\right)^2 \left[\frac{1}{\kappa-1} y\,(y-b) - \frac{1}{2}\,(2y-b)^2\right] \end{pmatrix}.$$

With this modification, we can assess our implementation by comparing the numerical results with the incompressible solution $U_{\text{exact}}^{\text{inc}}$. We also want to point out here, that in spite of this test case being laminar, we can also analyse the behavior of the RANS scheme to some extent, because the only fundamental difference between our RANS and NS implementation is the evaluation of the turbulence model source term S, which is in fact included in this test case. A very similar test has been carried out by Oliver [75], where a slightly different source term is used. On the inflow and outflow boundary the exact solution is used as boundary state, the upper and lower wall boundary is treated as adiabatic.

In figure 6.22 the L_2-error of the LDG method and the BR2 method is plotted against the grid density. As can be seen, both schemes achieve optimal order of convergence, whereas the LDG scheme seems to produce lower error levels than the BR2 method. For comparison, slopes of reference are also added in the diagrams.

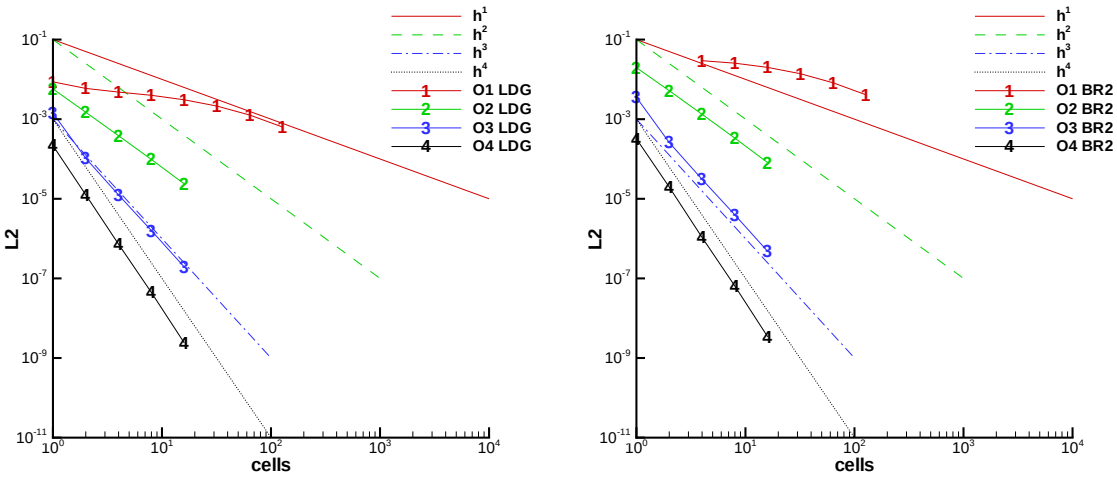
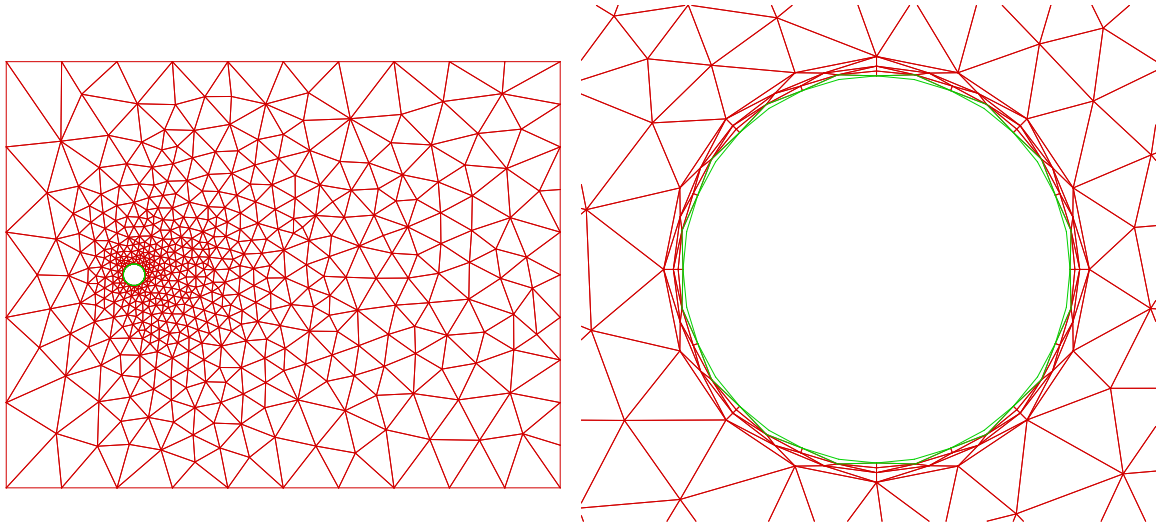


Figure 6.22.: Rate of convergence of LDG and BR2 scheme ($\mathrm{Ma}_\infty = 0.5$, $\mathrm{Re}_\infty = 10000$)

6.2.1.2. Circular cylinder

We calculated the laminar flow over a circular cylinder ($Ma_\infty = 0.1$, $Re_\infty = 150$). This unsteady flow case is often calculated [55] as well as measured [108] in the literature and therefore excellently suited for comparisons. Our unstructured computational grid (see figure 6.23) is extremely coarse ($\approx$ 1200 triangles) compared to that used in [55] (871x503 O-Grid). We utilise curved boundary triangles, see figure 6.23(b), in order to represent the real geometry of the cylinder and to preserve the formal order of accuracy, respectively.

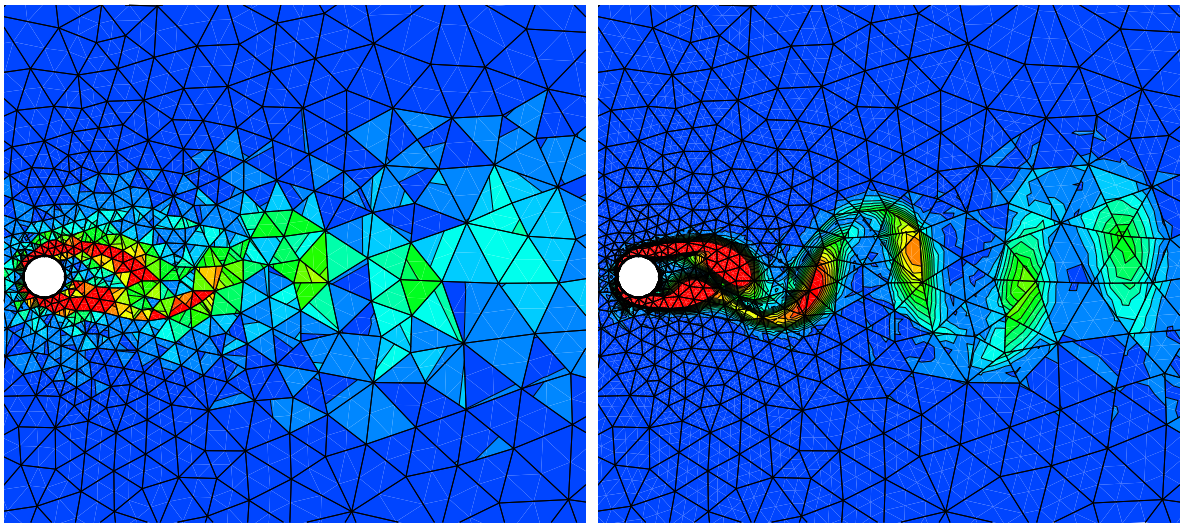


(a) Complete grid ($\approx$ 1300 triangles)

(b) Near cylinder grid with curved boundary triangles (green)

Figure 6.23.: Unstructured cylinder grid

Figure 6.24 shows a qualitative comparison of the computed instantaneous flowfield (von Karman vortex street) between a second and a fourth order spatial discretisation. The comparison demonstrates the lower dissipative behavior of the higher order discretisation, which conserves the shed vortices over a considerably longer distance. All cylinder flow simulations are performed with the classical explicit fourth order Runge-Kutta scheme and the LDG (BR1) method.



(a) Second order (P1 elements)

(b) Fourth order (P3 elements)

Figure 6.24.: Von Karman vortex street vorticity contours ($Ma_\infty = 0.1$, $Re_\infty = 150$)

In table 6.4, time variations of lift and drag coefficient are documented with mean values, amplitudes as well as the period of vortex shedding (Strouhal number) for several discretisation orders (O2-O4). For comparison, the results of the DNS with a sixth order finite difference scheme [55] are also included as a reference. As the coarse grid suggests, the second order DG method is not capable of computing the flowfield accurately. Especially the amplitudes of drag and lift are strongly underpredicted. Switching to the third order scheme drastically improves the situation, but best results in all categories are obtained with the fourth order scheme. There are still deviations to the reference, but this could be addressed to the really coarse computational grid used.

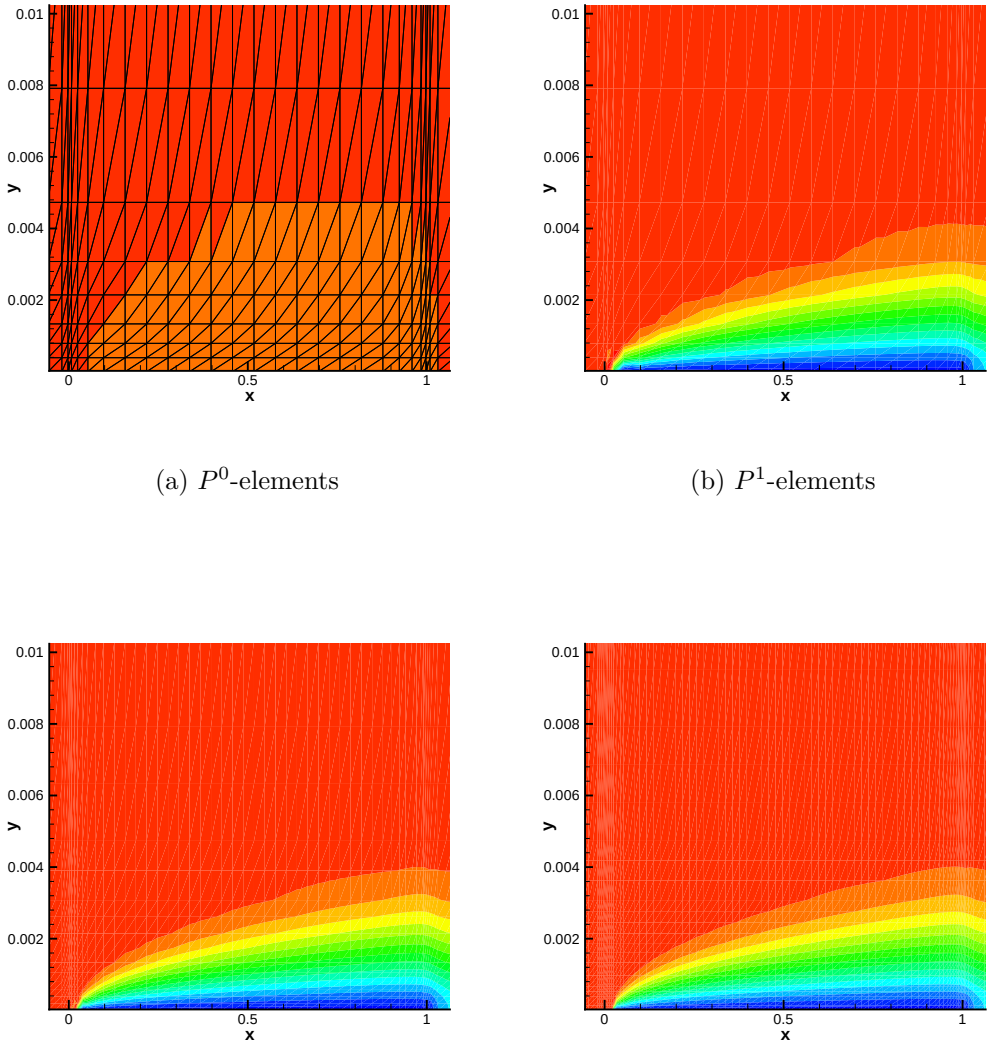
Element type	Spatial order	Strouhal	mean drag	drag ampl.	lift ampl.
Linear	2	0.168	1.383	0.0071	0.242
Quadratic	3	0.190	1.432	0.0308	0.594
Cubic	4	0.188	1.380	0.0270	0.578
Finite Differ. [55]	6	0.183	1.320	0.0260	0.520

Table 6.4.: Comparison between calculated results (O2-O4)

All further flow cases are computed implicitly in time with the first order Euler backward method and the BR2 scheme.

6.2.1.3. Flat plate

For detailed validation of the laminar implementation we calculated the flow over a flat plate ($\mathrm{Ma}_\infty = 0.3, \mathrm{Re}_\infty = 1 \times 10^6$) and compared the global force coefficients as well as local profiles between several spatial discretisation orders. The computations are performed on a very coarse triangularly split structured H-grid (49×22 with 20 cells along the plate).



(a) P^0-elements

(b) P^1-elements

(c) P^2-elements

(d) P^3-elements

Figure 6.25.: Grid and Mach number contours (y direction stretched by factor 100)

The computations are done implicitly in time with the first order Euler backward

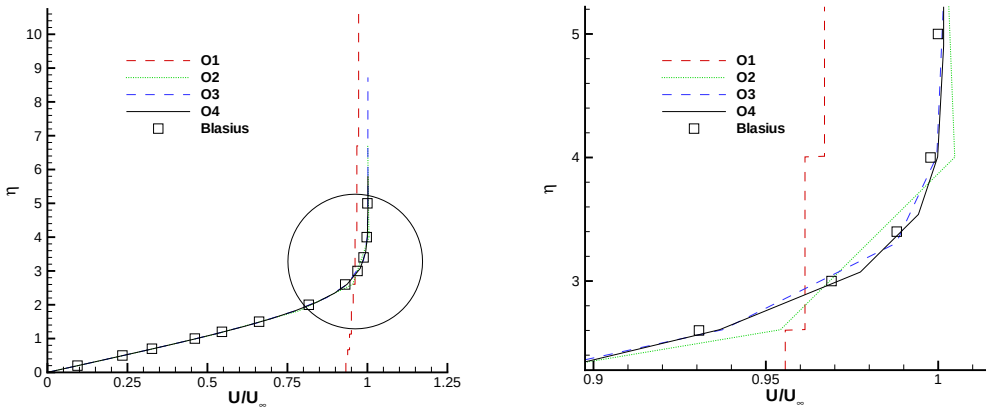
method and the BR2 scheme for spatial discretisation. The implicit approach is indispensable, since the explicit time step restriction is very sharp, in particular, for the stretched boundary layer triangles.

The boundary condition setup is as follows. At the left boundary and at the upper boundary, farfield conditions are prescribed with freestream flow. At the right boundary extrapolation boundary conditions are used. For the lower boundary symmetry conditions before ($\frac{x}{c} < 0$) and after ($\frac{x}{c} > 1$) the plate and for the plate ($0 \leq \frac{x}{c} \leq 1$) adiabatic wall boundary conditions are imposed. In figure 6.25 the Mach number contours for P^0-up to P^3-elements are shown. Note, that a smooth solution is only obtained for the third and fourth order solution on the coarse grid and that the boundary layer is resolved by 2-4 cells in wall normal direction.

Figure 6.26 shows the plate tangential velocity against normalised wall distance η at 70% plate length ($\frac{x}{c} = 0.7$). The nondimensional wall distance η is computed with the freestream velocity U_∞ and the kinematic viscosity ν as

$$\eta = y\frac{U_\infty}{2\nu x}$$

As expected from figure 6.25, the solution obtained with P^0-elements (O1) is completely away from the physics for the coarse grid we used in this study. Note that the boundary layer is discretised with only 3 cells here.



(a) Complete view

(b) Zoomed outer, nonlinear part

Figure 6.26.: Tangential velocity profiles at $x/c = 0.7$ compared to Blasius solution

Especially the nonlinear velocity distribution in the outer boundary layer part is better approximated by the third and fourth order scheme, see figure 6.26(b). Note that in our implementation, visualisation is always done piecewise linearly. For the higher order

(P^2 and P^3) cell solutions we divide these cells into several subcells. The solution in these subcells is visualised approximately linearly, which explains the piecewise linear distributions in figure 6.26(b).

In the left of figure 6.27, normal velocity profiles are compared to the Blasius solution. Noticeable here is, that the higher order polynomial discretisations are nearly producing continuous solutions on a coarse grid despite of the use of a discontinuous discretisation scheme. On the right of figure 6.27, the normalised temperature $\frac{T}{T\infty}$ layer is shown. The computed temperature profiles are consistent with the used adiabatic wall boundary conditon. The theoretical approximative value for the adiabatic wall temperature T_w according to [95] is

$$T_w = T_\infty \left(1 + \sqrt{Pr}\frac{\kappa - 1}{2} Ma_\infty^2\right). \tag{6.3}$$

The ratio of adiabatic wall temperature and farfield temperature is $T_w/T_\infty = 1.01524$. This is in very good agreement with the theoretical prediction of 1.01527 from (6.3).

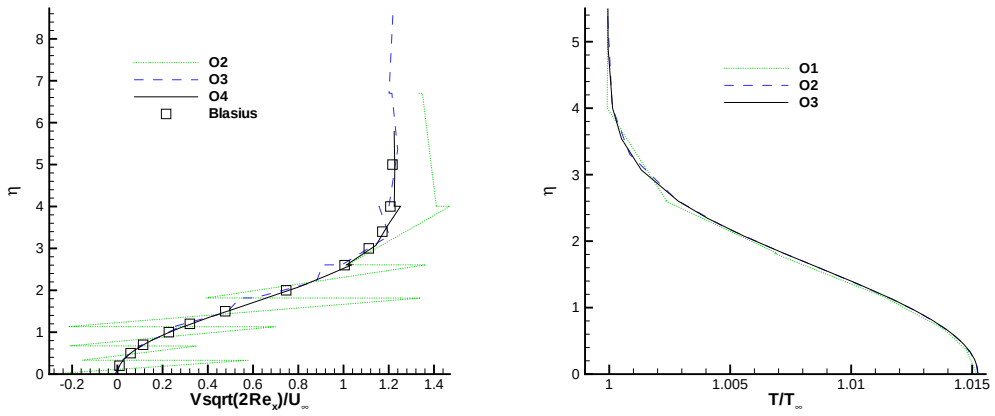


Figure 6.27.: Normal velocity and temperature profiles at $x/c = 0.7$ compared to Blasius solution

In figure 6.28, the skin friction coefficient is plotted against plate length. The solution quality, assessed by the deviation to the boundary layer theoretical Blasius solution, increases with increased polynomial ansatz order.

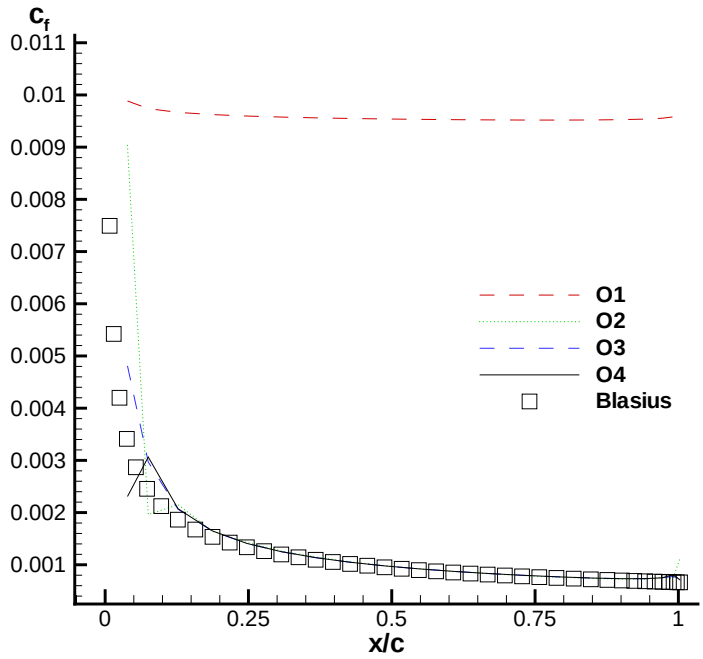
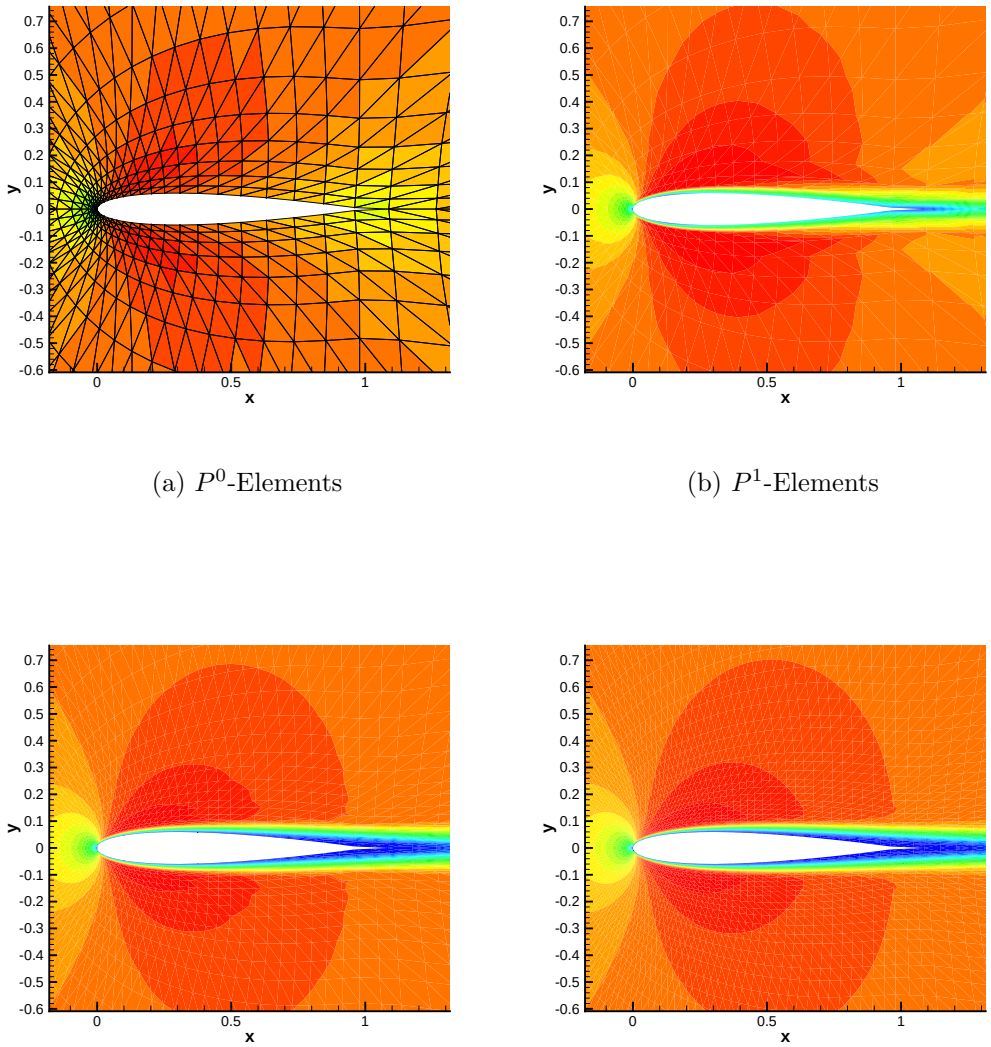


Figure 6.28.: Skin friction coefficient

A better skin friction solution, also at the leading edge area, can be obtained by refining the streamwise leading edge grid resolution, which is quite coarse here, see figure 6.25.

6.2.1.4. NACA0012

For the further validation of the laminar implementation we calculated the flow over a NACA0012 airfoil ($\mathrm{Ma}_\infty = 0.5, \alpha = 0^o,\ \mathrm{Re}_\infty = 5000$). In figure 6.29 the Mach number contours for P^0- up to P^3-elements are shown. Note, that the boundary layer is only resolved with the second and higher order scheme on the very coarse grid. A detailed analysis of the leading edge area, see figure 6.30, demonstrates that the smoothness of the solution at the stagnation point is strongly improved by the fourth order scheme. Also noteable is, that the boundary layer is well resolved by the fourth order scheme with only one cell in the wall normal direction.



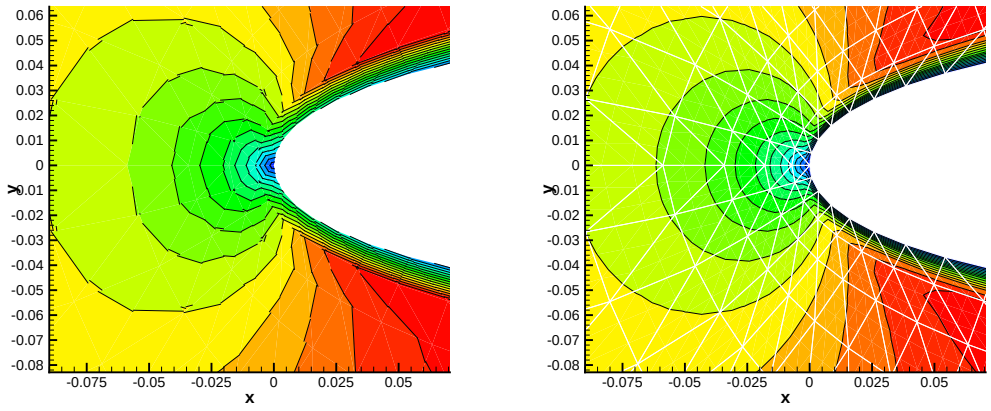
(a) P^0-Elements

(b) P^1-Elements

(c) P^2-Elements

(d) P^3-Elements

Figure 6.29.: Grid and Mach number contours of NACA0012 airfoil.



(a) P^1-Elements

(b) P^3-Elements

Figure 6.30.: Detailed comparison of leading edge area (Mach number contours).

We now want to compare the global pressure and drag coefficients for several order (O1-O4) computations on a very coarse triangularly split structured C-grid (64×16). The solution quality, assessed by the deviation to the reference values produced by the MSES [40] code, increases with increased polynomial ansatz order, see table 6.5 below. For comparison, we also added the results of the DG scheme of Bassi and Rebay [13] and of the triangular FV scheme from Mavriplis [70], who used a much finer split grid (320×64).

Element type	Triangles	Order	Pressure drag $C_{d,p}$	Viscous drag $C_{d,v}$
Constant	$64 \times 16 \times 2$	1	0.06495	0.06465
Linear	$64 \times 16 \times 2$	2	0.02394	0.01476
Quadratic	$64 \times 16 \times 2$	3	0.02038	0.03445
Cubic	$64 \times 16 \times 2$	4	0.02242	0.03290
MSES[40]	—	-	0.02318	0.03313
Cubic [13]	$64 \times 16 \times 2$	4	0.02208	0.03303
Constant (FV)[70]	$320 \times 64 \times 2$	2	0.02290	0.03320

Table 6.5.: Comparison between calculated drag coefficients for a NACA0012 section (O1-O4)

A local comparison of the surface pressure and friction coefficients is also shown in figure 6.31. As expected from the global force coefficients, the local coefficients are best predicted by the fourth order scheme. The differences between the solution with P^2 and

P^3 elements are very low for the pressure solution, whereas the friction coefficients show bigger differences.

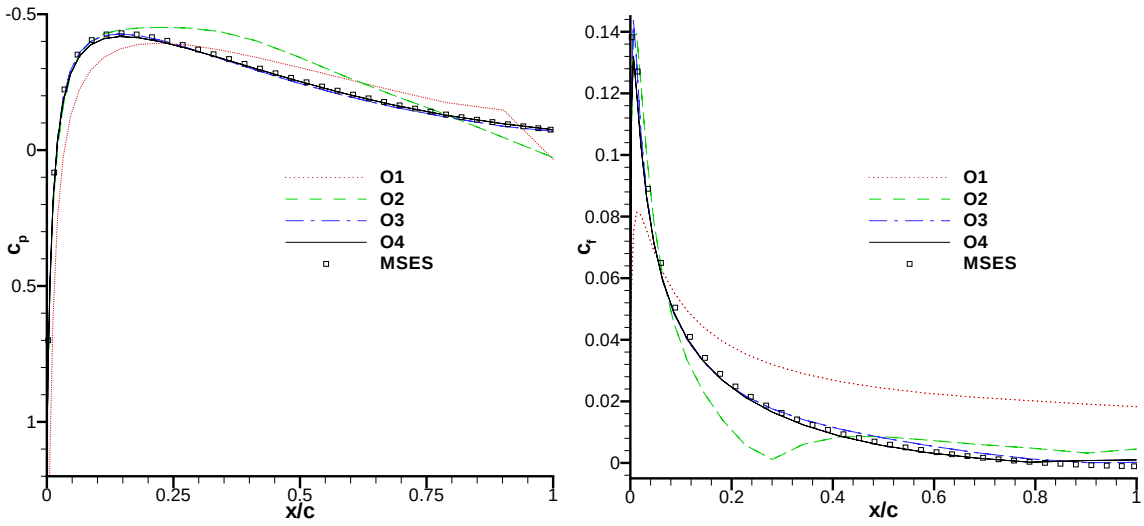
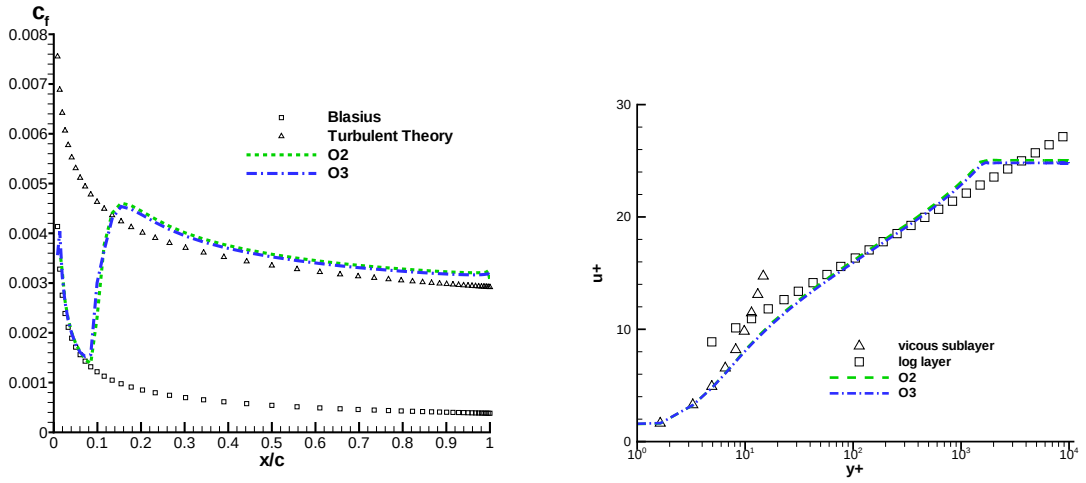


Figure 6.31.: Pressure coefficient and skin friction distribution for the Naca0012 airfoil ($Ma_\infty = 0.5$, $\alpha = 0^0$, $Re_\infty = 5000$)

6.2.2. Turbulent computations

6.2.2.1. Flat plate ($\mathbf{Ma}_\infty = 0.3$, $\mathbf{Re}_\infty = 3e6$)

For the validation of the turbulence model implementation we calculated the flow over a flat plate ($\mathrm{Ma}_\infty = 0.3$, $\mathrm{Re}_\infty = 3e6$) with prescribed transition at 10% plate length. We compared the skin friction distribution as well as tangential velocity profiles for P^1, P^2 and P^3 elements. For comparison theoretical predictions are plotted as reference results. Firstly, we assess the calculations with the k-ω model on a relatively fine triangularly split structered H-grid (88×38 with 48 cells along the plate).



(a) Skin friction distribution

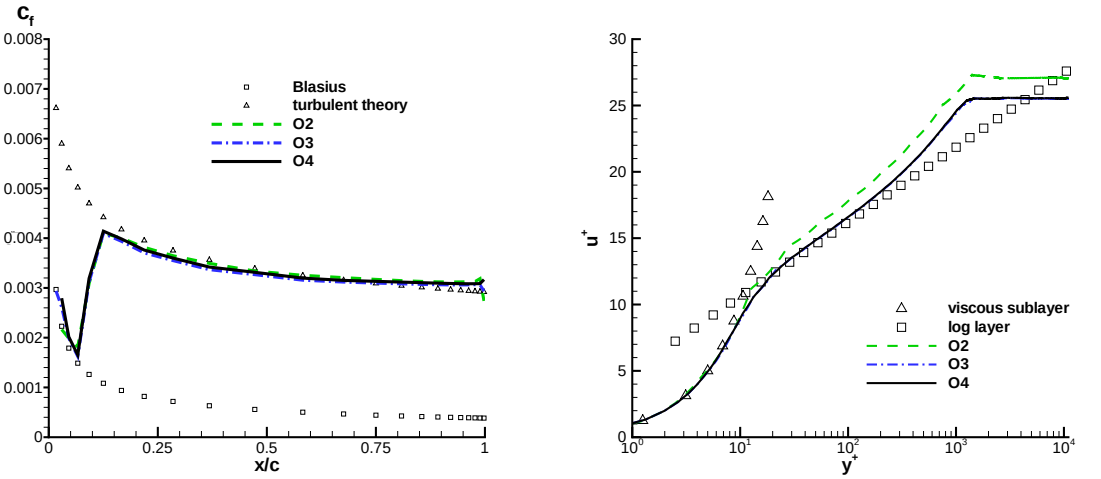
(b) Normalized velocitiy profile at x/c=0.7

Figure 6.32.: Turbulent boundary layer results with k-ω turbulence model

A good agreement between theoretical and computed skin friction can be observed, see figure 6.32(a). We also compared the normalised velocity profile in figure 6.32(b), namely $u+$ against $y+$, that also matches favourably. The fourth order solution is neglected in both diagrams here, because it is identical by line thickness to the third order simulation values.

The above comparisons for the same flow case are performed for validation of the SA turbulence model implementation as well. In order to better assess the influence of the discretisation order, we made these computations with a much coarser triangularly split structered H-grid (44×13 with 24 cells along the plate).

The computed skin friction distributions for all element types (P^0 to P^3) are in good agreement with the theoretical values, figure 6.33(a). Only the second order solution overpredicts the velocity in the logarithmic layer, see figure 6.33(b).

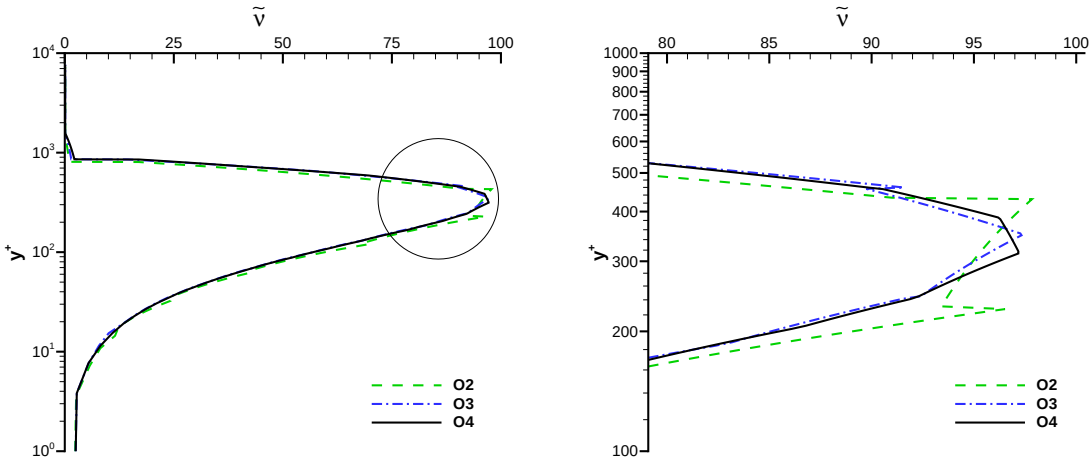


(a) Skin friction distribution

(b) Normalized velocitiy profile at x/c=0.7

Figure 6.33.: Turbulent boundary layer results with SA turbulence model

In figure 6.34 the working variable $\tilde{\nu}$ is plotted against y^+. The solution continuity improves with increasing polynomial degree of the ansatz function. Note, that only 3 cells are used for the resolution of the mainly parabolic distribution (figure 6.34(b)) of $\tilde{\nu}$, and that in our implementation, visualisation is always done piecewise linear. Note again that for the higher order (P^2 and P^3) cell solutions we divide these cells into several subcells. The solution in these subcells is visualised approximately linear, which explains the piecewise linear distributions in figure 6.34(b).



(a) Complete view

(b) Zoomed view, parabolic part

Figure 6.34.: $\tilde{\nu}$-profiles at $x/c = 0.7$

6.2.2.2. Aerospatiale-A airfoil ($\mathbf{Ma}_\infty = 0.15$, $\alpha = 3.4^0$,$\mathbf{Re}_\infty = 3.13e6$)

The next test case is taken from the European Computational Aerodynamics Research Project (ECARP) [47]. The low Mach number flow around an Aerospatiale-A airfoil ($\mathrm{Ma}_\infty = 0.15$, $\alpha = 3.4^0$,$\mathrm{Re}_\infty = 3.13e6$) is characterised by long laminar startup distances, namely 12% on the suction and 30% one the pressure side, respectively. We used the SA model, with prescribed transition for the simulation of the turbulent areas. The pressure and skin friction distribution of our second order computation (P^1 elements) are in excellent agreement with the experiment as wells as the computational values taken from [47], see figure 6.35. Note, that we used a normally coarsened version of the mandatory structured C-mesh (256×64).

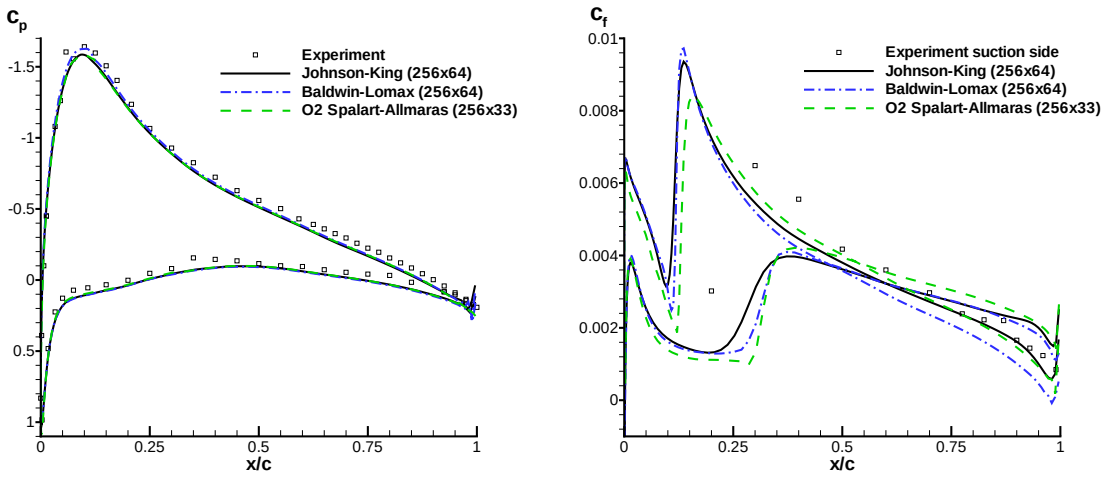
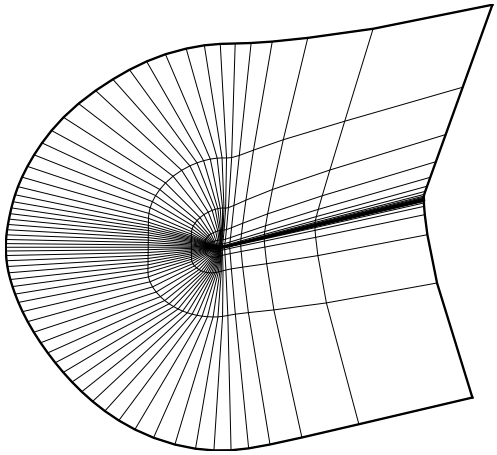
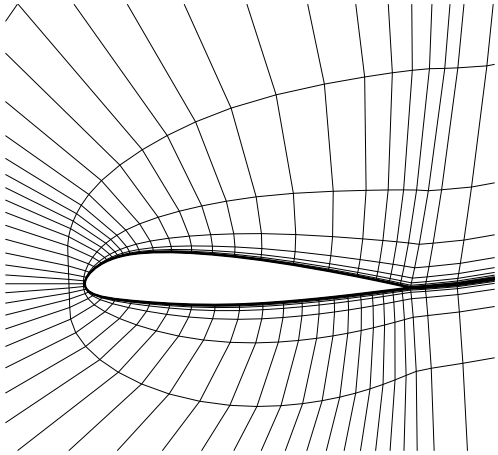


Figure 6.35.: Pressure coefficient and skin friction distribution for the Aerospatiale-A airfoil ($\mathrm{Ma}_\infty = 0.15$, $\alpha = 3.4^0$, $\mathrm{Re}_\infty = 3.13e6$)

In order to test our curved boundary approach for turbulent flow cases, we again calculated the flow around the Aerospatiale-A airfoil. However, now we coarsened the structured mandatory C-grid (256×64) in normal and streamwise direction by the factor four, see figure 6.36. Note, that only 4-7 cells in wall normal direction remain in the boundary layer.



(a) Complete grid (64×16)

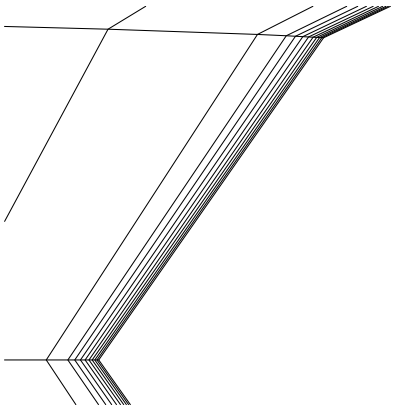


(b) Near grid

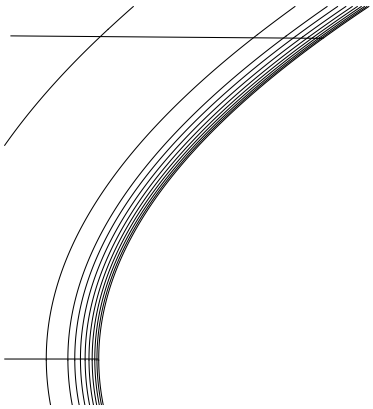
Figure 6.36.: Coarse C-grid for A-Airfoil

Since stretched one sided curved triangles would degenerate we require double sided curved quadrilaterals as discretisation elements in the boundary layer. Figure 6.37 shows a cutout at the leading edge area of the original structured grid (figure 6.37(a)) and the

partially curved grid (figure 6.37(b)). The maximum extension of curved layers is 12 at the leading edge, because this is the region with highest curvature of the Aerospatiale-A airfoil geometry.



(a) Original grid



(b) Curved grid

Figure 6.37.: Cutout of the leading edge area of the Aerospatiale-A airfoil, coarse C-grid (64x16)

In figure 6.38 the pressure and skin friction distributions computed with P^1, P^2 and P^3

elements are compared with experimental values and the second order solution obtained on the finer C-grid (256×33). The solution quality, assessed by the deviation to the experimental values and fine grid O2-solution, increases with increased polynomial ansatz order. Especially the prediction of the pressure at the suction peak is fundamentally better for the high-order computations. As expected, the deviations are bigger for the skin friction. Particularly, the accurate simulation of the laminar startup is only achieved by the third and fourth order solutions.

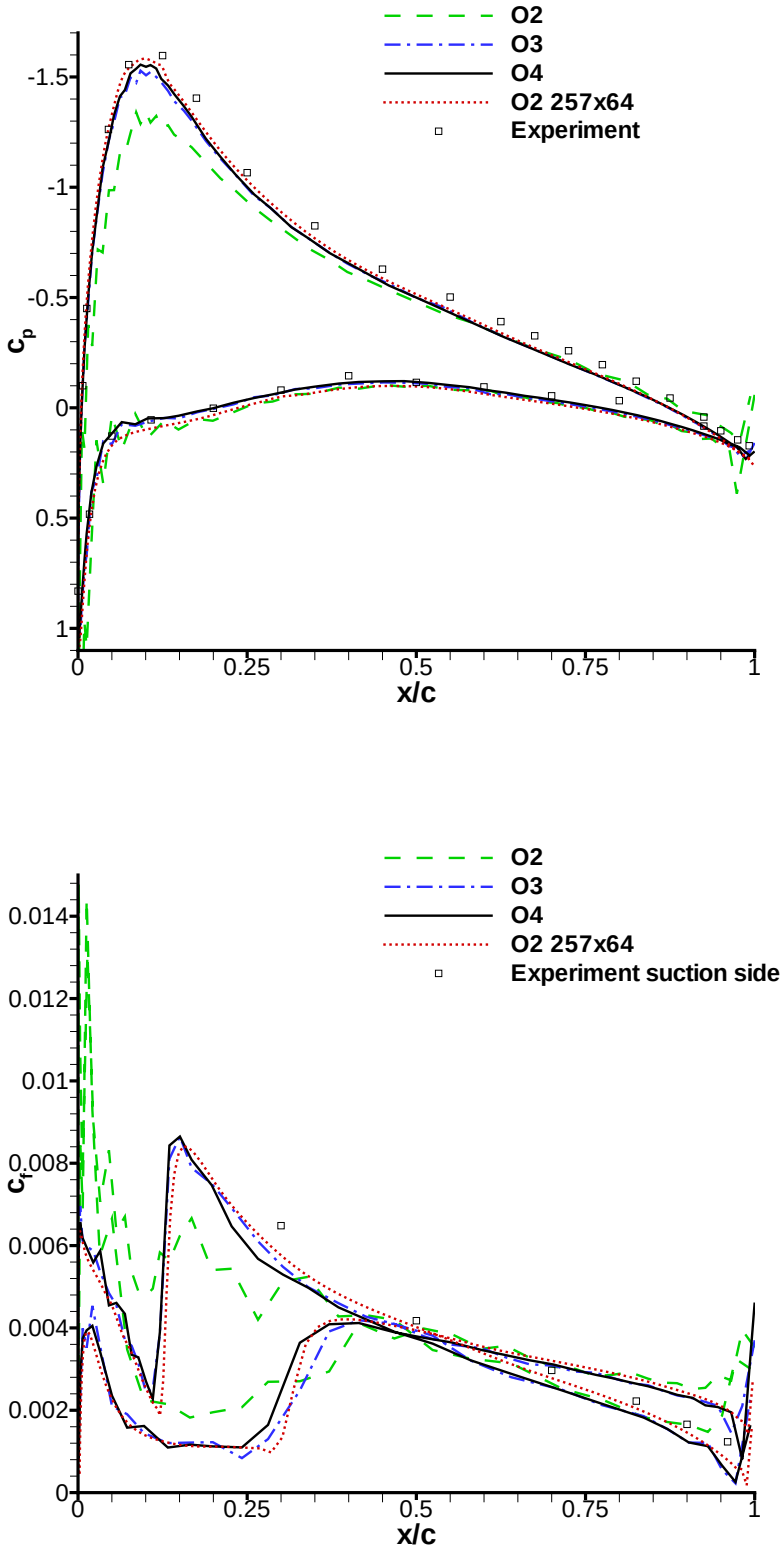
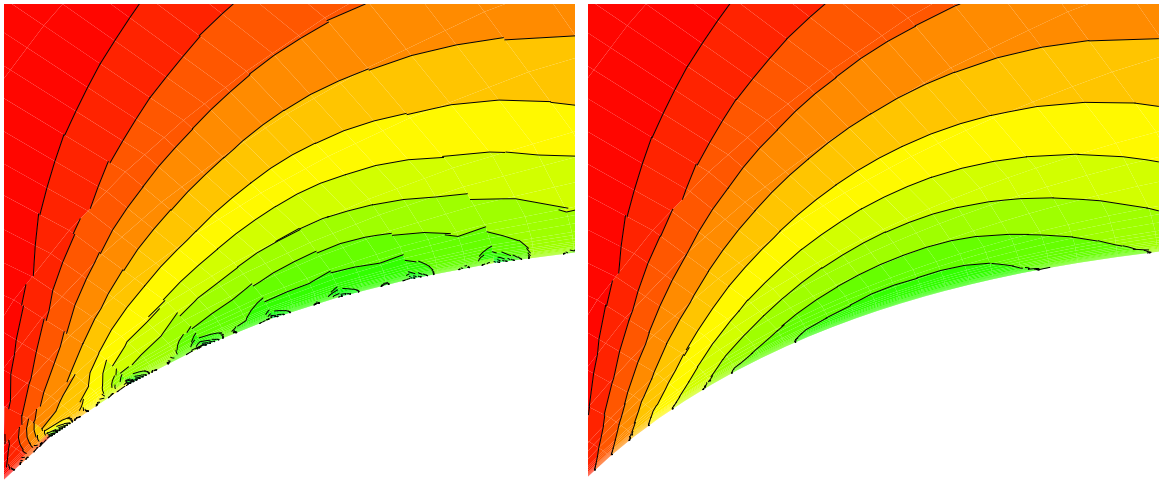


Figure 6.38.: High-order computed pressure and skin friction coefficient for the Aerospatiale-A airfoil with curved C-grid (64×16), SA turbulence model

The requirement of curved boundaries and layers for simulating high-order (turbulent) flows on coarse grids can be demonstrated by comparing results between simulations with straight and curved P^3 elements. As can be observed in the pressure contour at the region of the suction peak in figure 6.39(a), fluctuations at the kinks of the boundary develop in the straight element solution, whereas the curved boundary solution is smooth along the boundary 6.39(b).



(a) Straight elements (b) Curved elements

Figure 6.39.: Pressure contours at the upper side of the Aerospatiale-A airfoil with coarse C-grid (64×16)

A qualitative comparison of the surface pressure and skin friction distribution is shown in figure 6.40. The pressure distribution obtained with the straight elements contains strong oscillations at the leading edge as well as in the suction peak area at the upper airfoil side. The skin friction distribution also contains oscillations, but additionally the mean level of friction is drastically underpredicted.

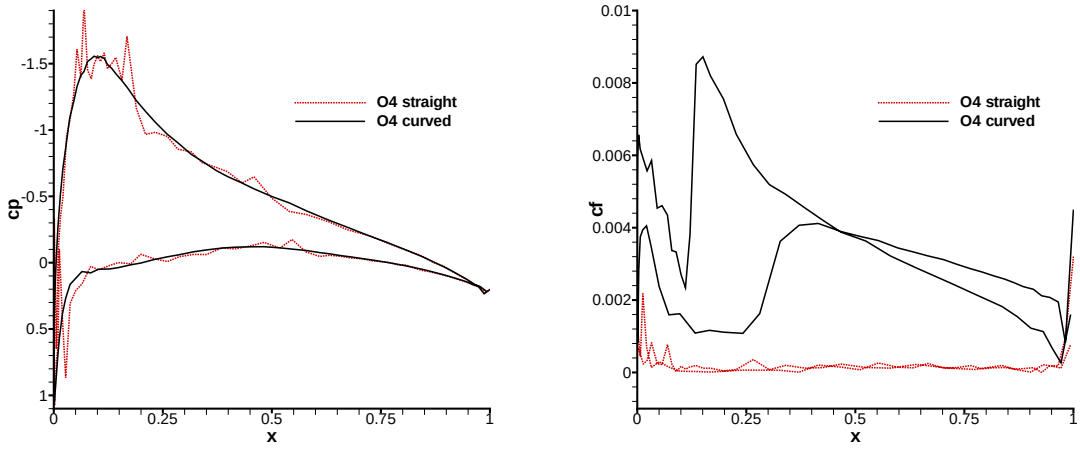
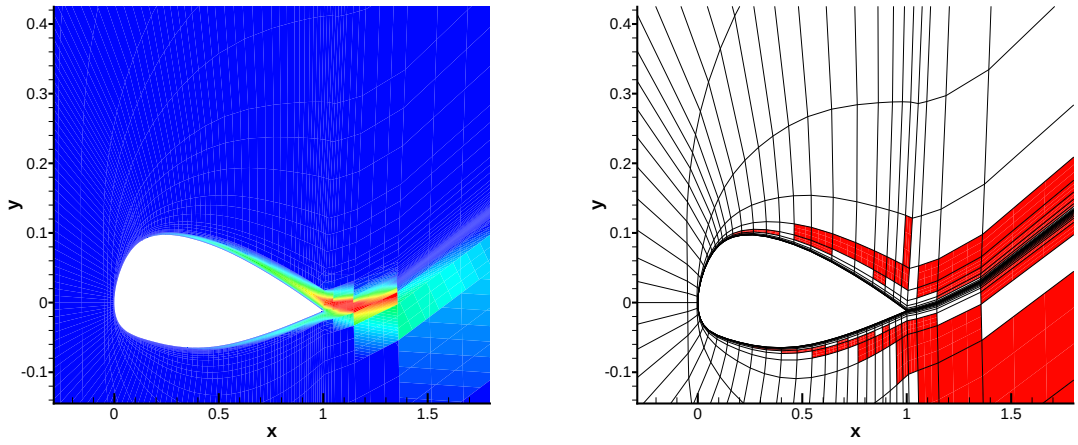


Figure 6.40.: Comparison of curved grid and straight grid solution O4 ($\mathrm{Ma}_\infty = 0.15$, $\alpha = 3.4^0$, $\mathrm{Re}_\infty = 3.13e6$)

6.2.2.3. Turbulence model limiting

An aspect not discussed so far for the individual test cases is the limiting of negative turbulence model quantities (k or $\tilde{\nu}$). All turbulent test cases are computed in the same fashion: Consecutively running the first up to the fourth order scheme by restarting (O1→O2→O3→O4). This leads to limiting of approximately 1 to 5% of the cells as soon as the solution scheme is switched to second or higher order,



(a) Turbulent viscosity

(b) Limited cells in red

Figure 6.41.: Turbulent viscosity distribution and limited cells in curved grid solution O4 (y direction stretched by factor 3)

and partially negative solution areas in underresolved regions are a fundamental ingredient of the discretisation method. Consequently, convergence speed as well as the maximum possible implicit time step size decreases. However, the limited cells are nearly exclusively cells in the strongly underresolved outer turbulent/not turbulent (TNT) region, see figure 6.41, which apparently seems not to affect the general solution quality.

6.3. Parallel performance

In order to assess the quality of our parallelisation we used again a circular cylinder mesh consisting of about 88.000 cells and split it into up to 128 zones. The computations are performed with triangular P^1 elements. On an ordinary Linux cluster consisting of dual Xeon nodes and Infiniband interconnect, the efficiency for this comparatively small case exceeds 88% even on 64 CPUs, which results in about 1200 cells per processor only. If the number of CPUs is doubled again, efficiency clearly decreases, but still is about 75%, see figure 6.42.

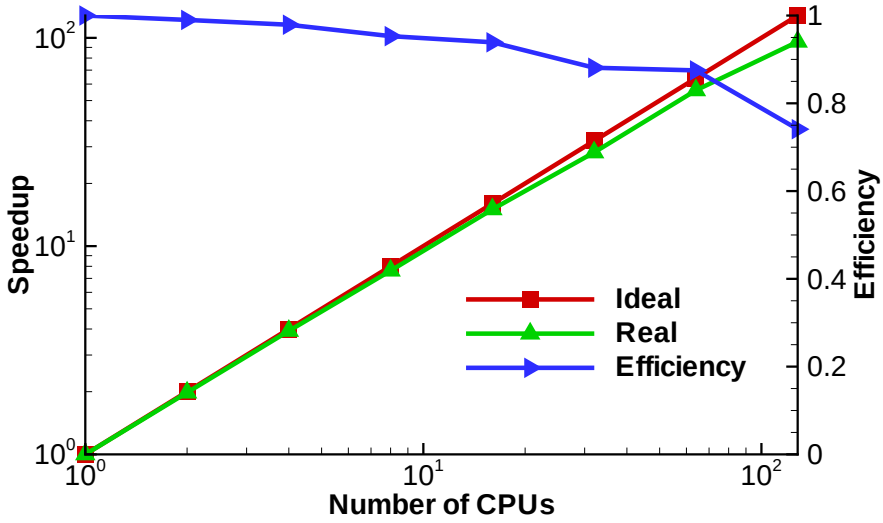


Figure 6.42.: Parallel speedup and efficiency

7. Conclusions and future prospects

7.1. Conclusion

The general conclusion is that in this work an important step has been taken in the assessment of high-order DG based discretisation techniques for aerodynamic flow problems on unstructured grids. It is clear that there is still room for improvement. The following sections summarise the conclusion of this work.

7.1.1. Euler equations

The discretisation of the inviscid Euler equations with the aid of the DG approach essentially has confirmed the recently made statements of several authors, cited in the introductory chapter. The validation with one- and two dimensional examples has shown that the DG Euler solver, implemented and used as basis for the viscous flow solver, delivers accurate results. The accuracy of the code, on unstructured grids, has been validated up to fourth order in space and time. Several approaches for the geometrically high-order treatment of boundaries have been implemented and compared. The main result is, that more accurate solutions can only be obtained, if the boundary is approximated in a continuously differentiable fashion by curvilinear elements. In order to obtain non oscillating pressure distributions in strongly curved regions, smoothness of the second order boundary derivatives seems also desirable. Finally, the efficiency of implicit backward Euler discretisation method in time has been analysed for steady flow problems. Here, different Krylov subspace methods in combination with several preconditioners were compared. The ILU preconditioned implicit GMRES DG solver turned out to be a powerful method for the solution of steady flow problems. Especially the high-order explicit simulation of steady flow, can be extremely accelerated with the implicit solution technique. Both temporal solution approaches (explicit and implicit) proved to be extremely robust.

7.1.2. Navier-Stokes equations

The situation drastically complicates, if viscous flows for moderate Reynolds numbers, should be calculated based on the (laminar) NS equations. This is mainly due to the introduction of diffusive second order derivatives, which are a lot more difficult to handle by the DG approach than the first order derivatives in the convective terms. There are several DG methods for second order terms, but only very few schemes of them are able to retain the abilities of the DG Euler solver prescribed in the previous section.

Compared to the inviscid DG schemes, the understanding of these schemes is by far less known. Hence, the implementation as well as the computational complexity increased considerably. However, the chosen two schemes (LDG and BR2), show optimal order of convergence in the selected test case (Poiseuille tube flow). This high-order behaviour impressively is reflected in the solution of unsteady and steady flow fields (circular cylinder, flat plate and NACA0012 airfoil) on extremely coarse grids, in both, stream-wise and wall normal direction. This means, that the boundary layer has been accurately resolved with the high-order computations, even if very few cells in wall normal direction were used for discretisation. The high-order capability of predicting smooth flow features, like vortices, was shown impressively by the example of cylinder flow. In general, comparisons have shown very good agreement with reference FV or FD solutions, which are obtained on much finer grids. If laminar flows with moderate Reynolds number (flat plate and NACA0012 airfoil) were to be calculated, the implicit solution approach in time, turned out to be a lot more efficient than the explicit one.

7.1.3. Reynolds-averaged Navier-Stokes

The implementation of viscous DG methods for the RANS equations showed to be more complex than that of the laminar NS equations.

One fundamental disadvantage or handicap of the DG approach applied for the simulation of high Reynolds number turbulent flows is the resolution of the boundary layer. Since coarse grids are sufficient if high order elements are used, triangular and tetrahedral curved boundary layer elements can easily degenerate. The problem can be avoided with the aid of curved quadrilateral, prismatic or heaxahedral elements in the boundary layer. However, the generation of the desired curved grid layers is no standard grid generation problem. It complicated the implementation and increases the computational costs.

The further assessment of the DG methods for the RANS equations strongly depends on the used turbulence model. Therefore the summary is split into the $k-\omega$ model and the SA model implementation.

7.1.3.1. Wilcox $k-\omega$ model

Due to the logarithmic behaviour of the specific dissipation rate ω, the simulation with coarse grid resolution in wall normal direction led to non physical positivity problems of ω, and furthermore the stability of the overall solution algorithm was strongly decreased. These experiences correspond with the recently published papers of Bassi et al. The logarithmised expression of the ω equation only insignificantly reduced these problems, since the turbulent kinetic energy k can take negative, unphysical values furthermore. This is surely due to the underesolved regions near the wall and the largely choosen time steps of the implicit time-stepping approach. Several limitation strategies are possible. We implemented the following two. First hard limitation, which is nothing else, than simple limitation of k solution values to positive values after every time step. The second approach is adopted from Bassi et al. again and consists in, allowing negative

values in the solution, but to limit the eddy viscosity and several source terms in the model equations to positive values. However, for both limitation strategies (in our implementation) further grid resolution and time step restrictions, had to be accepted, to ensure stable convergence. In spite of the robustness problems, the laminar-turbulent boundary layer results, obtained with the $k-\omega$ model are satisfactory. The turbulence model quantities as well as the flow field quantities reflected the desired behaviour.

Instead of trying further modifications for increasing the robustness of the DG based $k-\omega$ method, like Bassi et al. have proposed, we decided to test a further turbulence model, the one-equation model of Spalart and Allmaras.

7.1.3.2. Spalart-Allmaras model

The problem of unphysical, negative turbulent viscosity values is also present for the SA model variable $\tilde{\nu}$. The resolution of the decay of turbulence away from the wall is often underesolved if standard grids are used. That means, that the resolution is acceptable for the boundary layer values of the main equations (velocity, energy etc.), whereas the wall normal coarsening of the grid is arranged to fast for the SA turbulence model. It is mainly this region, namely the outer part of the flow boundary layer, where $\tilde{\nu}$ takes negative values. These negative values often led to numerical stability problems.

The choosen hard limitation approach, simple limitation of $\tilde{\nu}$ solution values to positiv valueas after every time-step, has shown to increase the robustness of the method and seems not to effect the overall solution quality. Therefore, in the present DG flow solver, the SA model is the preferred model for the turbulent computations. Excellent results with the SA model have been obtained in combination with very coarse curved (quadrilateral) boundary layer grids. Again, a comparison of curvilinear and straight boundary clearly showed the urgent need of the deformed curved grid solution technique.

7.1.4. Parallel efficiency

The parallel efficiency of the implemented DG solver is excellent for the (explicitely) computed viscous flow cases, documented by the presented test case in section 6.3. The assessment of the computational efficiency, compared to other standard CFD codes, was not part of this study. In order to analyse this aspect sensibly, we have to optimise our solution scheme before.

7.2. Prospects

The future prospect of DG work, can be clearly identified, by reflecting the conclusions of the previous section. In principle DG based simulations of the Euler- and more important the Navier-Stokes and RANS equations are possible. The quality of the obtained high-order results is very good, but the improvement of several DG solver parts has to be undertaken.

The robustness of the RANS DG solver has to be further increased. Two possible working directions could be the following. First, the introduction of further penalty

terms in the solution scheme as clever positivity constraints for the turbulence quantities. Or/and second, the inclusion of wall functions into the turbulence models, in order to defuse the sensitive behaviour near walls.

The efficiency of the solver has to be strongly increased. One possibility is the changeover to a matrix free Newton-Krylov method, in order to solve the linear system more efficiently concerning memory requirements as well as CPU time. Another important working point is the parallelisation of the implicit time-stepping approach.

A completely different approach to speedup convergence would be $hp-$multigrid. This has also been done, but mainly for the convergence acceleration of the inviscid Euler computations. Especially up to date, no experience in combination with turbulence modeling has been gained.

In the future the extension of the code by including (curved) tetrahedral, prismatic and hexahedral elements is planned, in order to test the scheme on three-dimensional detached eddy simulations with modified SA or k-ω model. Work in this direction has already begun, starting from the software developed in this thesis.

Finally, in order to tap the full potential of the DG discretisation, shock limitation and adaptation strategies should be included into the flow solver.

A. Results

In this chapter some results, which are not discussed in detail in the main results section, are shown for the sake of completeness.

A.1. Toros test cases

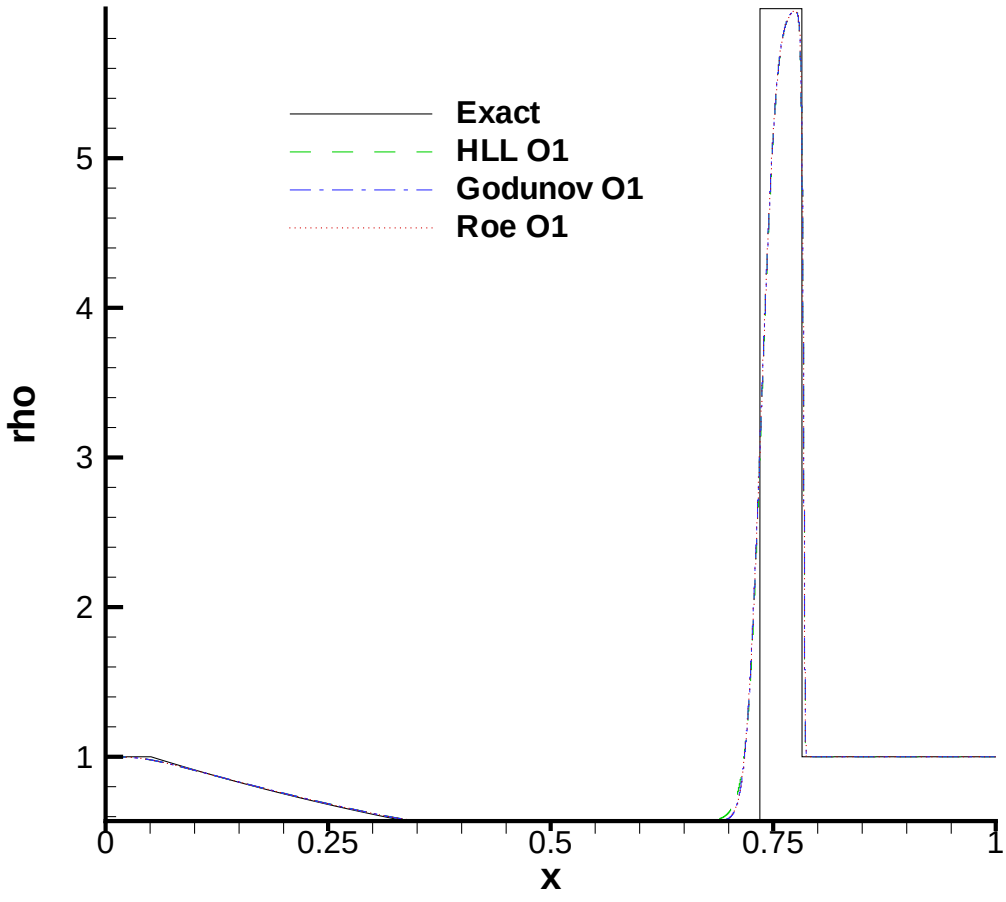


Figure A.1.: Test case number 3, density

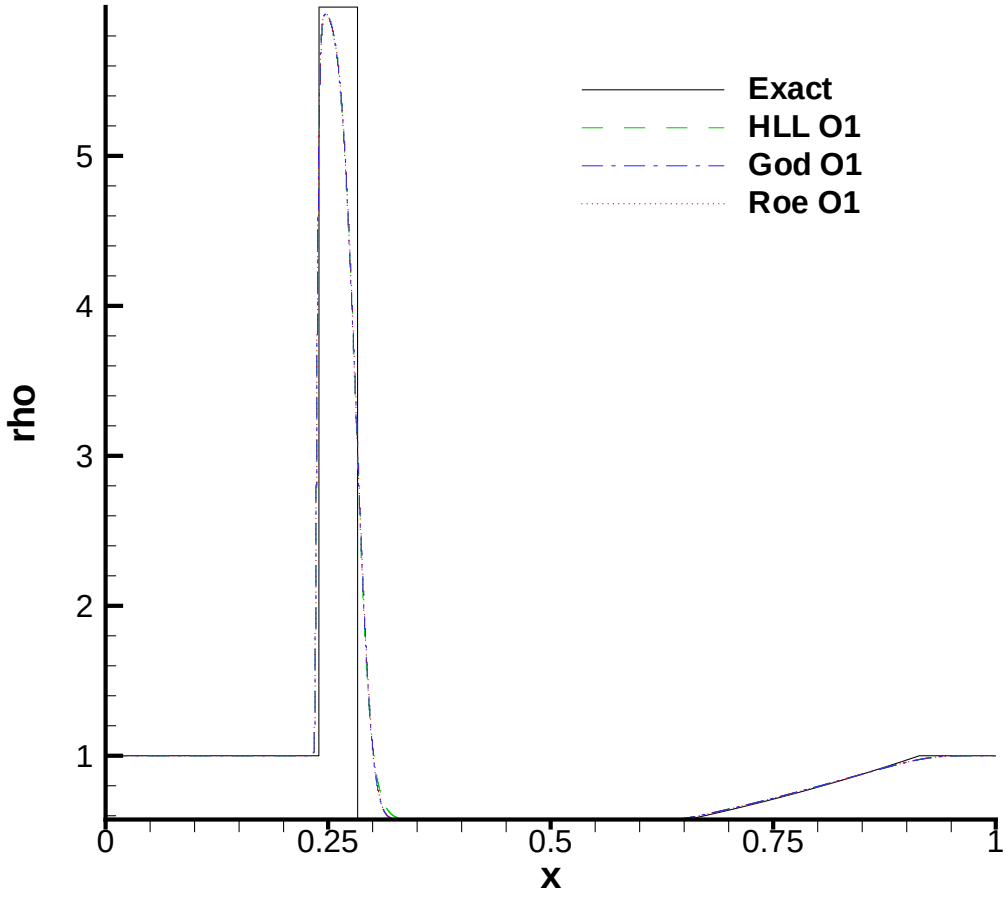


Figure A.2.: Test case number 4, density

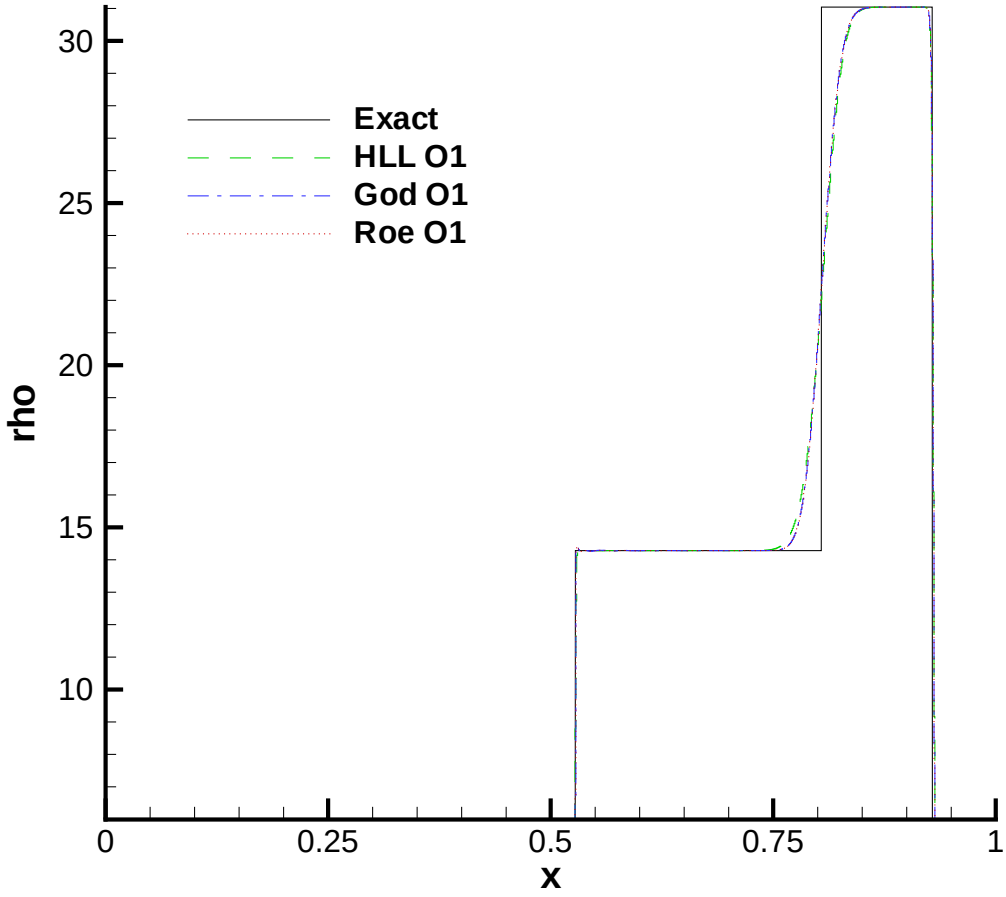


Figure A.3.: Test case number 5, density

B. Basis functions

In this section all the used basis function of all element types used in SUNWinT are listed for the sake of completeness. The formulas have been calculated from equation (3.45) and (3.47) with the help of the symbolic mathematic package Maple.

B.1. Line

Basis/Order	1	2	3	4
1	1	$1-\xi$	$(1-2\xi)(1-\xi)$	$-\frac{1}{2}(3\xi-1)(3\xi-2)(\xi-1)$
2	-	ξ	$4\xi(1-\xi)$	$\frac{9}{2}\xi(3\xi-2)(\xi-1)$
3	-	-	$\xi(2\xi-1)$	$-\frac{9}{2}\xi(3\xi-1)(\xi-1)$
4	-	-	-	$\frac{1}{2}\xi(3\xi-1)(3\xi-2)$

Table B.1.: Basis functions for reference line

B.2. Triangle

Basis/Order	1	2	3	4
1	1	1	1	1
2		$1+2\xi+\eta$	$1+2\xi+\eta$	$1+2\xi+\eta$
3		$-1+3\eta$	$-1+3\eta$	$-1+3\eta$
4			$1-2\eta+\eta^2+6\xi\eta-6\eta+6\eta^2$	$1-2\eta+\eta^2+6\xi\eta-6\eta+6\eta^2$
5		-	$1+5\eta^2+10\xi\eta-6\eta-2\xi$	$1+5\eta^2+10\xi\eta-6\eta-2\xi$
6		-	$1-8\eta+10\eta^2$	$1-8\eta+10\eta^2$
7		-		$-1+\eta^3-24\xi\eta+30\xi^2\eta+12\xi\eta^2+20\xi^3+12\xi+3\eta-3\eta^2-30\xi^2$
8		-		$-1+7\eta^3+42\xi\eta^2-15\eta^2-48\xi\eta+9\eta+42\xi^2\eta-6\xi^2+6\xi$
9		-		$-1+21\eta^3+42\xi\eta^2-33\eta^2-24\xi\eta+13\eta+2\xi$
10		-		$-1+15\eta-45\eta^2+35\eta^3$

Table B.2.: Basis functions for reference triangle

B.3. Quadrilateral

B/O	1	2	3	4
1	1	1	1	1
2		$-1+2\xi$	$-1+2\xi$	$-1+2\xi$
3		$-1+2\eta$	$-1+2\eta$	$-1+2\eta$
4			$1-6\xi+6\xi^2$	$1-6\xi+6\xi^2$
5			$1-2\eta-2\xi+4\xi\eta$	$1-2\eta-2\xi+4\xi\eta$
6			$1-6\eta-6\eta^2$	$1-6\eta-6\eta^2$
7				$-1+12\xi-30\xi^2+20\xi^3$
8				$-1+2\eta+6\xi-12\xi\eta-6\xi^2+12\xi^2\eta$
9				$-1+6\eta-6\eta^2+2\xi-12\xi\eta+12\xi\eta^2$
10				$-1+12\eta-20\eta^2+20\eta^3$

Table B.3.: Basis functions for reference quadrilateral

B.4. Tetrahedron

B/O	1	2	3	4
1	1	1	1	1
2		$-1+2\zeta+\xi+\eta$	$-1+2\zeta+\xi+\eta$	$-1+2\zeta+\xi+\eta$
3		$-1+\eta+3\xi$	$-1+\eta+3\xi$	$-1+\eta+3\xi$
4		$-1+4\eta$	$-1+4\eta$	$-1+4\eta$
5			$1-2\xi-2\eta+\xi^2+2\xi\eta+\eta^2$ $-6\zeta+6\zeta\xi+6\zeta\eta+6\zeta^2$	$1-2\xi-2\eta+\xi^2+2\xi\eta+\eta^2-6\zeta+6\zeta\xi+6\zeta\eta+6\zeta^2$
6			$1+\eta^2-2\eta+6\xi\eta+2\zeta\eta$ $-6\xi-2\zeta+5\xi^2+10\zeta\xi$	$1+\eta^2-2\eta+6\xi\eta+2\zeta\eta-6\xi-2\zeta+5\xi^2+10\zeta\xi$
7			$1-2\eta+\eta^2+8\xi\eta-8\xi+10\xi^2$	$1-2\eta+\eta^2+8\xi\eta-8\xi+10\xi^2$
8			$1+6\eta^2+6\xi\eta+12\zeta\eta-7\eta-2\zeta-\xi$	$1+6\eta^2+6\xi\eta+12\zeta\eta-7\eta-2\zeta-\xi$
9			$1+6\eta^2-7\eta+18\xi\eta-3\xi$	$1+6\eta^2-7\eta+18\xi\eta-3\xi$
10			$1-10\eta+15\eta^2$	$1-10\eta+15\eta^2$
11				$-1+\eta^3-3\eta^2-3\xi^2+\xi^3-24\zeta\xi-6\xi\eta+3\eta+3\xi+30\zeta^2\eta$ $-30\zeta^2+12\zeta-24\zeta\eta+20\zeta^3+12\zeta\eta^2+3\xi^2\eta$ $+3\eta^2\xi+12\zeta\xi^2+30\zeta^2\xi+24\zeta\xi\eta$
12				$-1+\eta^3+6\zeta\eta^2-3\eta^2+9\eta^2\xi-18\xi\eta+48\zeta\xi\eta$ $+6\zeta^2\eta-12\zeta\eta+3\eta+15\xi^2\eta-15\xi^2+7\xi^3-48\zeta\xi+9\eta\xi$ $-6\zeta^2+6\zeta+42\zeta\xi^2+42\zeta^2$
13				$-1+\eta^3+2\zeta\eta^2+13\eta^2\xi-3\eta^2+3\eta+33\xi^2\eta-26\xi\eta$ $+24\xi\eta\zeta-4\zeta\eta-24\zeta\eta+42\zeta\xi^2-33\xi^2+13\xi+2\zeta+21\xi^3$
14				$-1+\eta^3-3\eta^2-45\xi^2+35\xi^3-30\xi\eta+3\eta+15\xi$ $+45\xi^2\eta+15\eta^2\xi$
15				$-1+8\eta^3+48\zeta\eta^2+16\eta^2\xi-17\eta^2+10\eta+8\xi^2\eta-18\xi\eta$ $-54\zeta\eta+48\zeta\xi\eta+48\zeta^2\eta-\xi^2-6\zeta\xi-6\zeta^2+2\xi+6\zeta$
16				$-1+8\eta^3+48\eta^2\xi-17\eta^2+16\zeta\eta^2+10\eta-18\zeta\eta$ $-54\xi\eta+80\zeta\xi\eta+40\xi^2\eta-5\xi^2+2\zeta-10\zeta\xi+6\xi$
17				$-1+8\eta^3-17\eta^2+64\eta^2\xi-72\xi\eta+80\xi^2\eta$ $+10\eta+8\xi-10\xi^2$
18				$-1+28\eta^3-42\eta^2+28\eta^2\xi+56\zeta\eta^2-14\xi\eta$ $-28\zeta\eta+15\eta+\xi+2\zeta$
19				$-1+28\eta^3+84\eta^2\xi-42\eta^2-42\xi\eta+15\eta+3\xi$
20				$-1+18\eta-63\eta^2+56\eta^3$

Table B.4.: Basis functions for reference tetrahedron

C. Quadrature formulas

We introduce the space of polynomial functions of degree at most k

$$P^k := \{p(x) \mid p(x) \text{ is polynomial of degree } \leq k\}.$$

C.1. Line

The numerical integration of a polynomial function $p(x)$ on a reference line of unit length is approximated as

$$\int_0^1 p(x)\,dx = \sum_i \omega_i p(x_i) \text{ for polynomial functions } p(x) \in P^k.$$

The sampling points are nothing else than the roots of the Legendre polynomials.

Exact for $p(x) \in P^k$	Integration points x_i	Weights ω_i
$k=1$	$x_1 = \frac{1}{2}$	$\omega_1 = 1$
$k=3$	$x_1 = \frac{1}{2}\left(1 - \frac{1}{\sqrt{3}}\right)$	$\omega_1 = \frac{1}{2}$
	$x_2 = \frac{1}{2}\left(1 + \frac{1}{\sqrt{3}}\right)$	$\omega_2 = \frac{1}{2}$
$k=5$	$x_1 = \frac{1}{2}\left(1 - \sqrt{\frac{3}{5}}\right)$	$\omega_1 = \frac{5}{18}$
	$x_2 = \frac{1}{2}$	$\omega_2 = \frac{4}{9}$
	$x_3 = \frac{1}{2}\left(1 + \sqrt{\frac{3}{5}}\right)$	$\omega_3 = \frac{5}{18}$
$k=7$	$x_1 = 0.0694318442$	$\omega_1 = 0.1739274232$
	$x_2 = 0.3300094782$	$\omega_2 = 0.3260725770$
	$x_3 = 0.6699905218$	$\omega_3 = 0.3260725770$
	$x_4 = 0.9305681558$	$\omega_4 = 0.1739274232$

Table C.1.: Integration rules for the reference line

C.2. Triangle and quadrilateral

The numerical integration of a polynomial function $p(x,y)$ on a reference triangle is approximated as

$$\int_0^1 \int_0^{1-x} p(x,y)\,dxdy = \sum_i \omega_i p(x_i, y_i) \text{ for polynomial functions } p(x,y) \in P^k.$$

Exact for $p(x) \in P^k$	Integration points x_i, y_i	Weights ω_i
$k=1$	$x_1 = \frac{1}{3}$, $y_1 = \frac{1}{3}$	$\omega_1 = 1$
$k=2$	$x_1 = \frac{1}{6}$, $y_1 = \frac{1}{6}$	$\omega_1 = \frac{1}{3}$
	$x_2 = \frac{2}{3}$, $y_2 = \frac{1}{6}$	$\omega_2 = \frac{1}{3}$
	$x_3 = \frac{1}{6}$, $y_3 = \frac{2}{3}$	$\omega_3 = \frac{1}{3}$
$k=4$	$x_1 = 0.09157620$, $y_1 = 0.09157620$	$\omega_1 = 0.10995174$
	$x_2 = 0.81684759$, $y_2 = 0.09157620$	$\omega_2 = 0.10995174$
	$x_3 = 0.0915762$, $y_3 = 0.81684759$	$\omega_3 = 0.10995174$
	$x_4 = 0.4459485$, $y_4 = 0.4459485$	$\omega_4 = 0.22338159$
	$x_5 = 0.1081303$, $y_5 = 0.4459485$	$\omega_5 = 0.22338159$
	$x_6 = 0.4459485$, $y_6 = 0.1081303$	$\omega_6 = 0.22338159$

Table C.2.: Integration rules for the reference triangle

The numerical integration of a polynomial function $p(x, y)$ on a reference quadrilateral is approximated as

$$\int_0^1 \int_0^1 p(x,y)\, dxdy = \sum_i \omega_i p(x_i, y_i) \text{ for polynomial functions } p(x,y) \in P^k.$$

Exact for $p(x) \in P^k$	Integration points x_i, y_i	Weights ω_i
$k=1$	$x_1 = \frac{1}{2}$, $y_1 = \frac{1}{2}$	$\omega_1 = 1$
$k=3$	$x_1 = \frac{1}{2} + \frac{\sqrt{6}}{6}$, $y_1 = \frac{1}{2}$	$\omega_1 = \frac{1}{4}$
	$x_2 = \frac{1}{2} - \frac{\sqrt{6}}{6}$, $y_2 = \frac{1}{2}$	$\omega_2 = \frac{1}{4}$
	$x_3 = \frac{1}{2}$, $y_3 = \frac{1}{2} + \frac{\sqrt{6}}{6}$	$\omega_3 = \frac{1}{4}$
	$x_4 = \frac{1}{2}$, $y_4 = \frac{1}{2} - \frac{\sqrt{6}}{6}$	$\omega_4 = \frac{1}{4}$
$k=5$	$x_1 = \frac{1}{2} + \frac{\sqrt{15}}{10}$, $y_1 = \frac{1}{2} + \frac{\sqrt{15}}{10}$	$\omega_1 = \frac{25}{324}$
	$x_2 = \frac{1}{2} + \frac{\sqrt{15}}{10}$, $y_2 = \frac{1}{2}$	$\omega_2 = \frac{10}{81}$
	$x_3 = \frac{1}{2} + \frac{\sqrt{15}}{10}$, $y_3 = \frac{1}{2} - \frac{\sqrt{15}}{10}$	$\omega_3 = \frac{25}{324}$
	$x_4 = \frac{1}{2}$, $y_4 = \frac{1}{2} + \frac{\sqrt{15}}{10}$	$\omega_4 = \frac{10}{81}$
	$x_5 = \frac{1}{2}$, $y_5 = \frac{1}{2}$	$\omega_5 = \frac{16}{81}$
	$x_6 = \frac{1}{2}$, $y_6 = \frac{1}{2} - \frac{\sqrt{15}}{10}$	$\omega_6 = \frac{10}{81}$
	$x_7 = \frac{1}{2} - \frac{\sqrt{15}}{10}$, $y_7 = \frac{1}{2} + \frac{\sqrt{15}}{10}$	$\omega_7 = \frac{25}{324}$
	$x_8 = \frac{1}{2} - \frac{\sqrt{15}}{10}$, $y_8 = \frac{1}{2}$	$\omega_8 = \frac{10}{81}$
	$x_9 = \frac{1}{2} - \frac{\sqrt{15}}{10}$, $y_9 = \frac{1}{2} - \frac{\sqrt{15}}{10}$	$\omega_9 = \frac{25}{324}$

Table C.3.: Integration rules for the reference quadrilateral

D. Convergence analysis

The order of convergence p is defined as

$$||\Delta y(x,h)|| = y(x,h) - y(x) = \mathcal{O}(h^p), \tag{D.1}$$

where h is the discretisation stepsize, $y(x,h)$ is the numerical solution obtained at a discretisation step size h and $y(x)$ represents the analytical exact solution. Then $||\Delta y(x,h)||$ is called discretisation error. As the exact solution of the problem is required in order to compute the error in equation D.1, this kind of analysis is limited to some special test cases that can be solved analytically. The formal order p may be evaluated by comparison of two discretisation errors, obtained with two different step sizes h_1 and h_2:

$$\frac{||\Delta y(x,h_1)||}{||\Delta y(x,h_2)||} = \frac{h_1^p}{h_2^p}.$$

Applying logarithm yields

$$\ln \frac{||\Delta y(x,h_1)||}{||\Delta y(x,h_2)||} = p \ln \frac{h_1}{h_2},$$

thus p can be resolved as

$$p = \frac{\ln(||\Delta y(x,h_1)||) - \ln(||\Delta y(x,h_2)||)}{\ln \frac{h_1}{h_2}}.$$

For the onedimensional cases, we take the number of cells as grid resolution parameter and for or two-dimensional analysis, we introduce the grid resolution parameter "gridres"

$$\text{gridres} = \sqrt{\frac{\text{number of cells}}{\text{domain size}}},$$

resulting in the definition of formal order p

$$p = -\frac{\ln(||\Delta y(x,h_1)||) - \ln(||\Delta y(x,h_2)||)}{\ln(\text{gridres}_1) - \ln(\text{gridres}_2)}. \tag{D.2}$$

In equation (D.2) scalar error values are needed. Therefore, the discretisation error norm has to be of an integral nature. We first calculate the error norms in the solution domain Ω by summation over all the integral elemental errors Δu_E

$$
\begin{aligned}
||\Delta u_i||_1 &= \sum_E \int_E |\Delta u_E| dE \\
||\Delta u_i||_2 &= \sum_E \int_E [\Delta u_E]^2 dE \qquad 1 \leq i \leq n \\
||\Delta u_i||_\infty &= \max_{E \in \Omega} (|\Delta u_E|).
\end{aligned}
$$

Here, Δu_i represents a generic scalar variable of the conservative state error vector ΔU

$$\Delta U = (\Delta u_1 \; \Delta u_i \ldots \Delta u_n)^T .$$

The scalar error Δy, needed for the determination of the formal order p, is generated out of ΔU by the linear, quadratic and maximum norms

$$
\begin{aligned}
||\Delta y||_1 &= \frac{\sum_i ||\Delta u_i||_1}{n} \\
||\Delta y||_2 &= \frac{\sum_i ||\Delta u_i||_2}{n} \\
||\Delta y||_\infty &= \max_i (||\Delta u_i||_\infty).
\end{aligned}
$$

D.1. Convergence tables for the heat equation

The entry DOF in the table represents the number of degrees of freedom per state vector entry.

D.1.1. LDG scheme

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
20	20	6.398E-02	1.728E-04	2.584E-03	0.00	0.00	0.00
40	40	3.199E-02	8.564E-05	1.290E-03	1.00	1.01	1.00
80	80	1.599E-02	4.262E-05	6.444E-04	1.00	1.01	1.00
160	160	7.994E-03	2.127E-05	3.221E-04	1.00	1.00	1.00
320	320	3.997E-03	1.062E-05	1.611E-04	1.00	1.00	1.00
640	640	1.999E-03	5.309E-06	8.054E-05	1.00	1.00	1.00

Table D.1.: Convergence rates for P^0-elements, LDG fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
20	40	6.014E-03	1.161E-05	1.709E-04	0.00	0.00	0.00
40	80	1.511E-03	2.895E-06	4.269E-05	1.99	2.00	2.00
80	160	3.782E-04	7.232E-07	1.067E-05	2.00	2.00	2.00
160	320	9.457E-05	1.808E-07	2.667E-06	2.00	2.00	2.00

Table D.2.: Convergence rates for P^1-elements, LDG fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
5	15	9.095E-03	1.969E-05	2.766E-04	0.00	0.00	0.00
10	30	1.112E-03	2.401E-06	3.442E-05	3.03	3.04	3.01
20	60	1.423E-04	3.059E-07	4.303E-06	2.97	2.97	3.00
40	120	1.796E-05	3.814E-08	5.388E-07	2.99	3.00	3.00
80	240	2.285E-06	4.771E-09	6.785E-08	2.97	3.00	2.99

Table D.3.: Convergence rates for P^2-elements, LDG fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
5	20	4.854E-04	1.419E-06	2.110E-05	0.00	0.00	0.00
10	40	3.173E-05	9.154E-08	1.326E-06	3.94	3.95	3.99
20	80	1.962E-06	5.661E-09	8.300E-08	4.02	4.02	4.00
40	160	1.238E-07	3.530E-10	5.192E-09	3.99	4.00	4.00

Table D.4.: Convergence rates for P^3-elements, LDG fluxes

D.1.2. BR1 scheme

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
20	20	6.537E-02	1.924E-04	2.693E-03	0.00	0.00	0.00
40	40	3.217E-02	8.970E-05	1.304E-03	1.02	1.10	1.05
80	80	1.601E-02	4.353E-05	6.462E-04	1.01	1.04	1.01
160	160	7.997E-03	2.148E-05	3.224E-04	1.00	1.02	1.00
320	320	3.998E-03	1.068E-05	1.611E-04	1.00	1.01	1.00
640	640	1.999E-03	5.322E-06	8.054E-05	1.00	1.00	1.00

Table D.5.: Convergence rates for P^0-elements, BR1 fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
20	40	1.018E-02	2.605E-04	1.443E-03	0.00	0.00	0.00
40	80	5.101E-03	1.300E-04	7.216E-04	1.00	1.00	1.00
80	160	2.557E-03	6.497E-05	3.608E-04	1.00	1.00	1.00
160	320	1.279E-03	3.248E-05	1.804E-04	1.00	1.00	1.00

Table D.6.: Convergence rates for P^1-elements, BR1 fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
5	15	5.687E-03	1.698E-05	2.283E-04	0.00	0.00	0.00
10	30	5.174E-04	1.718E-06	2.334E-05	3.46	3.30	3.29
20	60	5.984E-05	1.999E-07	2.808E-06	3.11	3.10	3.06
40	120	7.375E-06	2.420E-08	3.479E-07	3.02	3.05	3.01

Table D.7.: Convergence rates for P^2-elements, BR1 fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
5	20	4.211E-04	1.954E-06	2.467E-05	0.00	0.00	0.00
10	40	5.588E-05	2.181E-07	2.858E-06	2.91	3.16	3.11
20	80	7.199E-06	2.517E-08	3.496E-07	2.96	3.12	3.03
40	160	9.075E-07	3.048E-09	4.349E-08	2.99	3.05	3.01

Table D.8.: Convergence rates for P^3-elements, BR1 fluxes

D.1.3. BR2 scheme

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
20	20	6.398E-02	4.801E-04	4.307E-03	0.00	0.00	0.00
40	40	3.199E-02	2.379E-04	2.150E-03	1.00	1.01	1.00
80	80	1.599E-02	1.184E-04	1.074E-03	1.00	1.01	1.00
160	160	7.994E-03	5.908E-05	5.369E-04	1.00	1.00	1.00
320	320	3.997E-03	2.951E-05	2.685E-04	1.00	1.00	1.00

Table D.9.: Convergence rates for P^0-elements, BR2 fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
20	40	7.125E-03	7.244E-05	5.746E-04	0.00	0.00	0.00
40	80	1.874E-03	1.948E-05	1.532E-04	1.93	1.89	1.91
80	160	4.775E-04	5.048E-06	3.941E-05	1.97	1.95	1.96
160	320	1.204E-04	1.284E-06	9.985E-06	1.99	1.97	1.98

Table D.10.: Convergence rates for P^1-elements, BR2 fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
5	15	1.011E-02	9.905E-05	8.399E-04	0.00	0.00	0.00
10	30	1.627E-03	2.143E-05	1.612E-04	2.64	2.21	2.38
20	60	2.211E-04	2.851E-06	2.315E-05	2.88	2.91	2.80
40	120	2.881E-05	3.716E-07	3.123E-06	2.94	2.94	2.89

Table D.11.: Convergence rates for P^2-elements, BR2 fluxes

Gridres	DOF	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
5	20	7.762E-04	5.844E-06	5.238E-05	0.00	0.00	0.00
10	40	5.096E-05	4.761E-07	4.095E-06	3.93	3.62	3.68
20	80	3.775E-06	3.559E-08	3.050E-07	3.75	3.74	3.75
40	160	2.580E-07	2.457E-09	2.086E-08	3.87	3.86	3.87

Table D.12.: Convergence rates for P^3-elements, BR2 fluxes

D.2. Convergence tables for the Gaussian pulse in density

The entry DOF in the table represents the number of degrees of freedom per state vector entry.

D.2.1. HLL flux

D.2.1.1. Quadrilaterals

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	32	0.000E+00	9.804E-02	5.346E-06	4.340E-04	0.00	0.00	0.00
16	128	1.000E-02	8.200E-02	5.576E-06	3.497E-04	0.26	-0.06	0.31
32	512	3.000E-02	7.240E-02	4.276E-06	2.879E-04	0.18	0.38	0.28
64	2048	9.000E-02	5.610E-02	2.843E-06	2.094E-04	0.37	0.59	0.46
128	8192	5.900E-01	3.843E-02	1.718E-06	1.358E-04	0.55	0.73	0.62
256	32768	4.500E+00	2.375E-02	9.748E-07	8.077E-05	0.69	0.82	0.75
512	131072	3.475E+01	1.340E-02	5.204E-07	4.437E-05	0.83	0.91	0.86

Table D.13.: Convergence rates for P^0-elements, regular quadrilateral grid, HLL flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	96	1.000E-02	5.428E-02	6.825E-06	2.762E-04	0.00	0.00	0.00
16	384	7.000E-02	2.895E-02	2.438E-06	1.298E-04	0.91	1.49	1.09
32	1536	2.100E-01	1.281E-02	6.410E-07	4.316E-05	1.18	1.93	1.59
64	6144	1.700E+00	3.038E-03	1.178E-07	8.890E-06	2.08	2.44	2.28
128	24576	1.435E+01	6.189E-04	2.179E-08	1.742E-06	2.30	2.43	2.35
256	98304	1.155E+02	1.376E-04	4.679E-09	3.883E-07	2.17	2.22	2.17
512	393216	9.247E+02	3.850E-05	1.119E-09	9.369E-08	1.84	2.06	2.05

Table D.14.: Convergence rates for P^1-elements, regular quadrilateral grid, HLL flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	192	6.000E-02	2.554E-02	3.795E-06	1.513E-04	0.00	0.00	0.00
16	768	2.500E-01	1.271E-02	5.589E-07	3.634E-05	1.01	2.76	2.06
32	3072	1.370E+00	3.380E-03	1.052E-07	7.073E-06	1.91	2.41	2.36
64	12288	1.150E+01	7.016E-04	1.531E-08	1.142E-06	2.27	2.78	2.63
128	49152	9.393E+01	1.142E-04	2.372E-09	1.850E-07	2.62	2.69	2.63
256	196608	7.560E+02	1.595E-05	3.424E-10	2.723E-08	2.84	2.79	2.76
512	786432	6.245E+03	2.054E-06	4.620E-11	3.697E-09	2.96	2.89	2.88

Table D.15.: Convergence rates for P^2-elements, regular quadrilateral grid, HLL flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	320	1.700E-01	1.693E-02	1.606E-06	7.705E-05	0.00	0.00	0.00
16	1280	8.800E-01	3.963E-03	2.240E-07	1.200E-05	2.10	2.84	2.68
32	5120	7.140E+00	4.955E-04	1.938E-08	1.240E-06	3.00	3.53	3.27
64	20480	5.901E+01	2.781E-05	1.204E-09	7.737E-08	4.16	4.01	4.00
128	81920	4.774E+02	1.734E-06	8.448E-11	5.425E-09	4.00	3.83	3.83
256	327680	3.862E+03	1.102E-07	7.522E-12	3.706E-10	3.98	3.49	3.87

Table D.16.: Convergence rates for P^3-elements, regular quadrilateral grid, HLL flux

D.2.1.2. Triangles

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	32	0.000E+00	9.820E-02	7.244E-06	5.432E-04	0.00	0.00	0.00
16	128	2.000E-02	8.647E-02	8.114E-06	4.788E-04	0.18	-0.16	0.18
32	512	1.100E-01	6.912E-02	5.902E-06	3.350E-04	0.32	0.46	0.52
64	2048	4.200E-01	5.127E-02	3.844E-06	2.350E-04	0.43	0.62	0.51
128	8192	3.680E+00	3.541E-02	2.289E-06	1.495E-04	0.53	0.75	0.65
256	32768	3.329E+01	2.136E-02	1.268E-06	8.653E-05	0.73	0.85	0.79
512	131072	2.686E+02	1.186E-02	6.710E-07	4.695E-05	0.85	0.92	0.88

Table D.17.: Convergence rates for P^0-elements, regular triangular grid, HLL flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	96	2.000E-02	7.586E-02	9.630E-06	4.027E-04	0.00	0.00	0.00
16	384	1.400E-01	2.189E-02	4.190E-06	1.756E-04	1.79	1.20	1.20
32	1536	4.800E-01	1.863E-02	9.826E-07	5.843E-05	0.23	2.09	1.59
64	6144	3.190E+00	5.669E-03	2.962E-07	1.869E-05	1.72	1.73	1.64
128	24576	2.939E+01	1.787E-03	7.333E-08	4.825E-06	1.67	2.01	1.95
256	98304	2.397E+02	4.460E-04	1.826E-08	1.210E-06	2.00	2.01	2.00
512	393216	1.926E+03	1.061E-04	4.554E-09	3.032E-07	2.07	2.00	2.00

Table D.18.: Convergence rates for P^1-elements, regular triangular grid, HLL flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	192	8.000E-02	2.124E-02	6.948E-06	2.063E-04	0.00	0.00	0.00
16	768	2.900E-01	1.760E-02	1.218E-06	6.552E-05	0.27	2.51	1.65
32	3072	1.810E+00	5.788E-03	2.228E-07	1.405E-05	1.60	2.45	2.22
64	12288	1.570E+01	1.471E-03	3.801E-08	2.692E-06	1.98	2.55	2.38
128	49152	1.308E+02	2.252E-04	6.715E-09	4.711E-07	2.71	2.50	2.51
256	196608	1.054E+03	3.075E-05	1.114E-09	7.514E-08	2.87	2.59	2.65
512	786432	8.597E+03	3.785E-06	2.495E-10	9.944E-09	3.02	2.16	2.92

Table D.19.: Convergence rates for P^2-elements, regular triangular grid, HLL flux

Gridr.	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	320	1.900E-01	7.440E-03	3.393E-06	8.849E-05	0.00	0.00	0.00
16	1280	8.600E-01	5.738E-03	5.315E-07	2.729E-05	0.37	2.67	1.70
32	5120	6.880E+00	1.052E-03	4.411E-08	2.784E-06	2.45	3.59	3.29
64	20480	5.870E+01	1.392E-04	4.117E-09	3.005E-07	2.92	3.42	3.21
128	81920	4.745E+02	1.136E-05	3.363E-10	2.365E-08	3.61	3.61	3.67
256	327680	3.808E+03	6.335E-07	1.886E-11	1.230E-09	4.16	4.16	4.26
512	1310720	3.101E+04	3.616E-08	1.226E-12	7.976E-11	4.13	3.94	3.95

Table D.20.: Convergence rates for P^3-elements, regular triangular grid, HLL flux

D.2.2. ROE flux

D.2.2.1. Quadrilaterals

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	32	0.000E+00	9.649E-02	4.875E-06	4.225E-04	0.00	0.00	0.00
16	128	1.000E-02	5.606E-02	2.975E-06	2.416E-04	0.78	0.71	0.81
32	512	3.000E-02	3.734E-02	1.582E-06	1.295E-04	0.59	0.91	0.90
64	2048	2.100E-01	2.215E-02	8.346E-07	6.982E-05	0.75	0.92	0.89
128	8192	1.490E+00	1.153E-02	4.293E-07	3.645E-05	0.94	0.96	0.94
256	32768	1.211E+01	5.918E-03	2.193E-07	1.874E-05	0.96	0.97	0.96
512	131072	9.457E+01	2.990E-03	1.106E-07	9.487E-06	0.99	0.99	0.98

Table D.21.: Convergence rates for P^0-elements, regular quadrilateral grid, Roe flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	96	1.000E-02	4.257E-02	5.201E-06	2.616E-04	0.00	0.00	0.00
16	384	1.300E-01	2.572E-02	1.193E-06	8.969E-05	0.73	2.12	1.54
32	1536	4.800E-01	1.048E-02	3.663E-07	2.891E-05	1.30	1.70	1.63
64	6144	3.130E+00	3.867E-03	9.692E-08	7.961E-06	1.44	1.92	1.86
128	24576	2.553E+01	1.069E-03	2.485E-08	2.079E-06	1.85	1.96	1.94
256	98304	2.048E+02	2.804E-04	6.279E-09	5.297E-07	1.93	1.98	1.97
512	393216	1.640E+03	7.132E-05	1.578E-09	1.336E-07	1.98	1.99	1.99

Table D.22.: Convergence rates for P^1-elements, regular quadrilateral grid, Roe flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	192	9.000E-02	2.735E-02	2.723E-06	1.417E-04	0.00	0.00	0.00
16	768	2.500E-01	1.365E-02	4.132E-07	3.103E-05	1.00	2.72	2.19
32	3072	2.040E+00	2.762E-03	6.626E-08	5.290E-06	2.31	2.64	2.55
64	12288	1.681E+01	3.883E-04	8.608E-09	7.116E-07	2.83	2.94	2.89
128	49152	1.364E+02	5.540E-05	1.099E-09	9.224E-08	2.81	2.97	2.95
256	196608	1.104E+03	7.121E-06	1.388E-10	1.172E-08	2.96	2.98	2.98
512	786432	8.825E+03	8.972E-07	1.756E-11	1.475E-09	2.99	2.98	2.99

Table D.23.: Convergence rates for P^2-elements, regular quadrilateral grid, Roe flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	320	3.600E-01	1.431E-02	1.201E-06	7.247E-05	0.00	0.00	0.00
16	1280	1.170E+00	3.613E-03	1.382E-07	1.088E-05	1.99	3.12	2.74
32	5120	8.790E+00	3.551E-04	1.018E-08	8.019E-07	3.35	3.76	3.76
64	20480	7.187E+01	3.589E-05	6.587E-10	5.439E-08	3.31	3.95	3.88
128	81920	5.785E+02	2.433E-06	4.397E-11	3.511E-09	3.88	3.91	3.95
256	327680	5.027E+03	1.582E-07	4.700E-12	2.240E-10	3.94	3.23	3.97

Table D.24.: Convergence rates for P^3-elements, regular quadrilateral grid, Roe flux

D.2.2.2. Triangles

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	32	0.000E+00	9.818E-02	7.232E-06	5.430E-04	0.00	0.00	0.00
16	128	6.000E-02	7.976E-02	6.253E-06	4.315E-04	0.30	0.21	0.33
32	512	3.200E-01	3.969E-02	2.974E-06	2.030E-04	1.01	1.07	1.09
64	2048	1.260E+00	3.213E-02	1.569E-06	1.107E-04	0.31	0.92	0.87
128	8192	9.760E+00	1.963E-02	8.601E-07	6.169E-05	0.71	0.87	0.84
256	32768	8.210E+01	1.023E-02	4.459E-07	3.216E-05	0.94	0.95	0.94
512	131072	6.593E+02	5.190E-03	2.244E-07	1.622E-05	0.98	0.99	0.99

Table D.25.: Convergence rates for P^0-elements, regular triangular grid, Roe flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	96	4.000E-02	7.371E-02	8.598E-06	3.905E-04	0.00	0.00	0.00
16	384	2.100E-01	2.017E-02	2.905E-06	1.489E-04	1.87	1.57	1.39
32	1536	7.800E-01	2.770E-02	6.556E-07	4.501E-05	-0.46	2.15	1.73
64	6144	6.390E+00	1.162E-02	2.318E-07	1.663E-05	1.25	1.50	1.44
128	24576	5.511E+01	3.395E-03	6.336E-08	4.636E-06	1.77	1.87	1.84
256	98304	4.554E+02	9.046E-04	1.607E-08	1.185E-06	1.91	1.98	1.97
512	393216	3.718E+03	2.300E-04	4.019E-09	2.970E-07	1.98	2.00	2.00

Table D.26.: Convergence rates for P^1-elements, regular triangular grid, Roe flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	192	1.300E-01	2.124E-02	5.896E-06	1.875E-04	0.00	0.00	0.00
16	768	5.400E-01	3.328E-02	7.710E-07	6.481E-05	-0.65	2.93	1.53
32	3072	2.760E+00	8.021E-03	1.894E-07	1.458E-05	2.05	2.03	2.15
64	12288	2.284E+01	1.482E-03	2.622E-08	1.977E-06	2.44	2.85	2.88
128	49152	1.872E+02	1.754E-04	3.794E-09	2.795E-07	3.08	2.79	2.82
256	196608	1.517E+03	2.275E-05	5.800E-10	3.542E-08	2.95	2.71	2.98
512	786432	1.212E+04	2.749E-06	1.832E-10	4.688E-09	3.05	1.66	2.92

Table D.27.: Convergence rates for P^2-elements, regular triangular grid, Roe flux

Gridres	DOF	CPU time	L_1	L_2	L_∞	$\mathcal{O}(L_1)$	$\mathcal{O}(L_2)$	$\mathcal{O}(L_\infty)$
8	320	2.600E-01	1.764E-02	2.439E-06	8.482E-05	0.00	0.00	0.00
16	1280	1.170E+00	1.029E-02	5.210E-07	3.182E-05	0.78	2.23	1.41
32	5120	8.360E+00	6.911E-04	2.914E-08	2.016E-06	3.90	4.16	3.98
64	20480	7.079E+01	1.204E-04	2.616E-09	2.024E-07	2.52	3.48	3.32
128	81920	5.709E+02	7.671E-06	1.946E-10	1.479E-08	3.97	3.75	3.77
256	327680	4.611E+03	5.585E-07	1.229E-11	9.391E-10	3.78	3.99	3.98
512	1310720	3.687E+04	3.433E-08	7.645E-13	5.819E-11	4.02	4.01	4.01

Table D.28.: Convergence rates for P^3-elements, regular triangular grid, Roe flux

E. NACA0012 airfoil

The contour line of the NACA0012 airfoil can described by the analytical function

$$y(x) = \frac{12}{20}\left(0.2969\sqrt{x} - 0.126\,x - 0.3516\,x^2 + 0.2843\,x^3 - 0.1015\,x^4\right) \tag{E.1}$$

where x stands for the normalised chord length. The thickness of the airfoil at length $x = 1$ is 0.00126. In order to simplify (structured) grid generation, we specified the chord length l to $l = 1.00893$, resulting in a sharp trailing edge of the airfoil ($y(x = 1.00893) \approx 0$).

Bibliography

[1] www.osl.iu.edu/research/itl/.

[2] www.osl.iu.edu/research/mtl/.

[3] www.top500.org.

[4] M. Ainsworth. Dispersive and dissipative behaviour of high order discontinuous Galerkin finite element methods. *Journal of Computational Physics*, 198(1):106–130, 2004.

[5] A. Altmikus, S. Wagner, P. Beaumier, and G. Servera. A comparison: Weak versus strong modular coupling for trimmed aeroelastic rotor simulations. *American Helicopter Society 58th Annual Forum*, 2002.

[6] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM J. Numer. Anal.*, 19(4):742–760, 1982.

[7] D. N. Arnold, F. Brezzi, B. Cockburn, and D. Marini. Discontinuous Galerkin methods for elliptic problems. In B. Cockburn, G. E. Karniadakis, and C.-W. Shu, editors, *Discontinuous Galerkin Methods*, volume 11 of *LNCSE*, pages 89–101. Springer, 2000.

[8] D. N. Arnold, F. Brezzi, B. Cockburn, and D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779, 2001.

[9] H. L. Atkins and C.-W. Shu. Quadrature-free implementation of discontinuous Galerkin methods for hyperbolic equations. *AIAA Journal*, 36:775–782, 1998.

[10] H. L. Atkins and C.-W. Shu. Analysis of the Discontinuous Galerkin Method Applied to the Diffusion Operator. *AIAA Paper 99-3306*, 1999.

[11] G. A. Baker. Finite element methods for elliptic equations using nonconforming elements. *Math. Comput.*, 31:45–59, 1977.

[12] T. Barth and P.O. Frederickson. Higher order solution of the Euler equations on unstructured grids using quadratic reconstruction. *AIAA Paper 90-0013*, 1990.

[13] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. *Journal of Computational Physics*, 131(2), 1997.

[14] F. Bassi and S. Rebay. GMRES for discontinuous Galerkin solution of the compressible Navier-Stokes equations. In B. Cockburn, G. E. Karniadakis, and C.-W. Shu, editors, *Discontinuous Galerkin Methods*, pages 197–208. Springer, 2000.

[15] F. Bassi and S. Rebay. A high order discontinuous Galerkin method for compressible turbulent flows. In B. Cockburn, G. E. Karniadakis, and C.-W. Shu, editors, *Discontinuous Galerkin Methods*, pages 77–88. Springer, 2000.

[16] F. Bassi and S. Rebay. Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k-ω turbulence model equations. *Journal of Computers & Fluids*, 34:507–540, 2005.

[17] C. E. Baumann and J. T. Oden. A discontinuous *hp* finite element method for the Euler and the Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 31:79–95, 1999.

[18] Y. Benarafaa, O. Cionia, F. Ducrosa, and P. Sagaut. RANS/LES coupling for unsteady turbulent flow simulation at high Reynolds number on coarse meshes. *Computer Methods in Applied Mechanics and Engineering*, 195(23–24):2939–2960, 2006.

[19] J. A. Benek, P. G. Buning, and J. L. Steger. A 3-D Chimera grid embedding technique. *AIAA–Paper 85-1523*, 1985.

[20] M. Blanco and D.W. Zingg. An Unstructured Newton-Krylov Algorithm with a Loosely-Coupled Turbulence Model for Aerodynamic Flows. *44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada*, AIAA Paper 2006-691, 2006.

[21] D.L. Bonhaus. A High-Order Accurate Finite Element Method for Viscous Compressible Flows. *Ph.D. Dissertation, Aerospace and Ocean Engineering, Virgina Tech University*, Dezember, 1998.

[22] A. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulation for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32:199–259, 1982.

[23] A. Burbeau, P. Sagaut, and Ch. H. Bruneau. A problem-Independent Limiter for High-Order Runge-Kutta Discontinuous Galerkin Methods. *Journal of Computational Physics*, 169(1):111–150, May 2001.

[24] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2nd edition, 2003.

[25] F.K. Chow and P. Moin. A further study of numerical errors in large-eddy simulations. *Journal of Computational Physics*, 184(2):366–380, 2003.

[26] B. Cockburn, S. Hou, and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case. *Math. Comput.*, 54:545–581, 1990.

[27] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. The development of discontinuous Galerkin methods. In B. Cockburn, G. E. Karniadakis, and C.-W. Shu, editors, *Discontinuous Galerkin Methods*, pages 3–50. Springer, 2000.

[28] 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. *J. Comput. Phys.*, 84:90–113, 1989.

[29] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework. *Math. Comput.*, 52:411–435, 1989.

[30] B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.*, 35:2440–2463, 1998.

[31] B. Cockburn and C.-W. Shu. The Runge-Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems. *J. Comput. Phys.*, 141:199–224, 1998.

[32] B. Cockburn and C.-W. Shu. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *SIAM J. Sci. Comput.*, 16:173–261, 2001.

[33] B. Cockburn and C.W. Shu. The local discontinuous galerkin finite element method for convection-diffusion systems. *SIAM Journal of Numerical Analysis*, 35:337–361, 1998.

[34] S. Collis. Discontinuous Galerkin methods for turbulence simulation. Technical report, Center for Turbulence Research, 2002.

[35] M. Delanaye and Y. Liu. Quadratic reconstruction finite volume schemes on 3d arbitrary unstructured polyhedral grids. *AIAA Paper, 99 (3259)*, 1999.

[36] M. Dietz, E. Krämer, and S. Wagner. Tip vortex conservation on a main rotor in slow descent flight using vortex-adapted Chimera grids. *24th AIAA Applied Aerodynamics Conference, San Francisco, CA*, AIAA–Paper 2006-3478, 2006.

[37] M. Dietz, E. Krämer, S. Wagner, and A. Altmikus. Weak coupling for active advanced rotors. *Proceedings of the 31st European Rotorcraft Forum, Florence, Italy*, 2005.

[38] V. Dolejsi. On the discontinuous Galerkin method for the numerical solution of the Euler and the Navier-Stokes equations. *International Journal of Numerical Methods Fluids*, 2002.

[39] J. Douglas and T. F. Dupont. *Interior penalty procedures for elliptic and parabolic Galerkin methods*, volume 58 of *Lecture Notes in Physics*. Springer, Berli, 1976.

[40] M. Drela and M.B. Giles. Viscous-inviscid analysis of transonic and low Reynolds number airfoils. *AIAA Journal,*, 25(10):1347–1355, 1987.

[41] M. Dumbser. *Arbitrary High Order Schemes for the Solution of Hyperbolic Conservation Laws in Complex Domains*. Dissertation, Institut für Aerodynamik und Gasdynamik, Universität Stuttgart, 2005.

[42] E.P.N. Duque, R.C. Strawn, and R. Biswas. A Solution Adaptive Structured/Unstructured Overset Grid Flow Solver with Applications to Helicopter Rotor Flows. *13th AIAA Applied Aerodynamics Conference, San Diego, CA*, 1995.

[43] A. Favre. Equation des gaz turbulents compressibles. *Journal de Mécanique*, 4(3), 1965.

[44] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer-Verlag, Berlin, 2001.

[45] K.J. Fidkowski, T.A. Oliver, J. Lu, and D.L. Darmofal. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 207(1):92–113, 2005.

[46] S.K. Godunov. A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sb.*, 1959.

[47] W. Haase, E. Chaput, E. Elsholz, M. Leschziner, and U. Müller. *ECARP - European Computational Aerodynamics Research Project: Validation of CFD Codes and Assessment of Turbulence Models*, volume 58 of *Notes on Numerical Fluid Mechanics*. Viehweg, 1997.

[48] A. Harten, P.D. Lax, and B. van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM review*, 25:35–61, 1983.

[49] R. Hartmann. Adaptive discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 51:1131–1156, 2006.

[50] R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier-Stokes equations I: Method formulation. *Int. J. Numer. Anal. Model.*, 3(1):1–20, 2006.

[51] R. Hartmann and P. Houston. Symmetric interior penalty DG methods for the compressible Navier-Stokes equations II: Goal-oriented a posteriori error estimation. *Int. J. Numer. Anal. Model.*, 3(2):141–162, 2006.

[52] C. Hirsch. *Numerical Computation of Internal and External Flows, Vol. 1.* John Wiley & Son, Ltd. Chichester, 1988.

[53] C. Hirsch. *Numerical Computation of Internal and External Flows, Vol. 2.* John Wiley & Son, Ltd. Chichester, 1990.

[54] F.Q. Hu, M. Y. Hussaini, and P. Rasetarinera. An analysis of the discontinuous Galerkin method for wave propagation problems. *Journal of Computational Physics*, 151(2):921–946, 1999.

[55] O. Inoue and N. Hatakeyama. Sound generation by a two-dimensional circular cylinder in a uniform flow. *J. of Fluid Mechanics*, 471:285–314, 2002.

[56] G. E. Karniadakis and S. J. Sherwin, editors. *Spectral/hp Element Methods in CFD.* Oxford University Press, 1999.

[57] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1998.

[58] C.M. Klaij, J.J.W. van der Vegt, and H. van der Ven. Pseudo-time stepping methods for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 219(2):622–643, 2006.

[59] C.M. Klaij, J.J.W. van der Vegt, and H. van der Ven. Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 217(2):589–611, 2006.

[60] M. Kloker. A robust high-resolution split-type compact FD scheme for spatial DNS of boundary-layer transition. *Appl. Sci. Res.*, 59, 1998.

[61] A. G. Kravchenko and P. Moin. On the Effect of Numerical Errors in Large Eddy Simulations of Turbulent Flows. *Journal of Computational Physics*, 131(2):310–322, 1997.

[62] N. Kroll, C.-C. Rossow, K. Becker, and F. Thiele. MEGAFLOW - A numerical flow simulation system. *ICAS-Paper 98-2.7.4*, 21st ICAS Congress, Melbourne, September 1998.

[63] B. Landmann and S. Wagner. Aeroelastic analysis of helicopter rotor blades. *22nd Applied Aerodynamics Conference and Exhibit, Providence, Rhode Island*, AIAA–Paper 2004-5286, 2004.

[64] P.D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Comm. Pure. Appl. Math.*, 1954.

[65] P. Le Saint and P.-A. Raviart. On a finite element method for solving the neutron transport equation. In C. de Boor, editor, *Mathematical aspects of finite elements in partial differential equations*, pages 89–123, New York, 1974. Academic Press.

[66] B. Van Leer. Upwind-difference methods for aerodynamic problems governed by the Euler equations. *in Large-Scale Computations in Fluid Mechanics, Lectures in Applied Mathematics*, 22, 1985.

[67] S.K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103:16–42, 1992.

[68] J.W. Lim and R. Strawn. Prediction of HART II Rotor BVI Loading and Wake System Using CFD/CSI Loose Coupling. *45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada*, AIAA Paper 2007-1281, 2007.

[69] Y. Liu, M. Vinokur, and Z.J. Wang. Spectral difference method for unstructured grids I: Basic Formulation. *Journal of Computational Physics*, 216:780–801, 2006.

[70] D.J. Mavriplis and A. Jameson. Multigrid solution of the Navier-Stokes equations on triangular meshes. *AIAA-Paper 89-0120*, 1989.

[71] F.R. Menter. Zonal Two Equation k-ω Turbulence Models for Aerodynamic Flows. *AIAA Paper 93-2906*, 1993.

[72] C.R. Nastase and D.J. Mavriplis. High-order discontinuous Galerkin methods using an hp-multigrid approach. *Journal of Computational Physics*, 213(1):330–357, 2006.

[73] J. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei der Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. *Abh. Math. Sem. Univ. Hamburg*, 36:9–15, 1971.

[74] J. T. Oden and C. E. Baumann. A conservative DGM for convection-diffusion and Navier-Stokes problems. In B. Cockburn, G. E. Karniadakis, and C.-W. Shu, editors, *Discontinuous Galerkin Methods*, pages 179–196. Springer, 2000.

[75] T.A. Oliver. Multigrid solution for high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Master's Thesis, MIT, Department of Aeronautics and Astronautics*, August 2004.

[76] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous galerkin methods. *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit, January 2006*, AIAA-Paper 2006-0112, 2006.

[77] H. Pomin and S. Wagner. Navier-Stokes analysis of helicopter rotor aerodynamics in hover and forward flight. *Journal of Aircraft*, 39(5):813–821, 2002.

[78] H. Pomin and S. Wagner. Aeroelastic analysis of helicopter rotor blades on deformable chimera grids. *Journal of Aircraft*, 41(3):577–584, 2004.

[79] M. Potsdam. Dynamic rotorcraft applications using overset grids. *Proceedings of the 31st European Rotorcraft Forum, Florence, Italy*, 2005.

[80] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 2nd edition, 1992.

[81] J. Qiu and C.-W. Shu. Runge-Kutta discontinuous Galerkin method using WENO limiters. *SIAM J. of. Sci. Comput.*, 26:907–929, 2005.

[82] P. Rasetarinera and M. Y. Hussaini. An Efficient Implicit Discontinuous Spectral Galerkin Method. *Journal of Computational Physics*, 172(2):718–738, 2001.

[83] W.H. Reed and T.R. Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Laboratory, 1973.

[84] P.L. Roe. Approximate Riemann solvers, parameter vectors and difference schemes. *J. Comput. Physics*, 43:357–372, 1981.

[85] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, USA, 2003.

[86] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:865, 1986.

[87] P. Sagaut. *Large Eddy Simulation for Incompressible Flows*. Springer, 2002.

[88] C. Schwab. *hp*-FEM for fluid flow simulation. In T. J. Barth and H. Deconinck, editors, *High-Order Methods for Computational Physics*, pages 325–438. Springer Verlag, 1999.

[89] T. Schwartzkopff, C.D. Munz, E.F. Toro, and R.C. Millington. ADER: A high order approach for linear hyperbolic systems in 2d. *Journal of Scientific Computing*, 17:231–240, 2002.

[90] P. R. Spalart, W-H. Jou, M. Strelets, and S.R. Allmaras. Comments on the feasibility of LES for wings, and on a hybrid RANS/LES approach. In *Advances in DNS/LES*, 1997.

[91] P.R. Spalart and S.R. Allmaras. A one-equation turbulence model for aerodynamic flows. Technical report, AIAA-92-0439, 1992.

[92] P.R. Spalart and S.R. Allmaras. A one-equation turbulence model for aerodynamic flows. *La Recherche Aerospatiale*, 1, 1994.

[93] R.C. Strawn and T. Barth. Helicopter Rotor Flows using Unstructured Grid Scheme. *American Helicopter Society Journal*, 38(2):61–67, 1993.

[94] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer, 2nd edition, 1999.

[95] E. Truckenbrodt. *Fluidmechnanik, Band 1: Elementare Strömungsvorgänge dichteveränderlicher Fluide sowie Potential- und Grenzschichtströmungen*. Springer-Verlag, Berlin; Heidelberg; New York, 1980.

[96] P. G. Tucker. Differential equation-based wall distance computation for DES and RANS. *Journal of Computational Physics*, 190(1):229–248, 2003.

[97] J. J. W. van der Vegt and H. van der Ven. Discontinuous Galerkin finite element method with anisotropic local grid refinement for inviscid compressible flows. *J. Comput. Phys.*, 141(1):46–77, 1998.

[98] J.J.W. van der Vegt and H. van der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. part I. general formulation. *J. Comput. Phys.*, 182:546–585, 2002.

[99] J.J.W. van der Vegt and H. van der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. part II. efficient flux quadrature. *Comput. Meth. Appl. Engrg.*, 191:4747–4780, 2002.

[100] H.A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 12:631–644, 1992.

[101] V. Venkatakrishnan, S. Allmaras, D. Kamenetskii, and F. Johnson. Higher Order Schemes for the Compressible Navier-Stokes Equations. *AIAA-2003-3987*, 2003.

[102] M.R. Visbal and D.V. Gaitonde. On the Use of High-Order Finite.Difference Schemes on Curvilinear and Deforming Meshes. *Journal of Computational Physics*, 181:155–185, 2002.

[103] J.B. Vos, A. Rizzi, D. Darracq, and E.H. Hirschel. Navier-Stokes solvers in European aircraft design. *Progress in Aerospace Sciences*, 38(8):601–697, 2002.

[104] T.C. Warburton. *Spectral/hp methods on ploymorphic multi-domains: Algorithms and Applications*. PhD thesis, Brown University, 1998.

[105] P. Wassermann and M. Kloker. Transition mechanisms induced by traveling cross-flow vortices in a three-dimensional boundary layer. *J. Fluid Mech.*, 483:67–89, 2003.

[106] M. F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM J. Numer. Anal.*, 15(1):152–161, 1978.

[107] D. Wilcox. Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal*, 26(11):1299–1310, 1988.

[108] C.H.K. Williamson. Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers. *J. of Fluid Mechanics*, 206:579–627, 1989.

[109] H.C. Yee. Explicit and Implicit Multidimensional Compact High-Resolution Shock-Capturing Methods: Formulation. *Journal of Computational Physics*, 131:216–232, 1997.

[110] M. Zhang and C.-W. Shu. An analysis of three different formulations of the discontinuous Galerkin method for diffusion equations. *Mathematical Models and Methods in Applied Sciences*, 13(3):395–413, 2003.

Lebenslauf

Persönliche Daten

Name:	Björn Landmann
Geburtsdatum:	18.02.1974
Geburtsort:	Kandel

Beruf

seit 08/06	Post-doctoral Research Engineer am Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS), Toulouse, Frankreich
01/01 - 07/06	Wissenschaftlicher Mitarbeiter am Institut für Aero- und Gasdynamik an der Universität Stuttgart

Studium

10/94 - 12/00	Studium der Luft- und Raumfahrttechnik an der Universität Stuttgart mit den Vertiefungsrichtungen Strömungslehre und Regelungstechnik
10/99 - 04/00	Studienarbeit im Fach Regelungstechnik am Institut für Flugmechanik u. Regelungstechnik der Universität Stuttgart "Aufbau und Validierung eines Hubschrauberrotormodels"
05/00 - 12/00	Diplomarbeit im Fach Strömungslehre bei EADS-Militärflugzeuge, "Validierung einer unstrukt. Navier-Stokes Methode zur Simulation der turbulenten Strömung um ausgewählte Flügelkonfigurationen"

Grundwehrdienst

07/1993 - 06/1994	ABC-Abwehrbataillon 750, Bruchsal

Schule

08/1984 - 05/1993	Goethe-Gymnasium Gaggenau
09/1980 - 08/1984	Hebelschule Gaggenau (Grundschule)